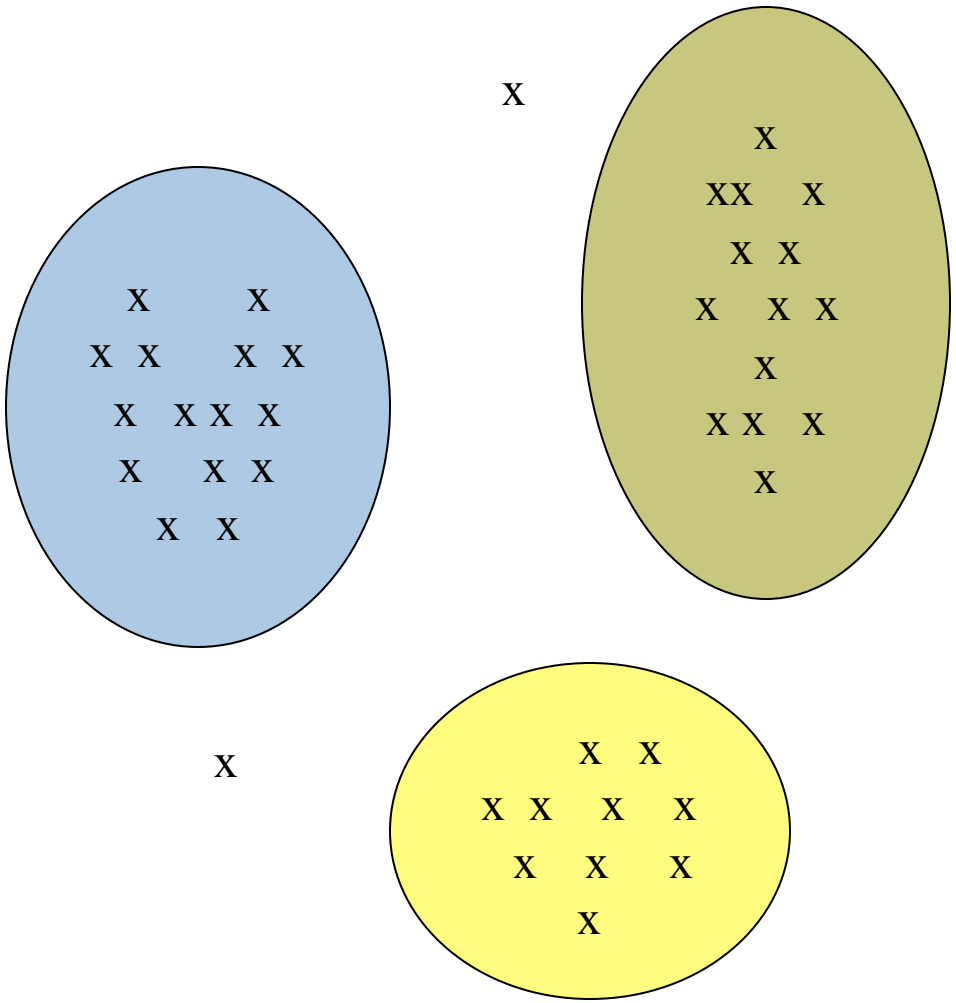
Clustering

The Problem of Clustering

- Given a set of points, with a notion of distance between points, group the points into some number of *clusters*, so that
  - members of a cluster are close/similar to each other
  - Members of different clusters are dissimilar
- Usually:
  - points are in a high---dimensional space
  - similarity is defined using a distance measure
    - Euclidean, Cosine, Jaccard, edit distance, …

Example: Clusters



Application: SkyCat

- A catalog of 2 billion "sky objects" represents objects by their radiation in 7 dimensions (frequency bands).
- Problem: Cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.
- Sloan Sky Survey is a newer, better version.

Clustering Movies

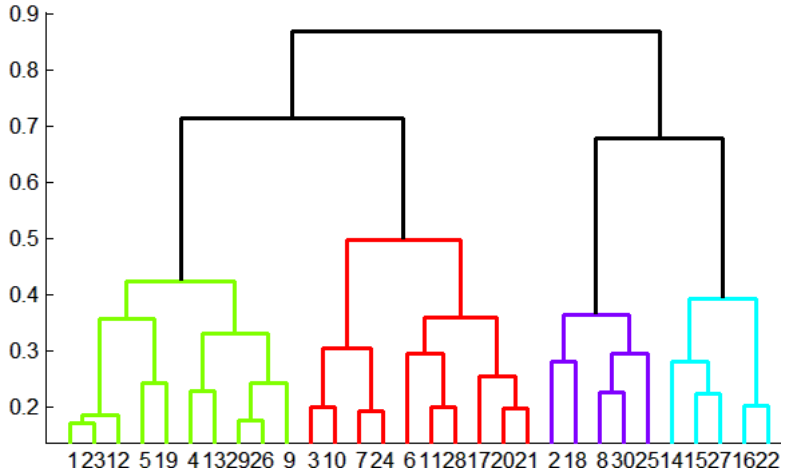
- Intuitively: Movies divide into categories, and customers prefer a few categories.
  - But what are categories really?
- Represent a movie by the customers who bought/rated it
- Similar movies have similar sets of customers, and vice-versa
- Space of all movies:
  - Think of a space with one dimension for each customer.
    - Values in a dimension may be 0 or 1 only.
    - A movies's point in this space is $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} customer bought the movie.

Cosine, Jaccard, and Euclidean

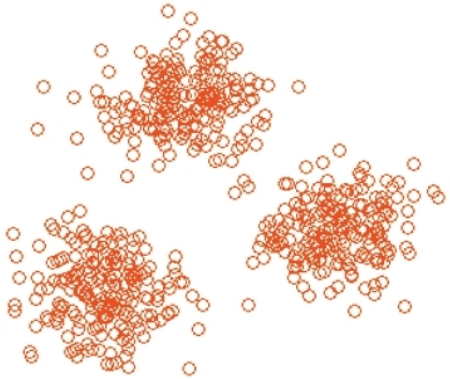
- We have a choice:
  1. Sets as vectors: measure similarity by the cosine distance.
  2. Sets as sets: measure similarity by the Jaccard distance.
  3. Sets as points: measure similarity by Euclidean distance.

Methods of Clustering

- Hierarchical:
  - Agglomerative (bottom up):
    - Initially, each point is a cluster
    - Repeatedly combine the two "nearest" clusters into one.
  - Divisive (top down):
    - Start with one cluster and recursively split it



- Point Assignment:
  - Maintain a set of clusters
  - Points belong to "nearest" cluster



Hierarchical Clustering

- Key operation:
  Repeatedly combine two nearest clusters

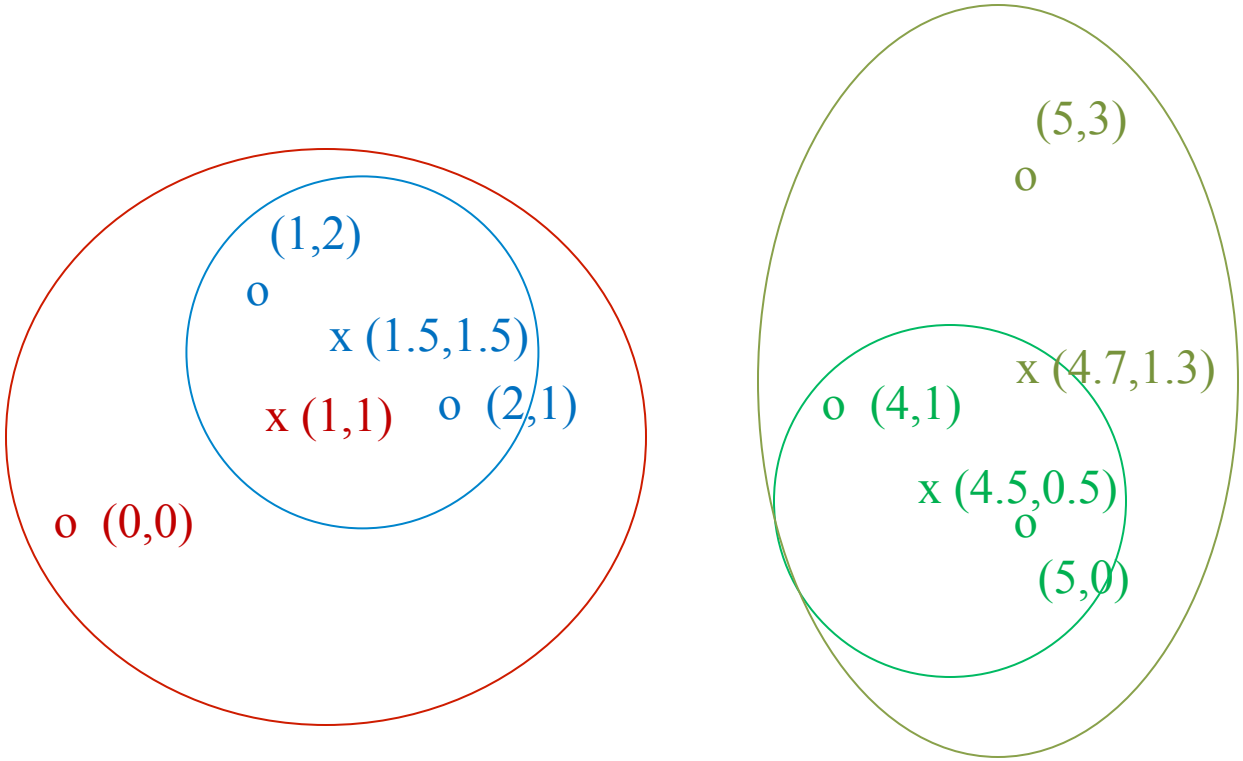
- Three important questions:
  1. How do you represent a cluster of more than one point?
  2. How do you determine the "nearness" of clusters?
  3. When to stop combining clusters?

Hierarchical Clustering – (2)

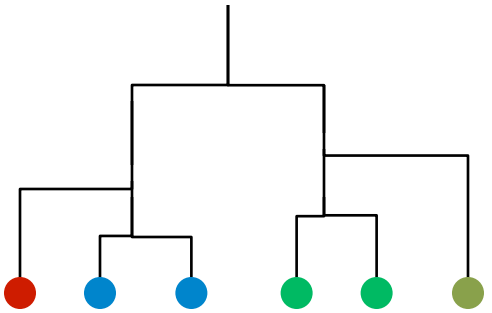
- Key problem: as you build clusters, how do you represent the location of each cluster, to tell which pair of clusters is closest?

- Euclidean case: each cluster has a *centroid* = average of its points
  - Measure cluster distances by distances of centroids

Example: Hierarchical clustering



Data



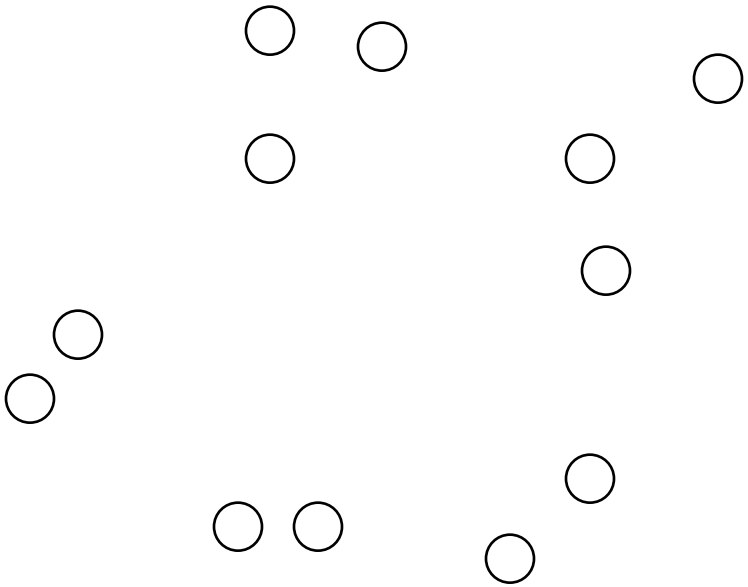
Dendrogram

Agglomerative clustering algorithm

- Most popular hierarchical clustering technique
- Basic algorithm
  1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the distance matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the distance between two clusters
  - Different definitions of the distance between clusters lead to different algorithms

Input/ Initial setting

- Start with clusters of individual points and a distance/proximity matrix

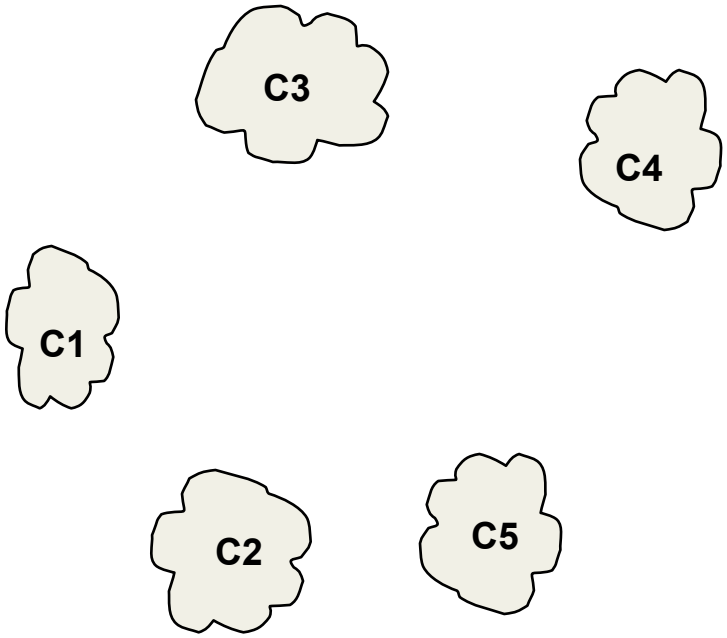


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						

Distance/Proximity Matrix

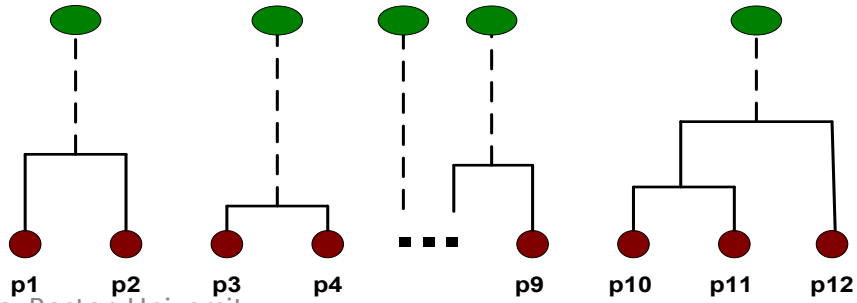
Intermediate State

- After some merging steps, we have some clusters



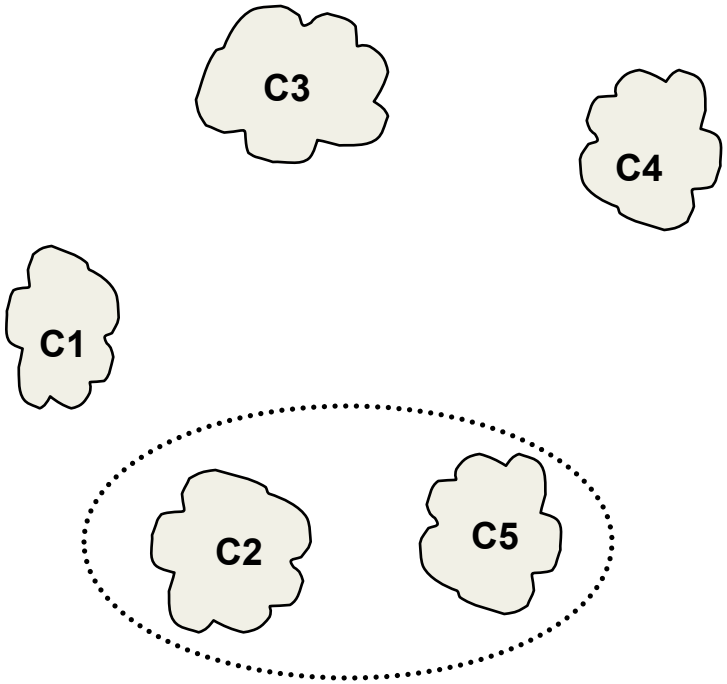
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix



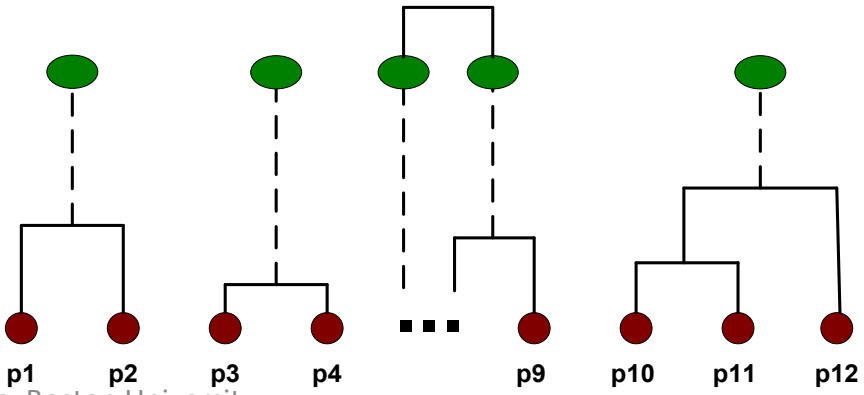
Intermediate State

- Merge the two closest clusters (C2 and C5) and update the distance matrix.



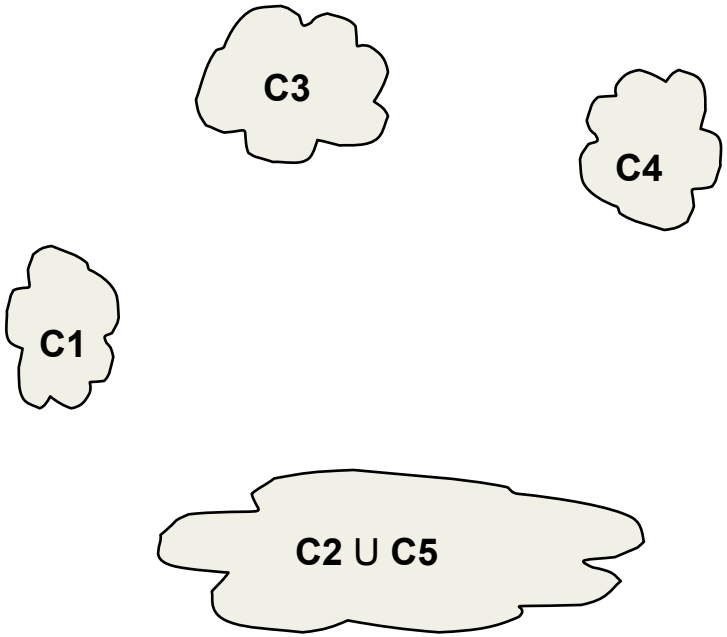
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix

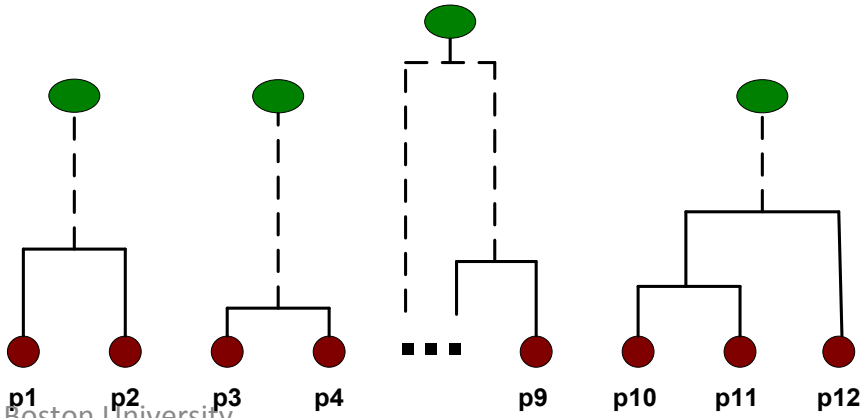


After Merging

- "How do we update the distance matrix?"



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		



Distance between two clusters

- Each cluster is a set of points

- How do we define distance between two sets of points
  - Lots of alternatives
  - Not an easy task

Distance between two clusters

- **Single-link distance** between clusters C_i and C_j is the ***minimum distance*** between any object in C_i and any object in C_j

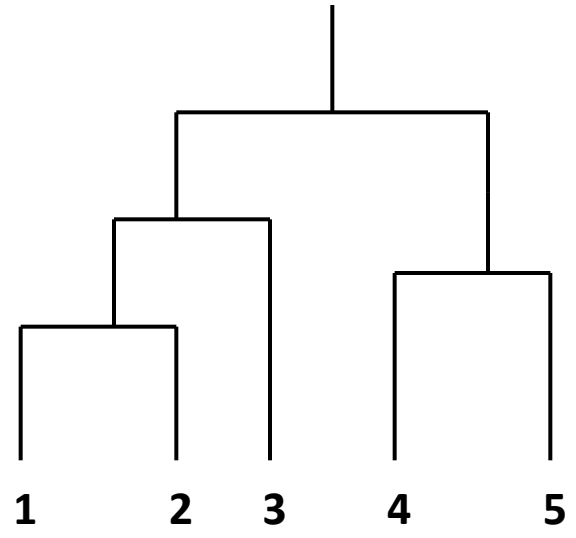
- The distance is **defined by the two most similar objects**

$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x, y) \middle| x \in C_i, y \in C_j \right\}$$

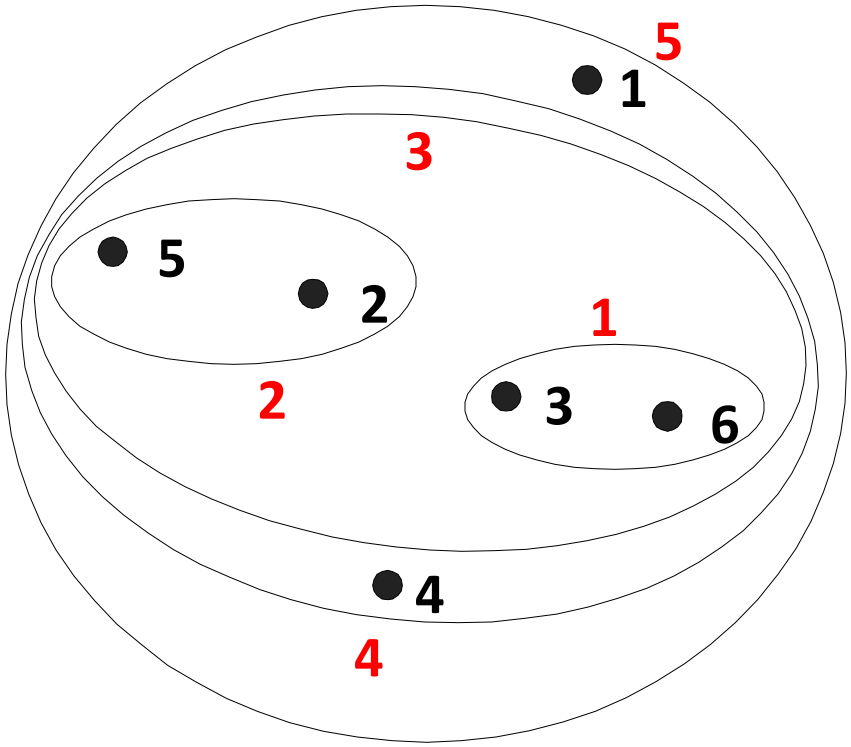
Single-link clustering: example

- Determined by one pair of points, i.e., by one link in the proximity graph.

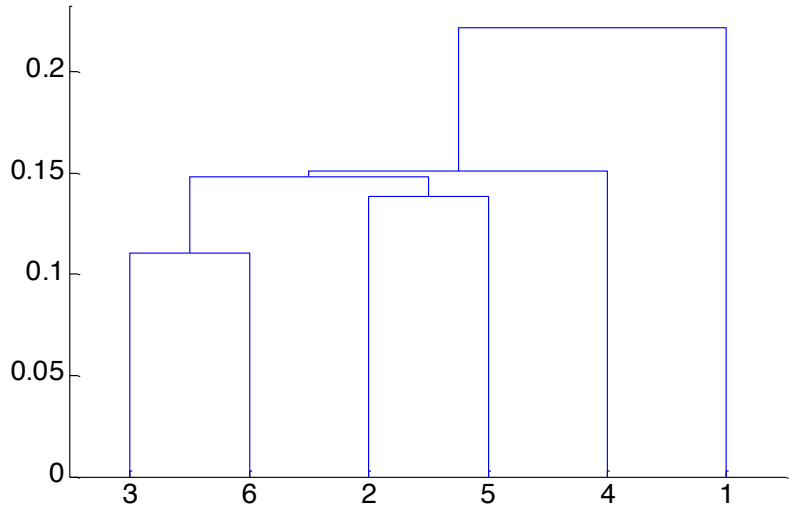
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Single-link clustering: example

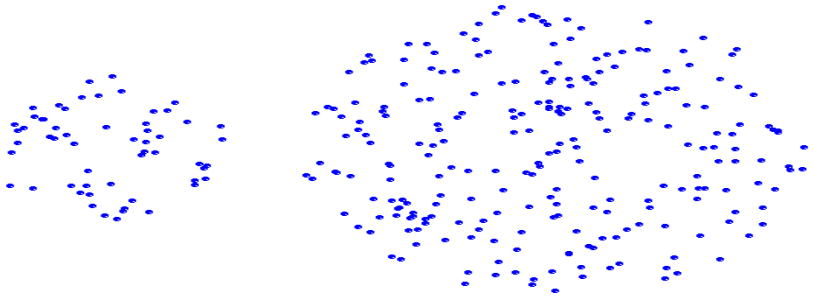


Nested Clusters

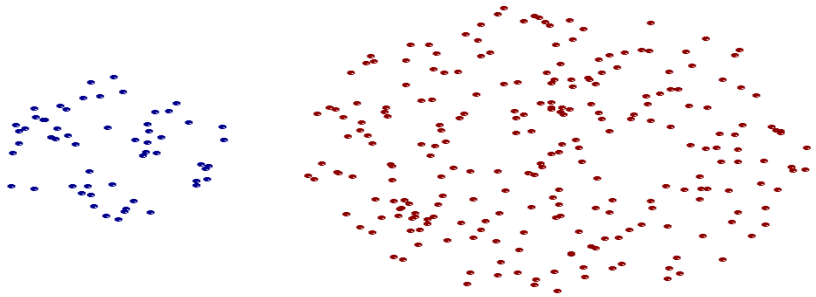


Dendrogram

Strengths of single-link clustering



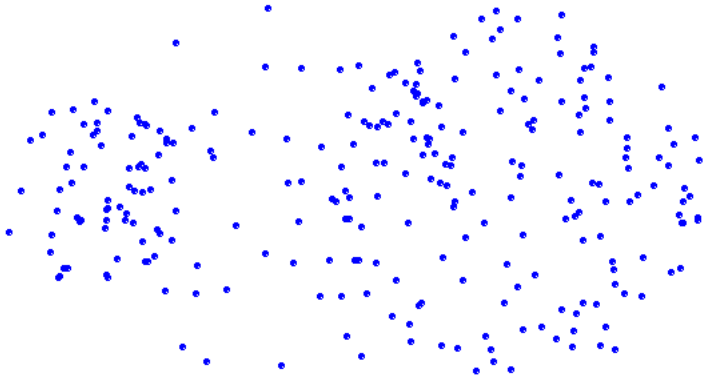
Original Points



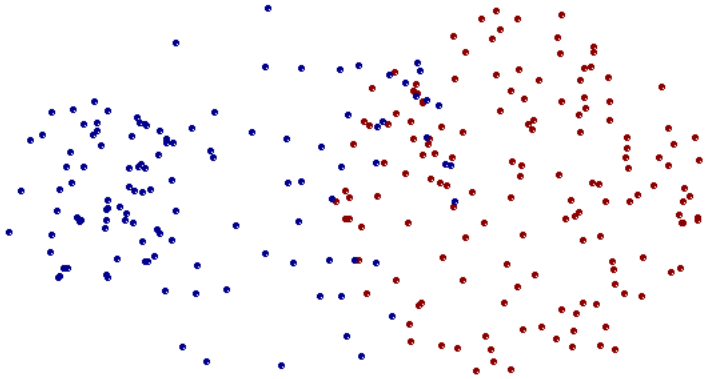
Two Clusters

- **Can handle non-elliptical shapes**

Limitations of single-link clustering



Original Points



Two Clusters

- **Sensitive to noise and outliers**
- **It produces long, elongated clusters**

Distance between two clusters

- **Complete-link distance** between clusters C_i and C_j is the ***maximum distance*** between any object in C_i and any object in C_j

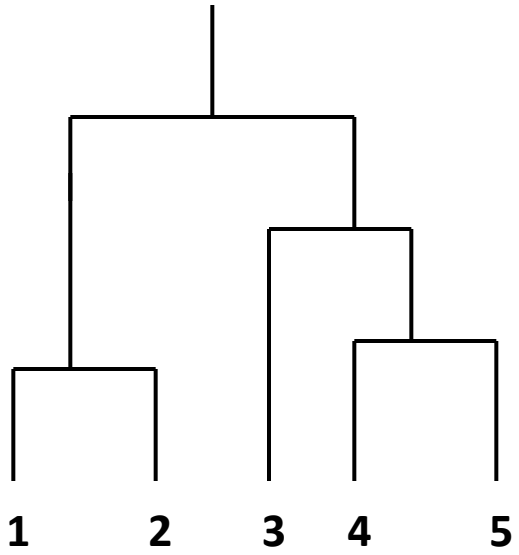
- The distance is **defined by the two most dissimilar objects**

$$D_{cl}(C_i, C_j) = \max_{x,y} \{ d(x, y) \mid x \in C_i, y \in C_j \}$$

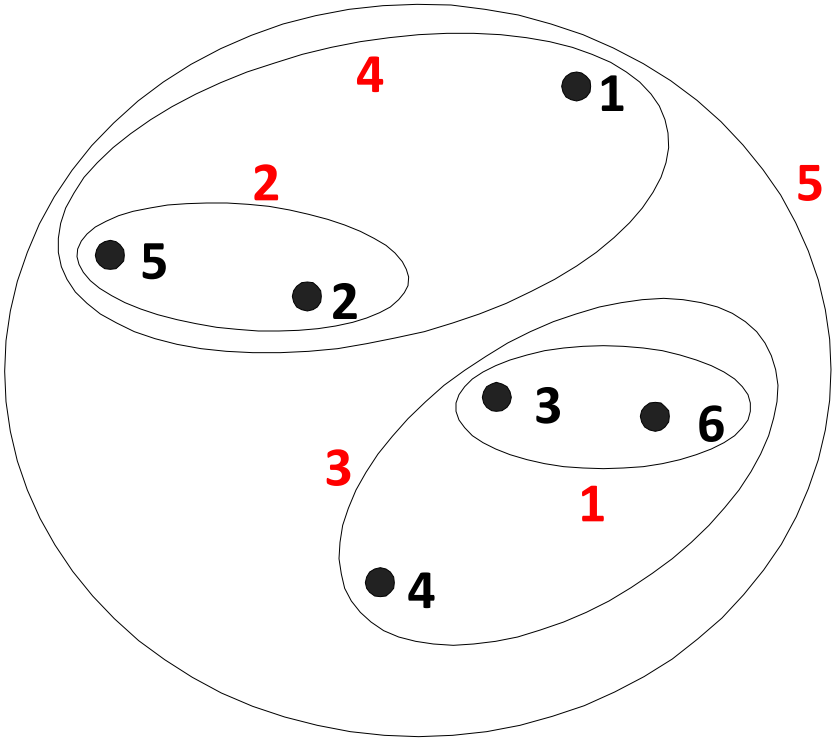
Complete-link clustering: example

- Distance between clusters is determined by the two most distant points in the different clusters

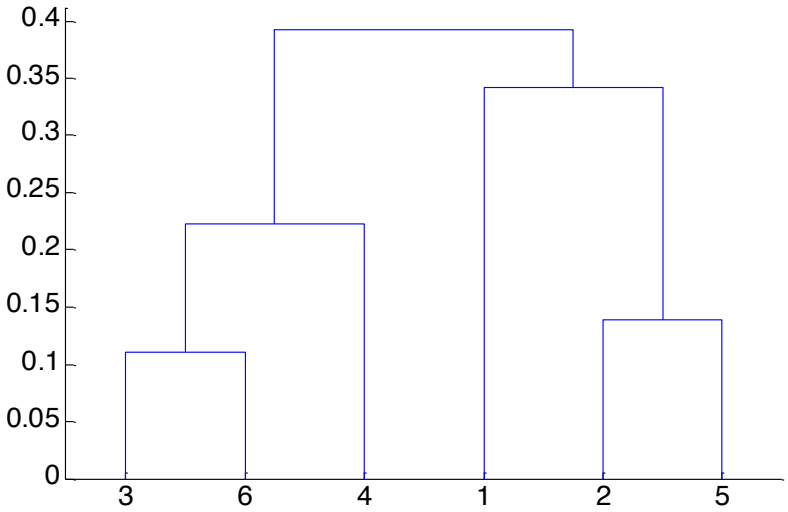
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Complete-link clustering: example

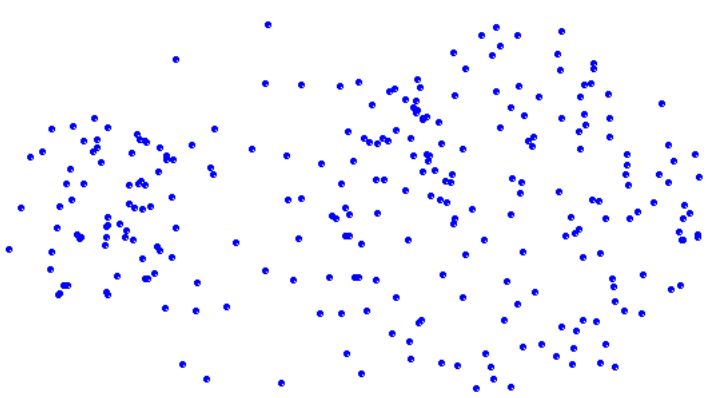


Nested Clusters

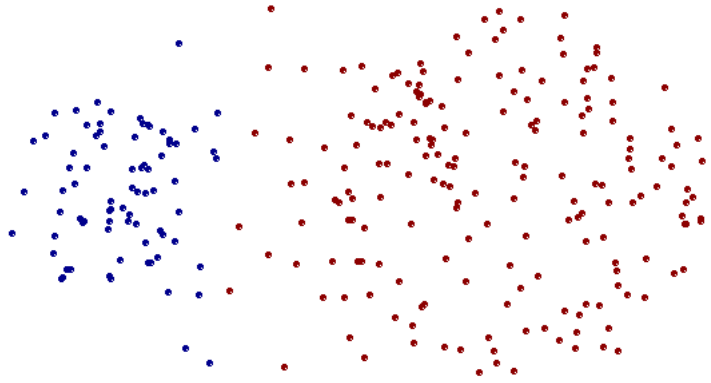


Dendrogram

Strengths of complete-link clustering



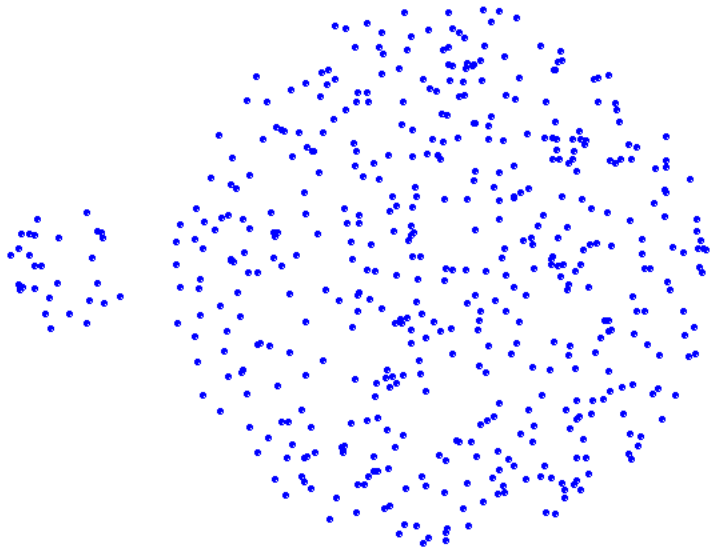
Original Points



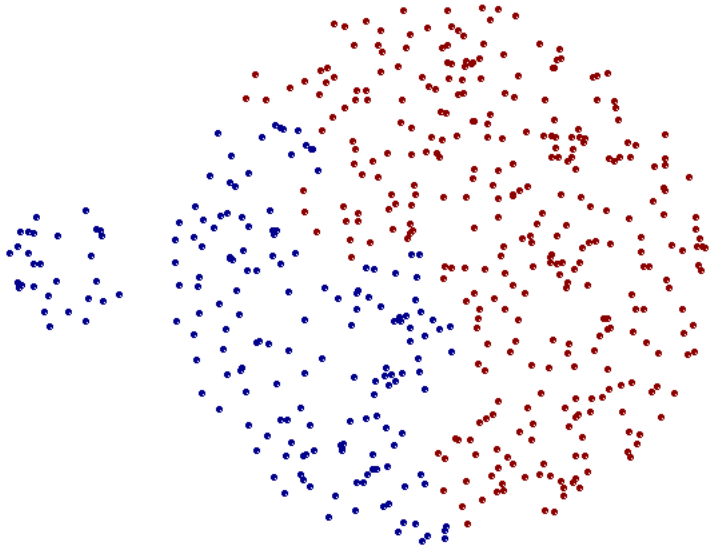
Two Clusters

- **More balanced clusters (with equal diameter)**
- **Less susceptible to noise**

Limitations of complete-link clustering



Original Points



Two Clusters

- **Tends to break large clusters**
- **All clusters tend to have the same diameter – small clusters are merged with larger ones**

Distance between two clusters

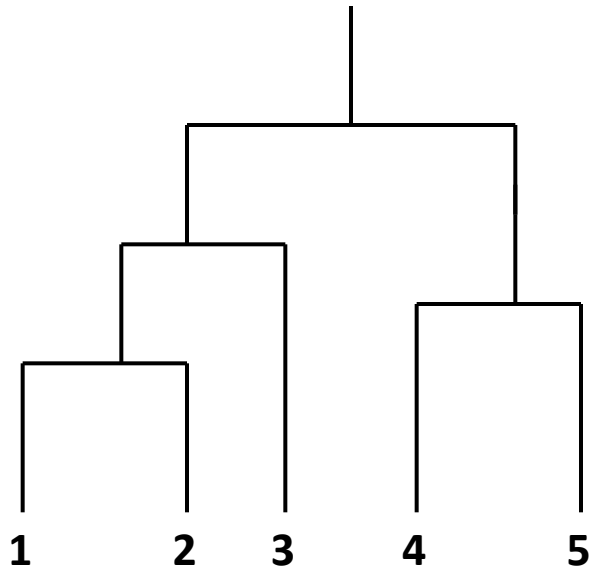
- **Group average distance** between clusters C_i and C_j is the ***average distance*** between any object in C_i and any object in C_j

$$D_{avg}(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

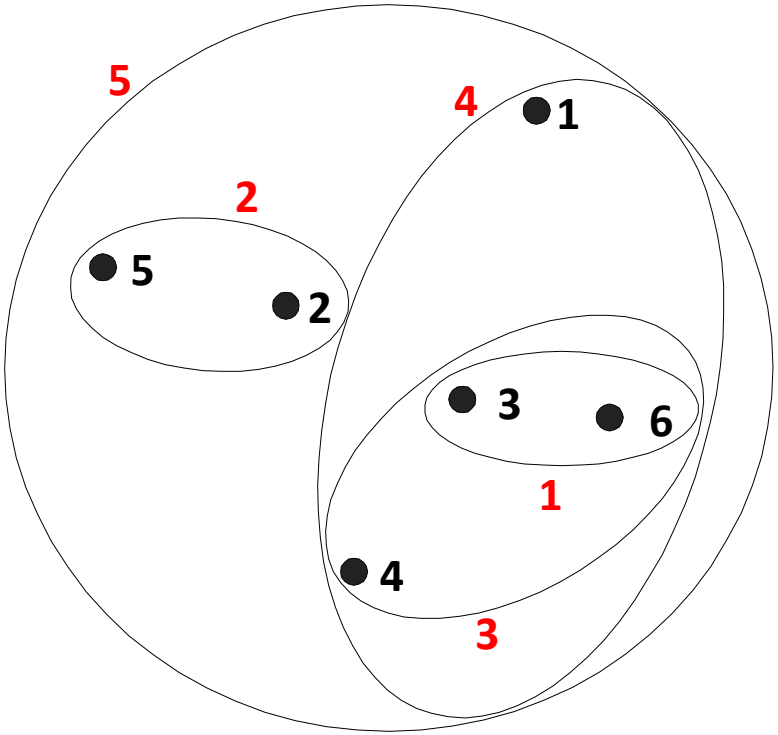
Average-link clustering: example

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

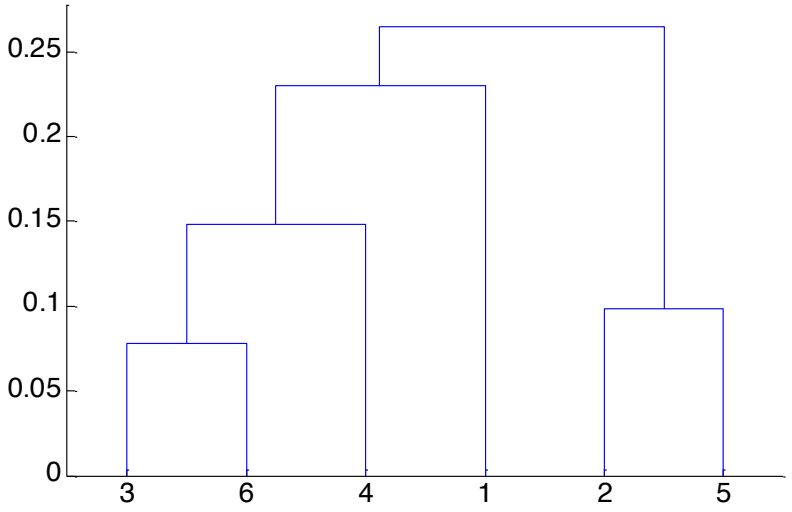
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Average-link clustering: example



Nested Clusters



Dendrogram

Average-link clustering: discussion

- Compromise between Single and Complete Link

- Strengths
  - Less susceptible to noise and outliers

- Limitations
  - Biased towards globular clusters

Distance between two clusters

- **Centroid distance** between clusters C_i and C_j is the distance between the centroid r_i of C_i and the centroid r_j of C_j

$$D_{centroids}(C_i, C_j) = d(r_i, r_j)$$

Distance between two clusters

- **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***, and the ***within cluster sum of squares resulting from merging the two clusters*** in cluster C_{ij}

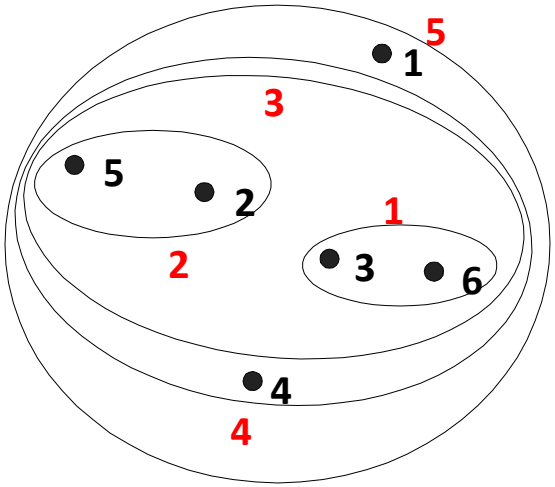
$$D_w(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

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

Ward's distance for clusters

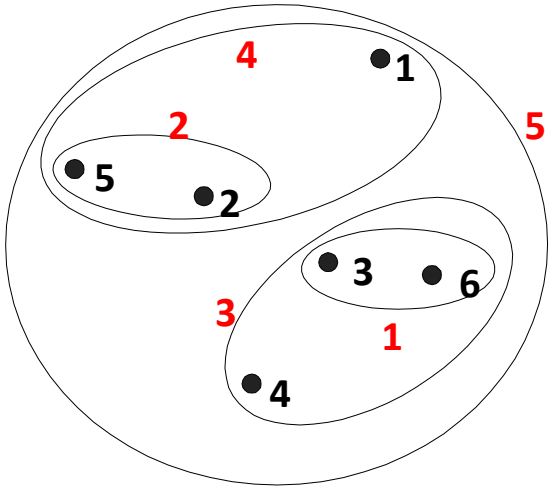
- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means
  - Can be used to initialize k-means

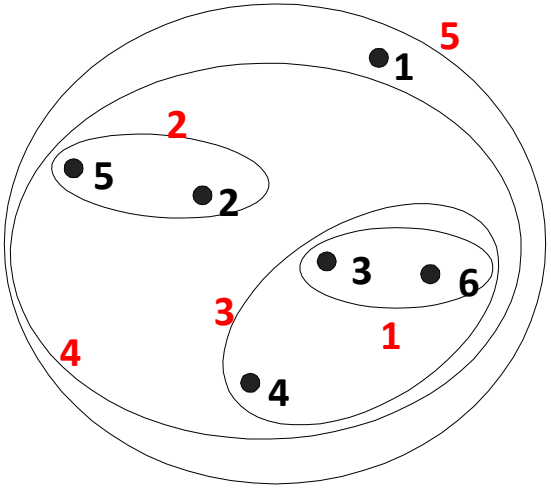
Hierarchical Clustering: Comparison



MIN

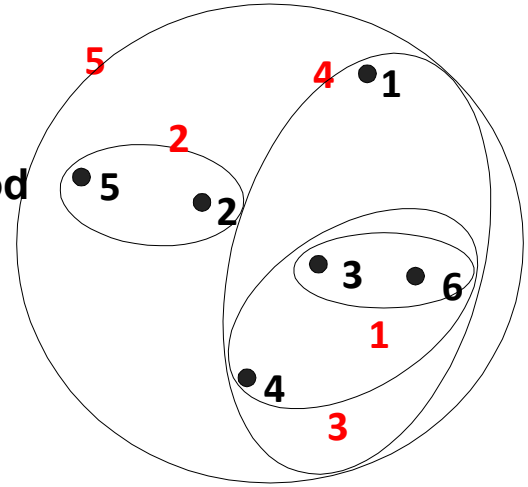
MAX





Group Average

Ward's Method



Implementing hierarchical clustering

- Naïve implementation:
  - At each step, compute pairwise distances between all pairs of clusters
  - $O(N^3)$
- Careful implementation using priority queue can reduce time to $O(N^2 \log N)$
  - Still too expensive for really big datasets that do not fit in memory

Hierarchical Clustering: Time and Space requirements

- For a dataset **X** consisting of **n** points
- **$O(n^2)$ space**; it requires storing the distance matrix
- **$O(n^3)$ time** in most of the cases
  - There are **n** steps and at each step the size **n^2** distance matrix must be updated and searched
  - Complexity can be reduced to **$O(n^2 \log(n))$** time for some approaches by using appropriate data structures

Divisive hierarchical clustering

- Start with a single cluster composed of all data points
- Split this into components
- Continue recursively
- *Monothetic* divisive methods split clusters using one variable/dimension at a time
- *Polythetic* divisive methods make splits on the basis of all variables together
- Any intercluster distance measure can be used
- Computationally intensive, less widely used than agglomerative methods

And in the Non-Euclidean Case?

- The only "locations" we can talk about are the points themselves.
  - i.e., there is no "average" of two points.

- Approach 1: *clustroid* = point "closest" to other points.
  - Treat clustroid as if it were centroid, when computing intercluster distances.

"Closest" Point?

- Possible meanings:
  1. Smallest maximum distance to the other points.
  2. Smallest average distance to other points.
  3. Smallest sum of squares of distances to other points.
     - For distance metric d clustroid c of cluster C is:

$$\min_{c} \sum_{x \in C} d(x,c)^2$$

Defining "Nearness" of Clusters

- Approach 2: intercluster distance = minimum of the distances between any two points, one from each cluster.

- Approach 3: Pick a notion of "cohesion" of clusters, e.g., maximum distance from the clustroid.
  - Merge clusters whose *union* is most cohesive.

Cohesion

- Approach 1: Use the *diameter* of the merged cluster = maximum distance between points in the cluster.
- Approach 2: Use the average distance between points in the cluster.
- Approach 3: Use a density-based approach: take the diameter or avg. distance, e.g., and divide by the number of points in the cluster.
  - Perhaps raise the number of points to a power first, e.g., square-root.

Partitioning algorithms: basic concept

- Construct a partition of a set of ***n*** objects into a set of ***k*** clusters
  - Each object belongs to **exactly one** cluster
  - The number of clusters **k** is given in advance

The k-means problem

- Given a set **X** of **n** points in a **d**-dimensional space and an integer **k**

- **Task:** choose a set of **k** points **{c_1, c_2,…,c_k}** in the **d**-dimensional space to form clusters **{C_1, C_2,…,C_k}** such that

$$Cost(C) = \sum_{i=1}^{k} \sum_{x \in C_i} L_2{}^2 (x - c_i)$$

  is minimized
- Some special cases: k = 1, k = n

Algorithmic properties of the k-means problem

- NP-hard if the dimensionality of the data is at least 2 (**d>=2**)
- Finding the best solution in polynomial time is infeasible
- For **d=1** the problem is solvable in polynomial time (how?)
- A simple iterative algorithm works quite well in practice

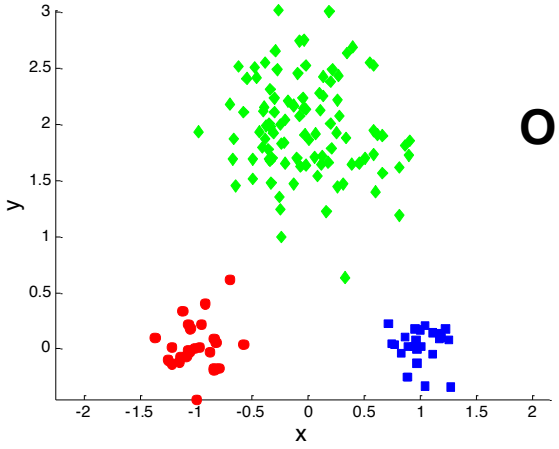
The k-means algorithm

- One way of solving the **k**-means problem
- Randomly pick **k** cluster centers $\{c_1,\ldots,c_k\}$
- For each **i**, set the cluster C_i to be the set of points in **X** that are closer to c_i than they are to c_j for all $i \neq j$
- For each **i** let c_i be the center of cluster C_i (mean of the vectors in C_i)
- Repeat until convergence

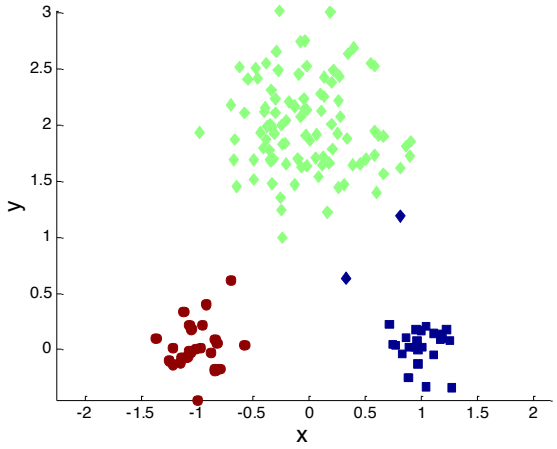
Properties of the k-means algorithm

- Finds a local optimum
- Converges often quickly (but not always)
- The choice of initial points can have large influence in the result

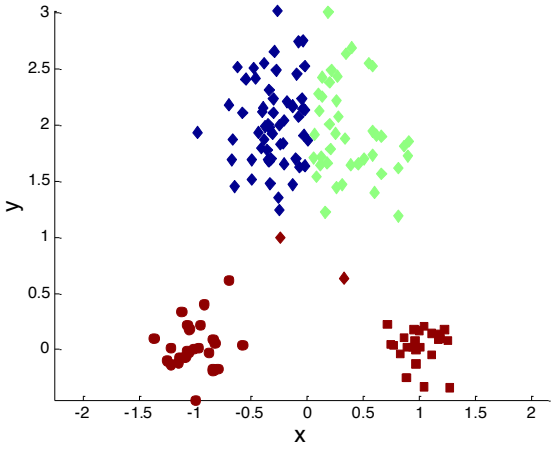
Two different K-means Clusterings



Original Points



Optimal Clustering



Sub-optimal Clustering

k – Means Algorithm(s)

- Assumes Euclidean space/distance.
- Start by picking *k*, the number of clusters.
- Initialize clusters by picking one point per cluster.
  - Example: pick one point at random, then *k*-1 other points, each as far away as possible from the previous points.

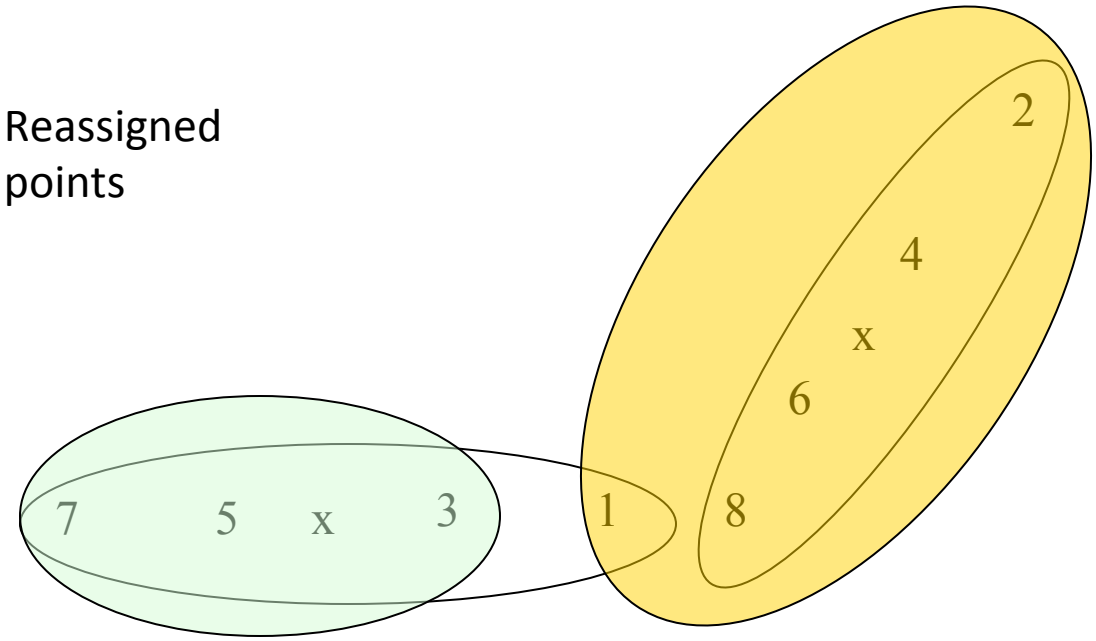
Populating Clusters

1. For each point, place it in the cluster whose current centroid it is nearest.

2. After all points are assigned, fix the centroids of the *k* clusters.

3. Optional: reassign all points to their closest centroid.
   - Sometimes moves points between clusters.

Example: Assigning Clusters

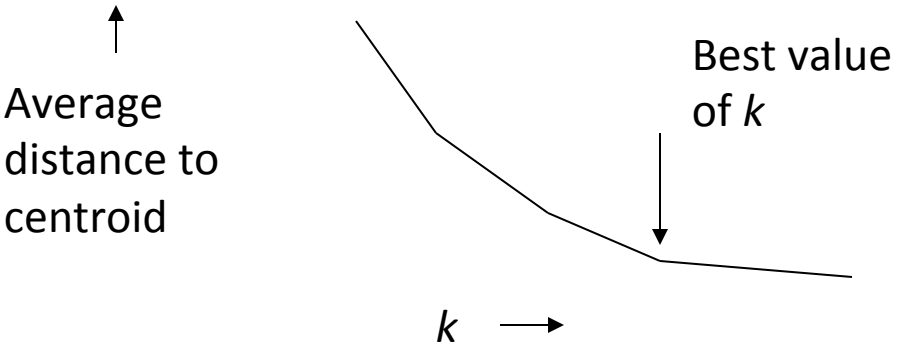


Clusters after first round

Getting *k* Right

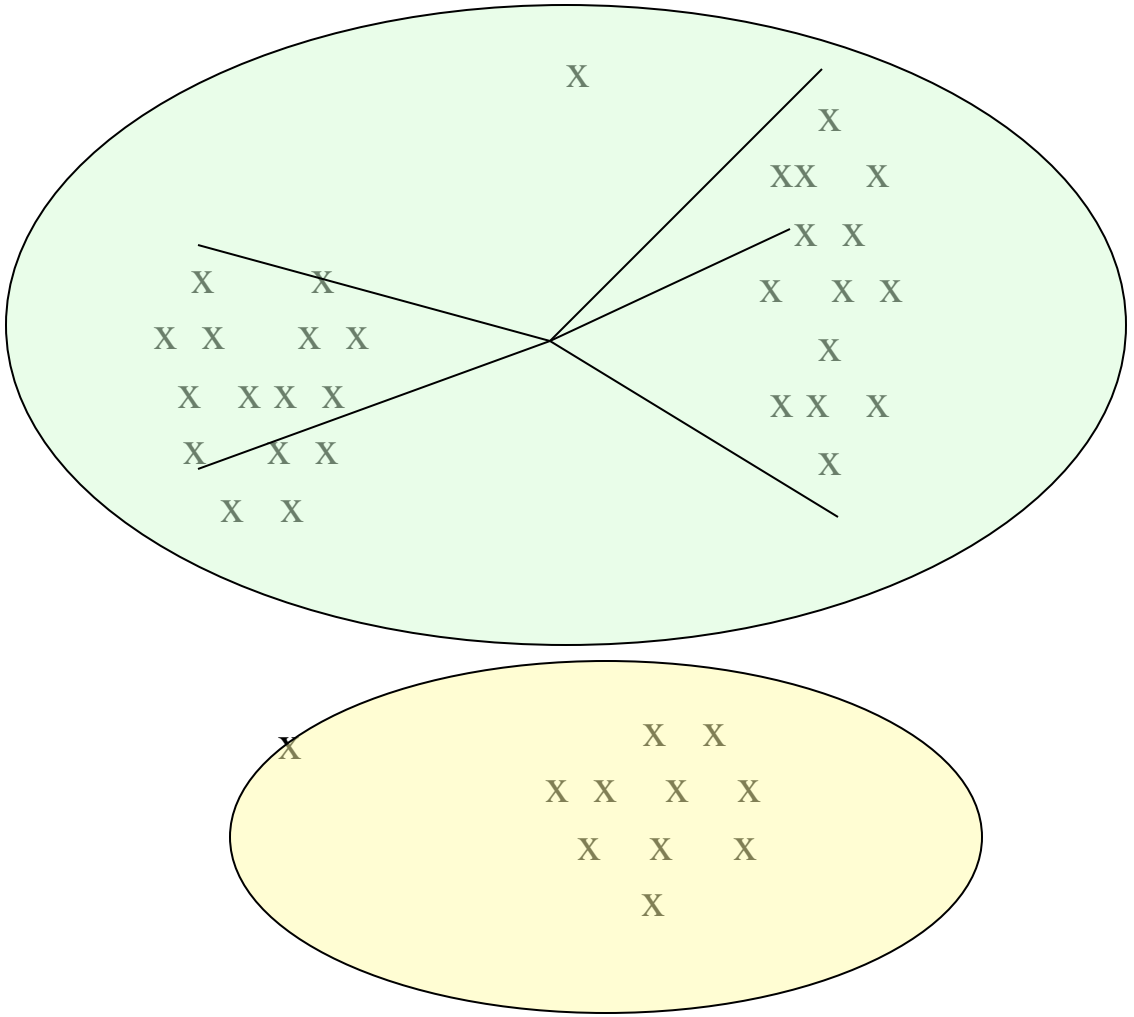
– Try different *k*, looking at the change in the average distance to centroid, as *k* increases.

- Average falls rapidly until right *k*, then changes little.



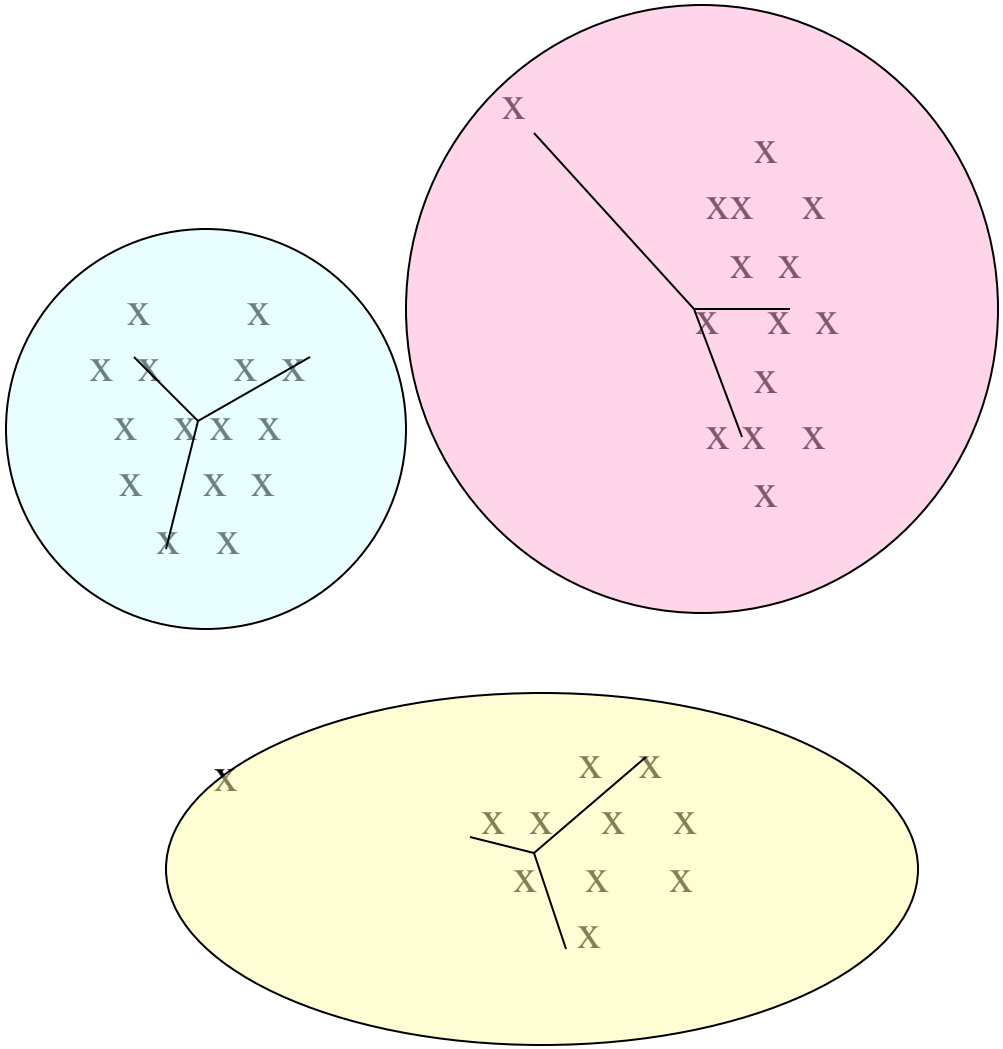
Example: Picking *k*

Too few;
many long
distances
to centroid.



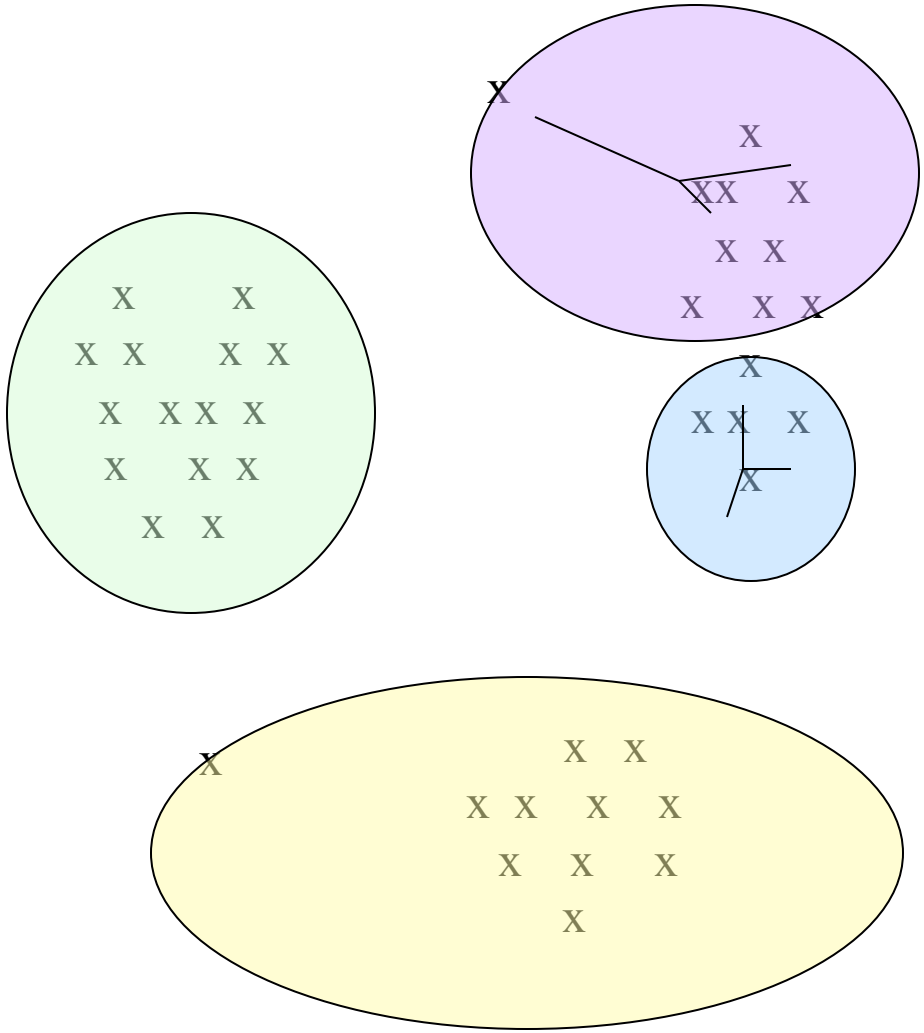
Example: Picking *k*

Just right; distances rather short.



Example: Picking *k*

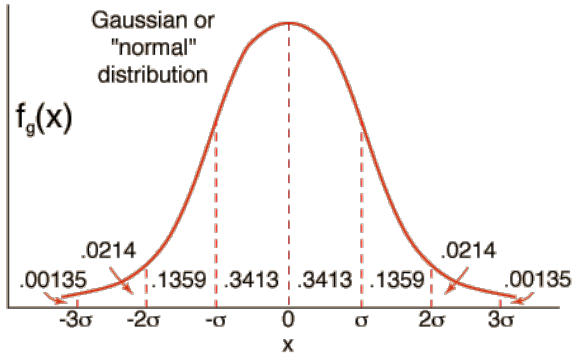
Too many;
little improvement
in average
distance.



The BFR Algorithm

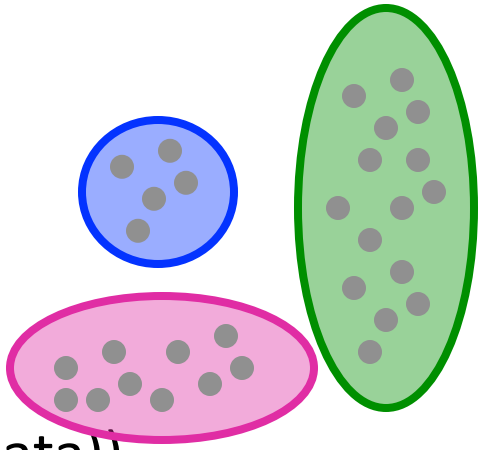
Extension of *k*-means to large data

BFR Algorithm



- **BFR** [Bradley-Fayyad-Reina] is a variant of *k*-means designed to handle **very large** (disk-resident) data sets

- **Assumes** that clusters are normally distributed around a centroid in a Euclidean space
  - Standard deviations in different dimensions may vary
    - Clusters are axis-aligned ellipses



- **Efficient way to summarize clusters** (want memory required O(clusters) and not O(data))

BFR Algorithm

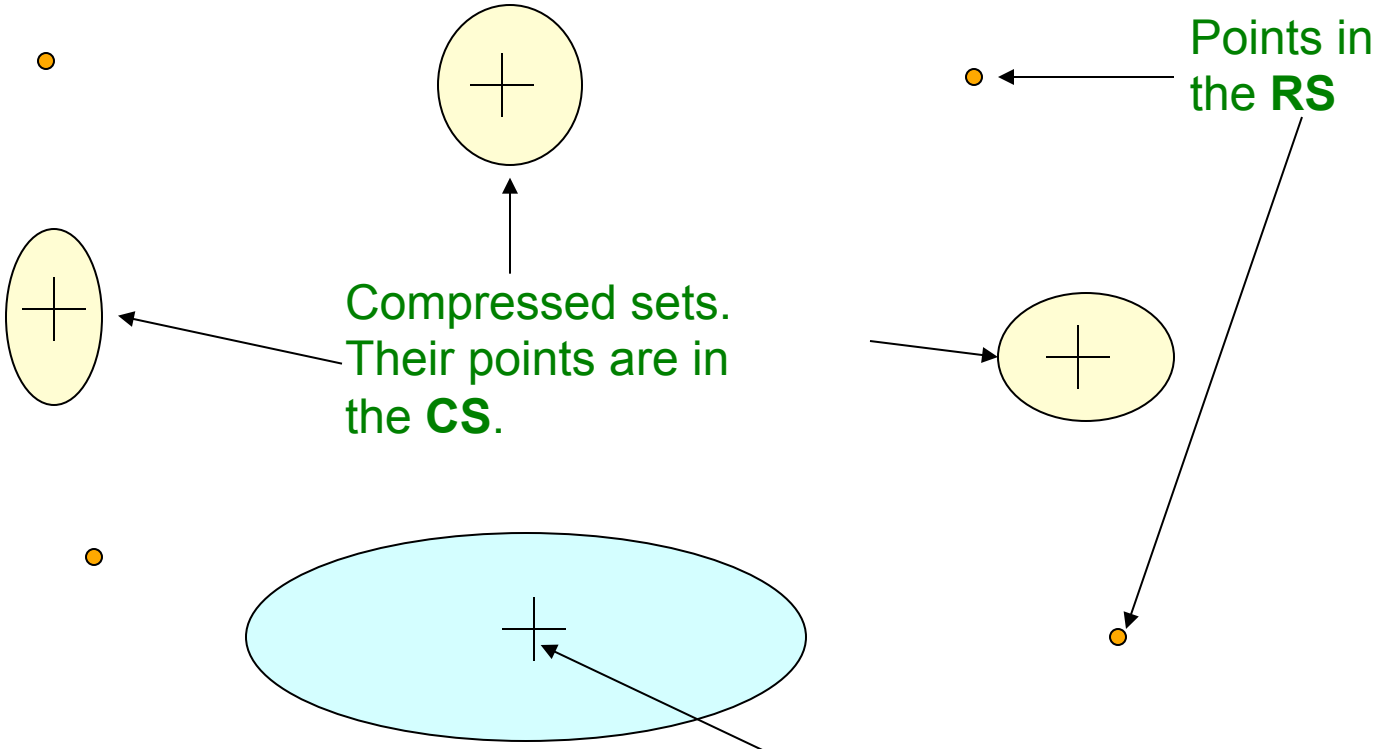
- Points are read from disk one main-memory-full at a time
- **Most points from previous memory loads are summarized by simple statistics**
- To begin, from the initial load we select the initial ***k*** centroids by some sensible approach:
  - Take ***k*** random points
  - Take a small random sample and cluster optimally
  - Take a sample; pick a random point, and then ***k–1*** more points, each as far from the previously selected points as possible

Three Classes of Points

3 sets of points which we keep track of:

- **Discard set (DS):**
  - Points close enough to a centroid to be summarized
- **Compression set (CS):**
  - Groups of points that are close together but not close to any existing centroid
  - These points are summarized, but not assigned to a cluster
- **Retained set (RS):**
  - Isolated points waiting to be assigned to a compression set

BFR: "Galaxies" Picture

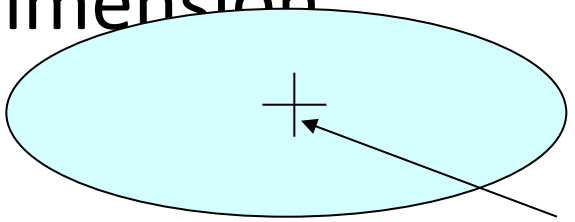


Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

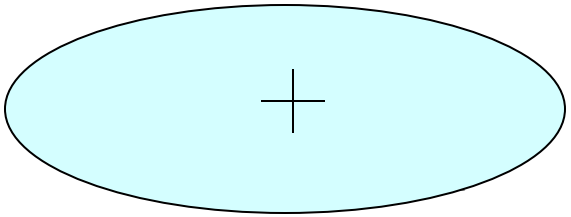
For each cluster, the discard set (DS) is summarized by:

- The number of points, ***N***
- The vector ***SUM***, whose i^{th} component is the sum of the coordinates of the points in the i^{th} dimension
- The vector ***SUMSQ***: i^{th} component = sum of squares of coordinates in i^{th} dimension



Summarizing Points: Comments

- **$2d + 1$** values represent any size cluster
  - ***d*** = number of dimensions
- Average in **each dimension** (**the centroid**) can be calculated as **SUM_i / N**
  - **SUM_i** = i^{th} component of SUM
- Variance of a cluster's discard set in dimension *i* is: **$(SUMSQ_i / N) – (SUM_i / N)^2$**
  - And standard deviation is the square root of that
- **Next step: Actual clustering**



Note: Dropping the "axis-aligned" clusters assumption would require storing full covariance matrix to summarize the cluster. So, instead of **SUMSQ** being a ***d***-dim vector, it would be a ***d* x *d*** matrix, which is too big!

The "Memory-Load" of Points

Processing the "Memory-Load" of points (1):

- **1)** Find those points that are "**sufficiently close**" to a cluster centroid and add those points to that cluster and the **DS**
  - These points are so close to the centroid that they can be summarized and then discarded
- **2)** Use any main-memory clustering algorithm to cluster the remaining points and the old **RS**
  - Clusters go to the **CS**; outlying points to the **RS**

Discard set (DS): Close enough to a centroid to be summarized.
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

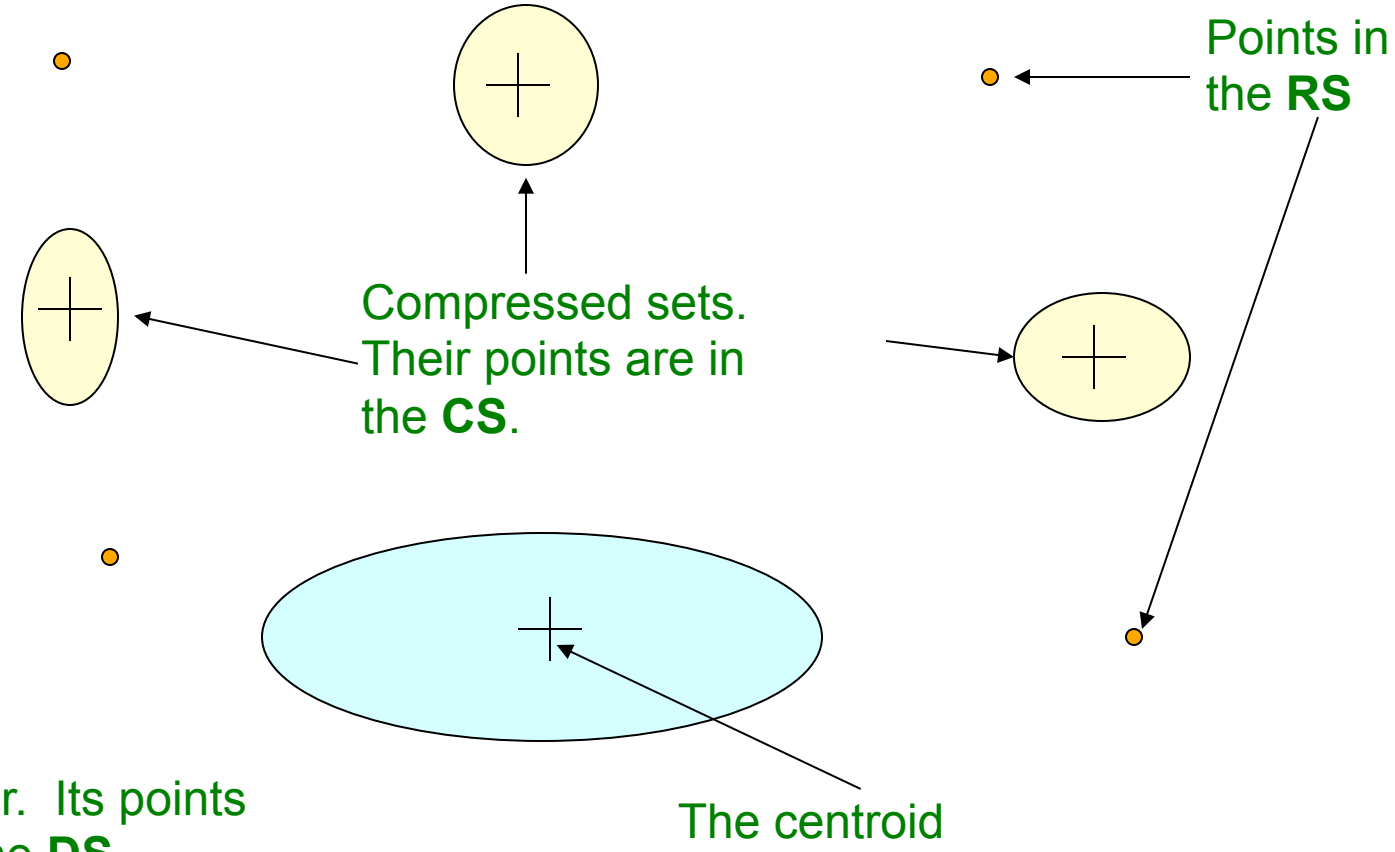
The "Memory-Load" of Points

Processing the "Memory-Load" of points (2):

- **3) DS set:** Adjust statistics of the clusters to account for the new points
  - Add ***N***s, ***SUM***s, ***SUMSQ***s
- **4)** Consider merging compressed sets in the **CS**
- **5)** If this is the last round, merge all compressed sets in the **CS** and all **RS** points into their nearest cluster

Discard set (DS): Close enough to a centroid to be summarized.
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

BFR: "Galaxies" Picture



Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

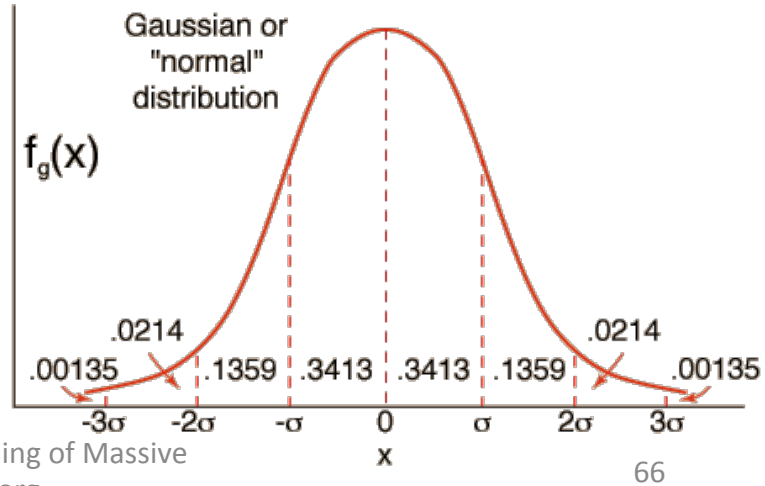
A Few Details…

- **Q1) How do we decide if a point is "close enough" to a cluster that we will add the point to that cluster?**
- **Q2) How do we decide whether two compressed sets (CS) deserve to be combined into one?**

How Close is Close Enough?

- **Q1) We need a way to decide whether to put a new point into a cluster (and discard)**

- **BFR suggests two ways:**
  - The **Mahalanobis distance** is less than a threshold
  - **High likelihood of the point belonging to currently nearest centroid**



Mahalanobis Distance

- **Normalized Euclidean distance from centroid**

- For point $(x_1, \ldots, x_d)$ and centroid $(c_1, \ldots, c_d)$
  1. Normalize in each dimension: $y_i = (x_i - c_i) / \sigma_i$
  2. Take sum of the squares of the y_i
  3. Take the square root

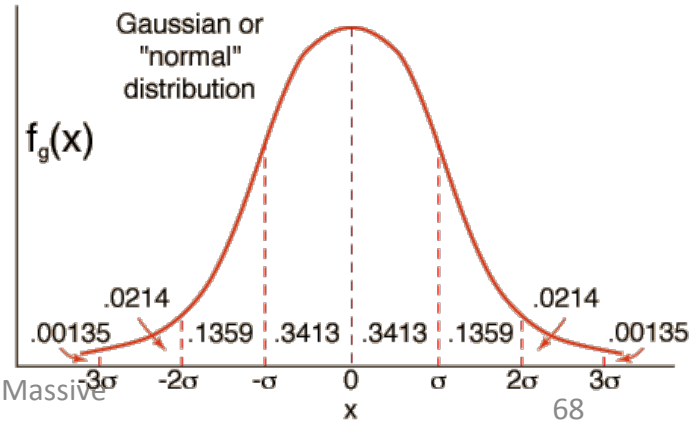
$$d(x,c)=\sqrt{\sum_{i=1}^{d}(x_i - c_i / \sigma_i)^2}$$

σ_i … standard deviation of points in the cluster in the i^{th} dimension

Mahalanobis Distance

- If clusters are normally distributed in d dimensions, then after transformation, one standard deviation = $\sqrt{d}$
  - i.e., 68% of the points of the cluster will have a Mahalanobis distance $< \sqrt{d}$

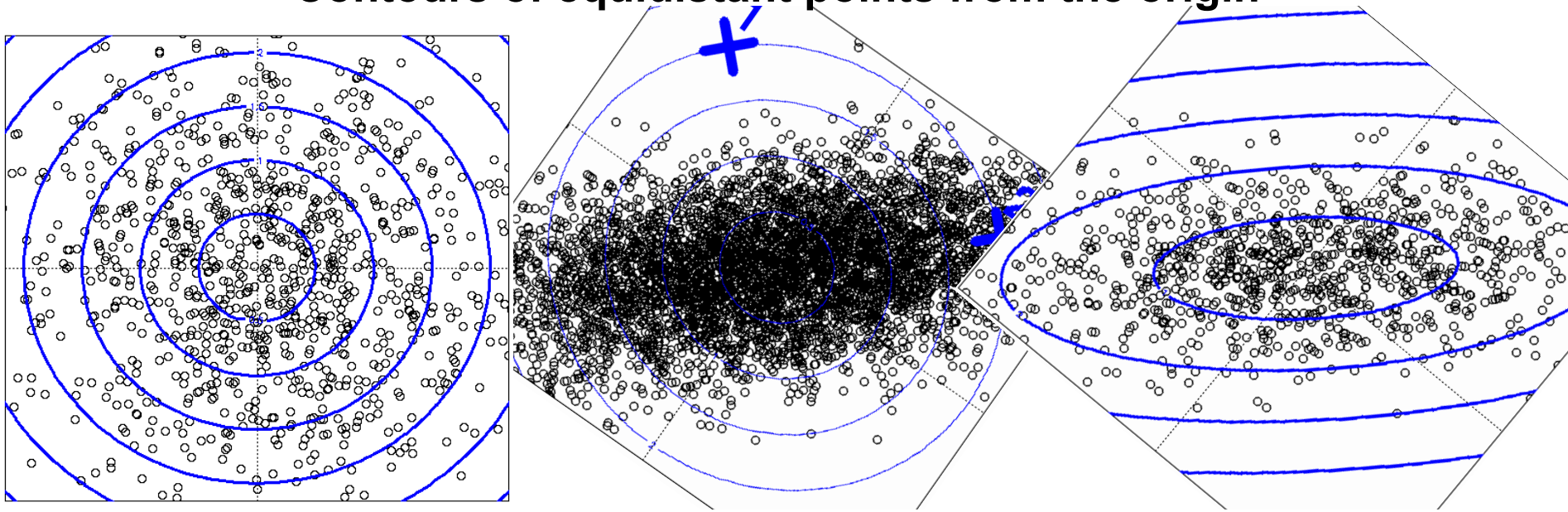
- Accept a point for a cluster if its M.D. is **<** some threshold, e.g. **2** standard deviations



Picture: Equal M.D. Regions

- **Euclidean vs. Mahalanobis distance**

Contours of equidistant points from the origin



Uniformly distributed points, Euclidean distance

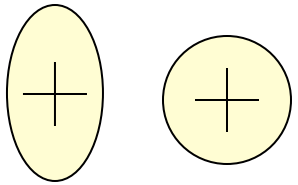
Normally distributed points, Euclidean distance

Normally distributed points, Mahalanobis distance

Should 2 CS clusters be combined?

Q2) Should 2 CS subclusters be combined?

- Compute the variance of the combined subcluster
  - ***N***, ***SUM***, and ***SUMSQ*** allow us to make that calculation quickly
- Combine if the combined variance is below some threshold



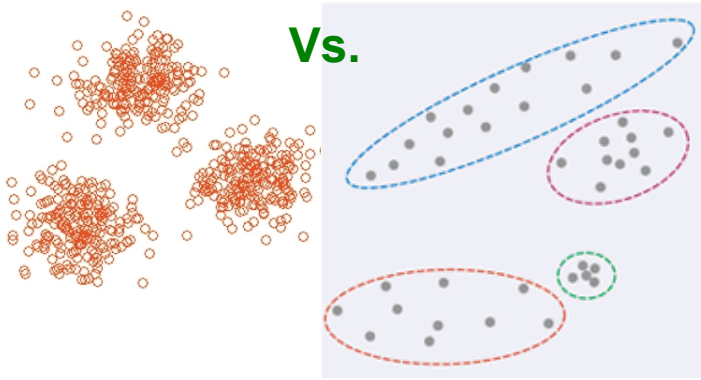
- **Many alternatives:** Treat dimensions differently, consider density

The CURE Algorithm

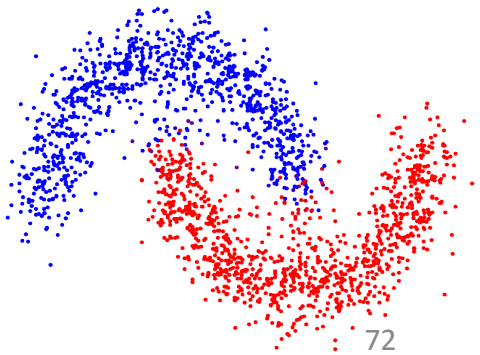
Extension of *k*-means to clusters of arbitrary shapes

The CURE Algorithm

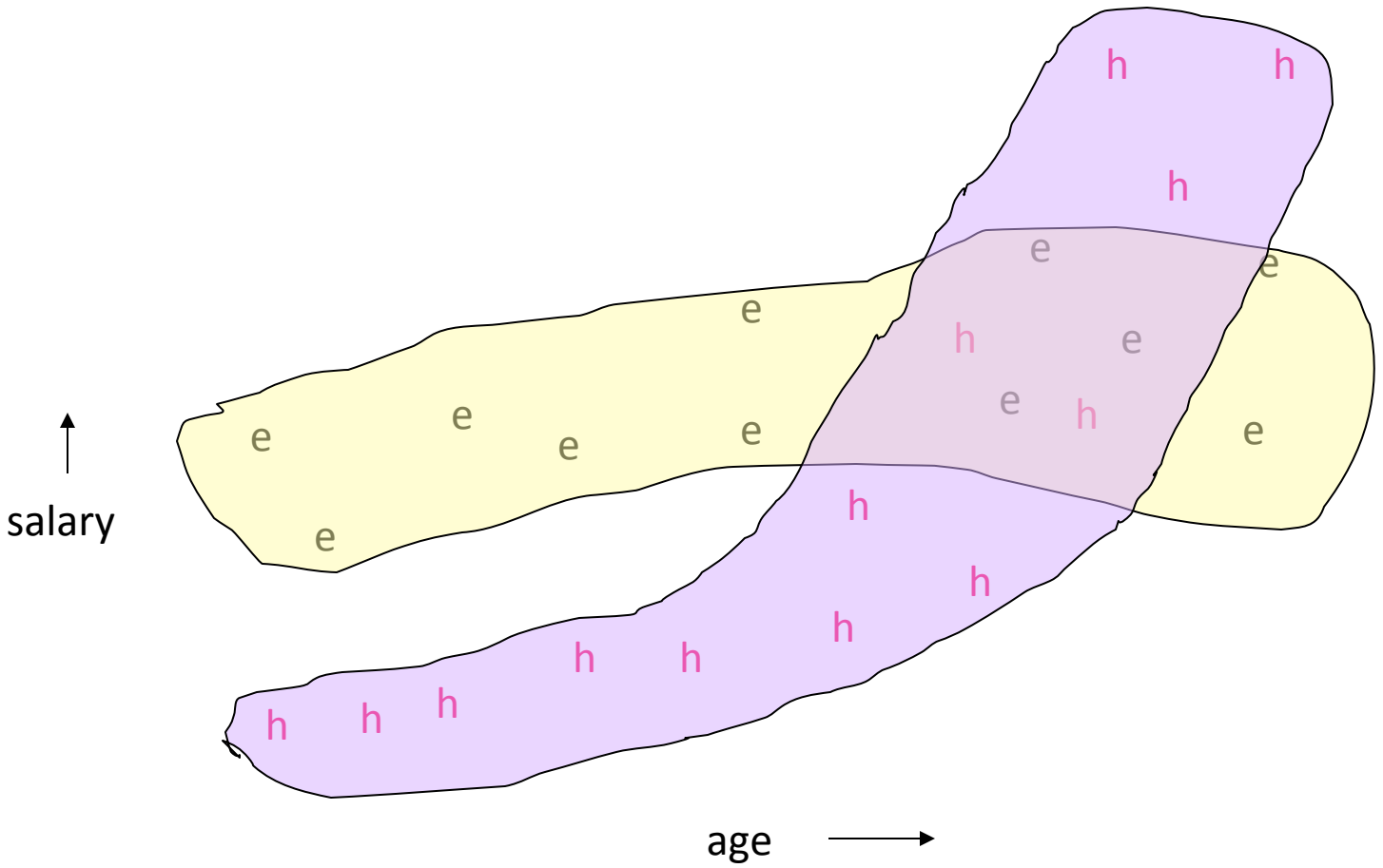
- **Problem with BFR/*k*-means:**
  - Assumes clusters are normally distributed in each dimension
  - And axes are fixed – ellipses at an angle are ***not OK***



- **CURE (Clustering Using REpresentatives):**
  - Assumes a Euclidean distance
  - Allows clusters to assume any shape
  - **Uses a collection of representative points to represent clusters**



Example: Stanford Salaries

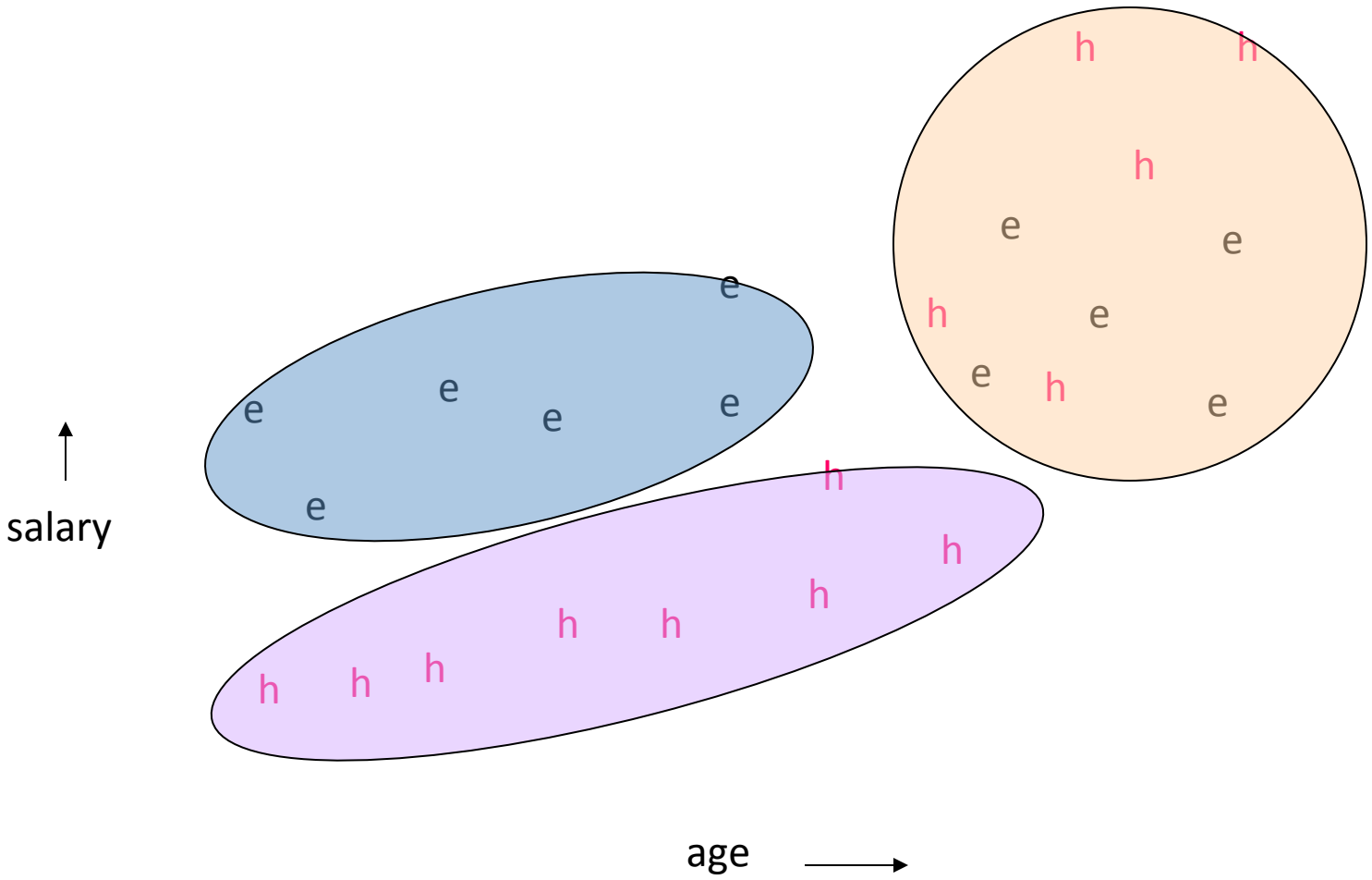


Starting CURE

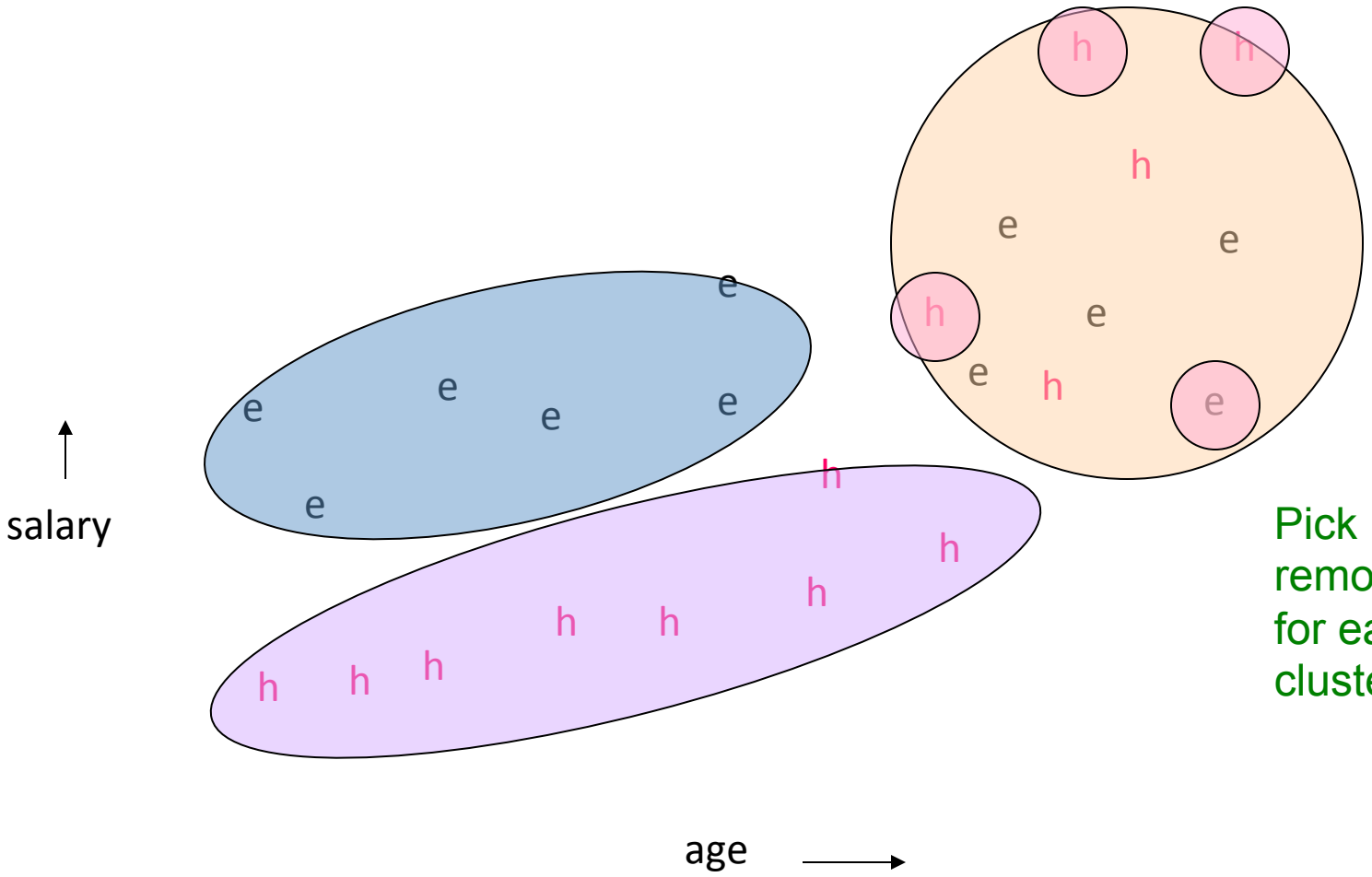
2 Pass algorithm. Pass 1:

- **0) Pick a random sample of points that fit in main memory**
- **1) Initial clusters:**
  - Cluster these points hierarchically – group nearest points/clusters
- **2) Pick representative points:**
  - For each cluster, pick a sample of points, as dispersed as possible
  - From the sample, pick representatives by moving them (say) 20% toward the centroid of the cluster

Example: Initial Clusters

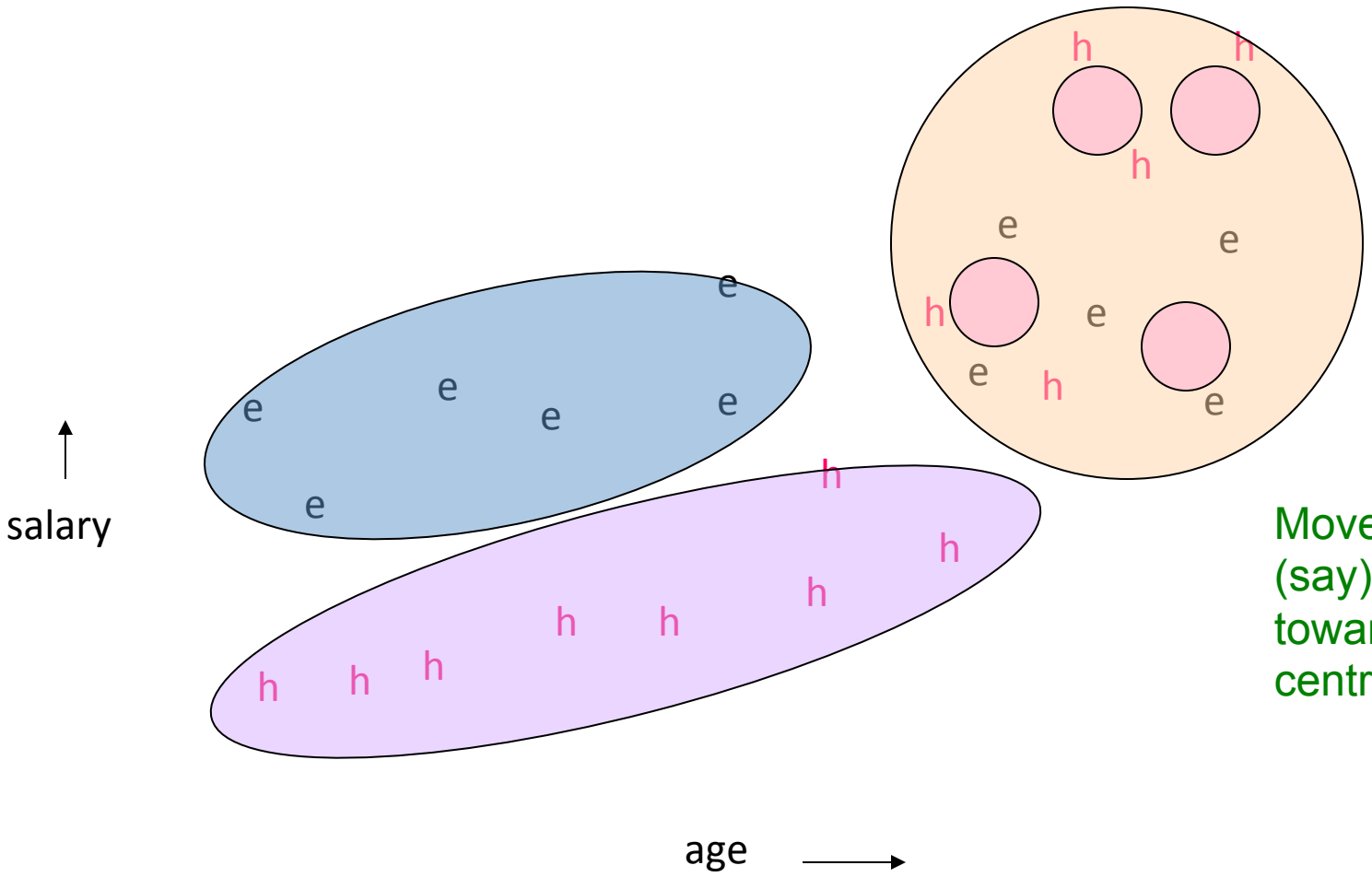


Example: Pick Dispersed Points



Pick (say) 4 remote points for each cluster.

Example: Pick Dispersed Points

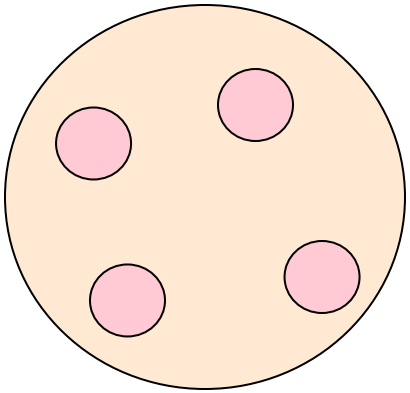


Move points (say) 20% toward the centroid.

Finishing CURE

Pass 2:

- Now, rescan the whole dataset and visit each point ***p*** in the data set



p

- **Place it in the "closest cluster"**
  - Normal definition of "closest": Find the closest representative to ***p*** and assign it to representative's cluster

Summary

- **Clustering:** Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***
- **Algorithms:**
  - Agglomerative **hierarchical clustering**:
    - Centroid and clustroid
  - ***k*-means:**
    - Initialization, picking *k*
  - **BFR**
  - **CURE**

Model-based clustering

- Assume data generated from **k** probability distributions
- ***Goal:*** find the distribution parameters
- ***Algorithm:*** Expectation Maximization (EM)
- ***Output:*** Distribution parameters and a **soft** assignment of points to clusters

Model-based clustering

- Assume **k** probability distributions with parameters: $(\theta_1,\ldots,\theta_k)$
- Given data **X**, compute $(\theta_1,\ldots,\theta_k)$ such that $Pr(X|\theta_1,\ldots,\theta_k)$ [likelihood] or $\ln(Pr(X|\theta_1,\ldots,\theta_k))$ [loglikelihood] is maximized.
- Every point $x \epsilon X$ need not be generated by a single distribution but it can be generated by multiple distributions with some probability [soft clustering]

Expectation-maximization algorithm

- Iterative procedure to compute the ***Maximum Likelihood (ML)*** estimate – even in the presence of missing or hidden data
- **EM** consists of two steps:
  - **Expectation step:** the (missing) data are estimated given the observed data and current estimates of model parameters
  - **Maximization step:** The likelihood function is maximized under the assumption that the (missing) data are known

EM Algorithm

- Initialize **k** distribution parameters **(θ_1,…, θ_k)**; Each distribution parameter corresponds to a cluster center
- Iterate between two steps
  - **E**xpectation step: (probabilistically) assign points to clusters
  - **M**aximation step: estimate model parameters that maximize the likelihood for the given assignment of points

EM Algorithm

- Initialize **k** cluster centers
- Iterate between two steps
  - **E**xpectation step: assign points to clusters

$$\Pr(x_i \in C_k) = \Pr(x_i \mid C_k) \Big/ \sum_j \Pr(x_i \mid C_j)$$

$$w_k = \frac{\sum_i \Pr(x_i \in C_k)}{n}$$

  - **M**aximation step: estimate model parameters

$$r_k = \frac{1}{n} \sum_{i=1}^{n} \frac{\Pr(x_i \in C_k)}{\sum_k \Pr(x_i \in C_j)}$$

What Is A Good Clustering?

- Internal criterion: A good clustering will produce high quality clusters in which:
  - the intra-class (that is, intra-cluster) similarity is high
  - the inter-class similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used

Cluster Validity

- For cluster analysis, the question is how to evaluate the "goodness" of the resulting clusters?
- But "clusters are in the eye of the beholder"!
- Then why do we want to evaluate them?
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two sets of clusters
  - To compare two clusters

Different Aspects of Cluster Validation

1. Determining the clustering tendency of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
   - Use only the data
4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the 'correct' number of clusters.

For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

Framework for Cluster Validity

- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more "atypical" a clustering result is, the more likely it represents valid structure in the data
  - Can compare the values of an index that result from random data or clusterings to those of a clustering result.
    - If the value of the index is unlikely, then the cluster results are valid
  - These approaches are more complicated and harder to understand.
- For comparing the results of two different sets of cluster analyses, a framework is less necessary.
  - However, there is the question of whether the difference between two index values is significant

External criteria for clustering quality

- Quality measured by its ability to discover some or all of the hidden patterns or latent classes in gold standard data
- Assesses a clustering with respect to ground truth … requires *labeled data*
- Assume documents with C gold standard classes, while our clustering algorithms produce K clusters, $\omega_1, \omega_2, \ldots, \omega_K$ with n_i members.

Evaluating clusters

- Function **H** computes the cohesiveness of a cluster (e.g., smaller values larger cohesiveness)
- Examples of cohesiveness?
- Goodness of a cluster **c** is **H(c)**
- **c** is better than **c'** if **H(c) < H(c')**

Evaluating **clusterings** using cluster cohesiveness?

- For a clustering **C** consisting of **k** clusters $c_1, \ldots, c_k$
- $H(C) = \Phi_i H(c_i)$
- What is **Φ** ?

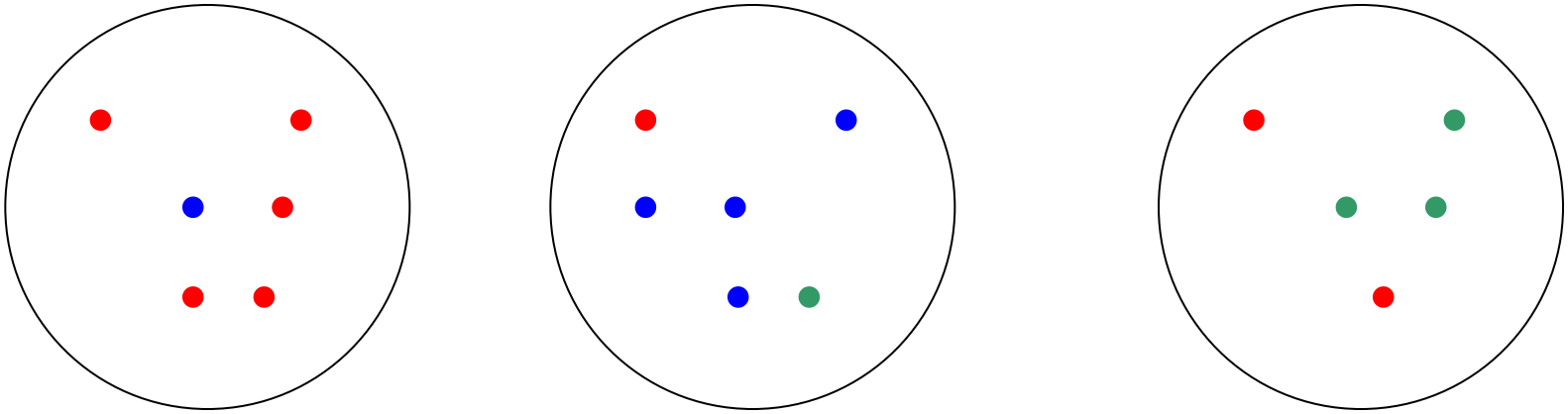
External Evaluation of Cluster Quality

- Simple measure: purity, the ratio between the dominant class in the cluster π_i and the size of cluster ω_i

$$Purity(\omega_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

- Biased because having *n* clusters maximizes purity
- Others are entropy of classes in clusters (or mutual information between classes and clusters)

Purity example



Cluster I Cluster II Cluster III

Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Cluster separation?

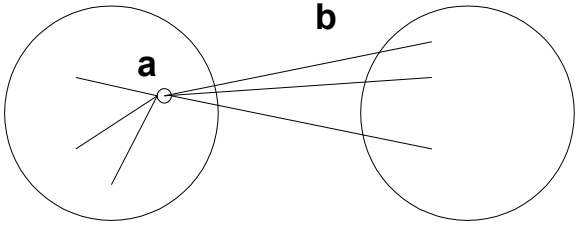
- Function **S** that measures the separation between two clusters c_i, c_j
- Ideas for **$S(c_i,c_j)$?**
- How can we measure the goodness of a clustering **$C = \{c_1,\ldots,c_k\}$** using the separation function **S**?

Silhouette Coefficient

- Silhouette Coefficient combines ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings
- For an individual point, *I*
  - ***a*** = average distance of ***i*** to the points in the same cluster
  - ***b*** = min (average distance of ***i*** to points in another cluster)
  - silhouette coefficient of **i**:

    s = 1 – a/b if **a < b**

  - Typically between 0 and 1.
  - The closer to 1 the better.



- Can calculate the Average Silhouette width for a cluster or a clustering

Final Comment on Cluster Validity

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

Algorithms for Clustering Data, Jain and Dubes

High Dimensional Data