Data Mining: Clustering
Advanced Clustering

Laura Brown

Some slides adapted from P. Smyth; A. Moore; C.F. Aliferis; S. Russell; Han, Kamber, & Pei; Tan, Steinbach, & Kumar; B. Taskar; R. Tibshirani; E. Keogh; Z. Bar-Joseph; D. Klein; and L. Kaebling

Other Types of Sets of Clusters

- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or 'border' points
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- Partial versus complete
  - In some cases, we only want to cluster some of the data
- Heterogeneous versus homogeneous
  - Cluster of widely different sizes, shapes, and densities

Slide from Tan, Steinbach, Kumar

Input Data is Important

- Type of proximity or density measure
  - This is a derived measure, but central to clustering
- Sparseness
  - Dictates type of similarity
  - Adds to efficiency
- Attribute type
  - Dictates type of similarity
- Type of Data
  - Dictates type of similarity
  - Other characteristics, e.g., autocorrelation
- Dimensionality
- Noise and Outliers
- Type of Distribution

Slide from Tan, Steinbach, Kumar

Major Clustering Approaches (I)

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSACN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

Slide from Han, Kamber, and Pei

Major Clustering Approaches (II)

- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: p-Cluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering
- Link-based clustering:
  - Objects are often linked together in various ways
  - Massive links can be used to cluster objects: SimRank, LinkClus

Slide from Han, Kamber, and Pei

Outline

- Extensions to Hierarchical Clustering
  - BIRCH, CHAMELEON
- Density-based Clustering
  - DBSCAN, OPTICS, DENCLUE
- Grid-Based Clustering
  - STING, CLIQUE
- Probabilistic Clustering
  - EM Clustering
- Clustering High-Dimensional Data
  - Subspace Clustering, Spectral Clustering
- Clustering Graphs and Network Data

Extensions to Hierarchical Clustering

- Major weaknesses of agglomerative clustering methods
  - Can never undo what was done previously
  - Do not scale well: at least $O(n^2)$
- Integrate of hierarchical and distance-based clustering
  - BIRCH: ‘96, uses Cluster Feature trees (CF-tree) and incrementally adjust the quality of sub-clusters
  - CHAMELEION: ‘99, hierarchical clustering using dynamic modeling

Slide from Han, Kamber, and Pei

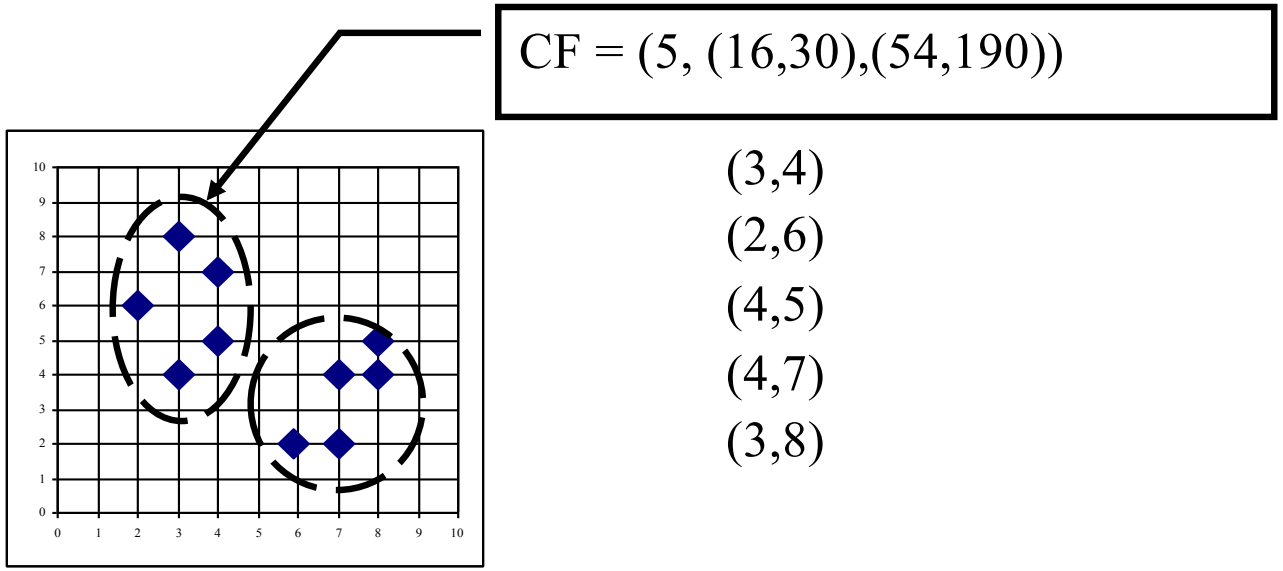
BIRCH

- BIRCH – Balanced Iterative Reducing Clustering Using Hierarchies
  - Zhang, Ramakrishnan, & Livny SIGMOD '96
- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data
  - Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- Scales linearly – finds a good clustering with a single scan and improves the quality with a few additional scans
- Weakness: handles only numeric data, and sensitive to the order of the data record

Slide from Han, Kamber, and Pei

Clustering Feature Vector

- Clustering Feature (CF): CF = (N, LS, SS)
  - N – number of data points
  - LS – linear sum of N points: $\sum_{i=1}^{N} X_i$
  - SS – square sum of N points: $\sum_{i=1}^{N} X_i^2$



Slide from Han, Kamber, and Pei

CF-Tree in BIRCH

- Clustering feature:
  - Summary of the statistics for a given subcluster: the 0-th, 1st, and 2nd moments of the subcluster from the statistical point of view
  - Registers crucial measurements for computing cluster and utilizes storage efficiently
- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering
  - A nonleaf node in a tree has descendants or “children”
  - The nonleaf nodes store sums of the CFs of their children
- A CF tree has two parameters
  - Branching factor: max # of children
  - Threshold: max diameter of sub-clusters stored at the leaf nodes

Slide from Han, Kamber, and Pei

CF properties

- A CF is a summary of statistics for the given cluster and can be used to derive many useful values:
  - The Cluster's centroid, $\overrightarrow{x_0} = \frac{\sum_{i=1}^{n} \overrightarrow{x_i}}{n} = \frac{LS}{n}$
  - Cluster's radius, *R*, average distance from cluster objects to the centroid

$$R = \sqrt{\frac{\sum_{i-1}^{n}(\overrightarrow{x_i} - \overrightarrow{x_0})^2}{n}} = \sqrt{\frac{nSS - 2LS^2 + nLS}{n^2}}$$

  - Cluster diameter, *D*, average pairwise distance within a cluster

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\overrightarrow{x_i} - \overrightarrow{x_j}\right)^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}$$

BIRCH Algorithm

- Cluster Diameter $\sqrt{\frac{1}{n(n-1)}\sum(x_i - x_j)^2}$
- For each point in the input
  - Find closest leaf entry
  - Add point to leaf entry and update CF
  - If entry diameter > max_diameter, then split leaf, and possibly parents
- Algorithm is O(n)
- Concerns
  - Sensitive to insertion order of data points
  - Since we fix the size of leaf nodes, so clusters may not be so natural
  - Clusters tend to be spherical given the radius and diameter measures

Slide from Han, Kamber, and Pei

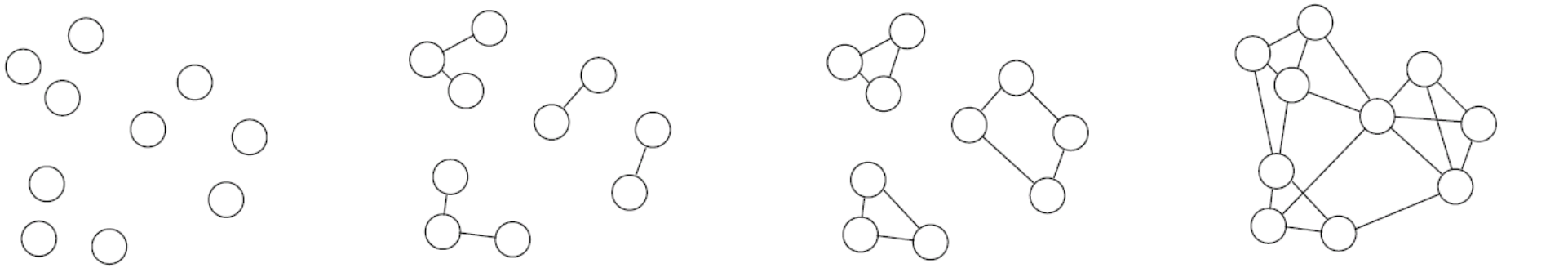
CHAMELEON

- CHAMELEON – Hierarchical Clustering using Dynamic Modeling
  - Karypis, Han, and Kumar, ‘99
- Measures the similarity based on a dynamic model
  - Two clusters are merged only if the interconnectivity and closeness (proximity) between two clusters are high relative to the internal connectivity of the cluster and closeness of items within the clusters
- Graph-based and a two-phase algorithm
  - Use a graph-partitioning algorithm: cluster objects into a large number of relatively small sub-clusters
  - Use an agglomerative hierarchical clustering algorithm: find the genuine clusters by repeatedly combining these sub-clusters

Slide from Han, Kamber, and Pei

k-nearest neighbor graph

- k-nearest graphs from original data in 2D



(a) Original Data in 2D (b) 1-nearest neighbor graph (c) 2-nearest neighbor graph (d) 3-nearest neighbor graph

Slide from Han, Kamber, and Pei

Inter-connectivity

- $EC_{\{Ci,Cj\}}$ The absolute inter-connectivity between a pair of clusters C_i and C_j is defined to be as the sum of the weight of the edges that connect vertices in C_i to vertices in C_j
- The internal inter-connectivity of a cluster C_i can be easily captured by the size of its min-cut bisector EC_{Ci} (the weighted sum of edges that partition the graph into two roughly equal parts)
- Relative inter-connectivity (RI)

$$RI(C_i, C_j) = \frac{\left|EC_{\{Ci,Cj\}}\right|}{\frac{\left|EC_{Ci}\right| + \left|EC_{Cj}\right|}{2}}$$

Slide from Han, Kamber, and Pei

Relative Closeness

- The relative closeness between a pair of clusters C_i and C_j is defined as the absolute closeness between C_i and C_j normalized with respect to the internal closeness of the the two clusters C_i and C_j

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{Ci,Cj\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{Ci}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{Cj}}}$$

- $\overline{S}_{EC_{Ci}}$ and $\overline{S}_{EC_{Cj}}$ are the average weights of the edges that belong in the min-cut bisector of clusters C_i and C_j, respectively, and $\overline{S}_{EC_{\{Ci,Cj\}}}$ is the average weight of the edges that connect vertices in C_i to vertices in C_j

Slide from Han, Kamber, and Pei

Merge Sub-Clusters

- The first scheme merges only those pairs of clusters whose relative inter-connectivity and relative closeness are both above some user specified threshold T_{RI} and T_{RC} respectively

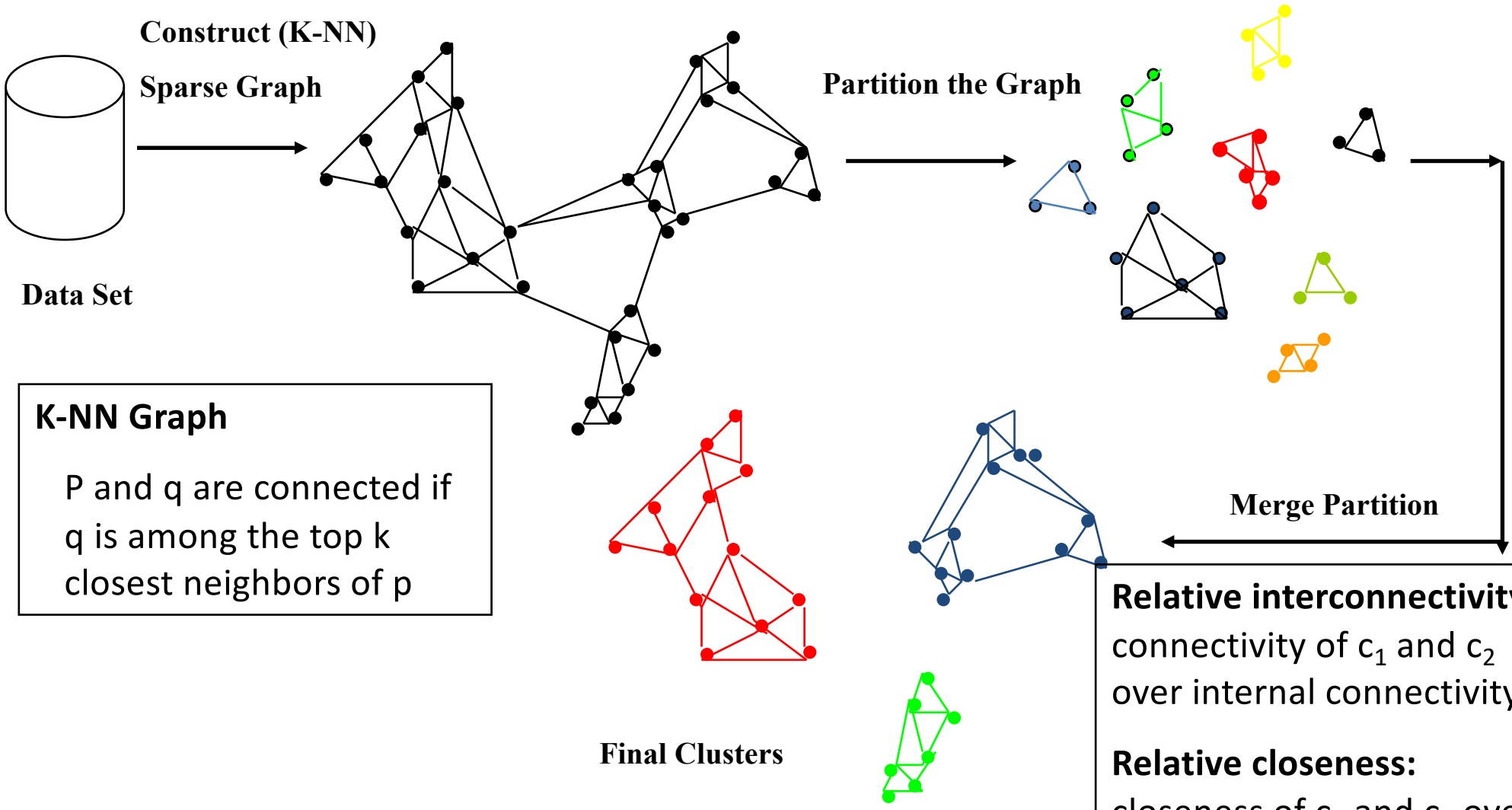
$$RI(C_i, C_j) \geq T_{RI} \quad and \quad RC(C_i, C_j) \geq T_{RC}$$

- The second scheme uses a function to combine the relative inter-connectivity and relative closeness, and then selects to merge the pair of clusters that maximizes this function

$$RI(C_i, C_j) * RC(C_i, C_j)^{\alpha}$$

Slide from Han, Kamber, and Pei

Overall Framework of CHAMELEON



K-NN Graph

P and q are connected if q is among the top k closest neighbors of p

Relative interconnectivity: connectivity of c_1 and c_2 over internal connectivity

Relative closeness: closeness of c_1 and c_2 over internal closeness

Slide from Han, Kamber, and Pei

Chameleon: Steps

- Preprocessing Step:
  Represent the Data by a Graph
  - Given a set of points, construct the k-nearest-neighbor (k-NN) graph to capture the relationship between a point and its k nearest neighbors
  - Concept of neighborhood is captured dynamically (even if region is sparse)

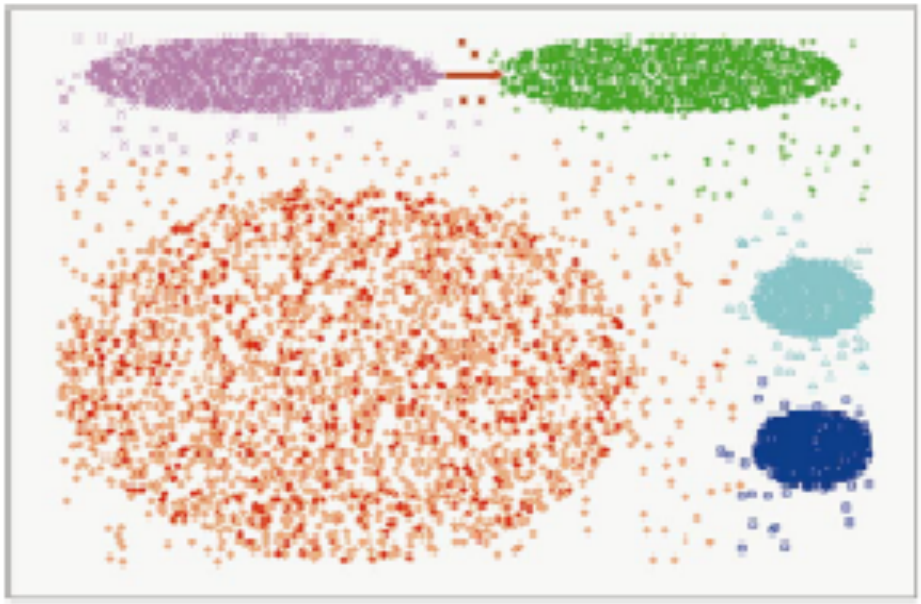
- Phase 1: Use a multilevel graph partitioning algorithm on the graph to find a large number of clusters of well-connected vertices
  - Each cluster should contain mostly points from one "true" cluster, i.e., is a sub-cluster of a "real" cluster

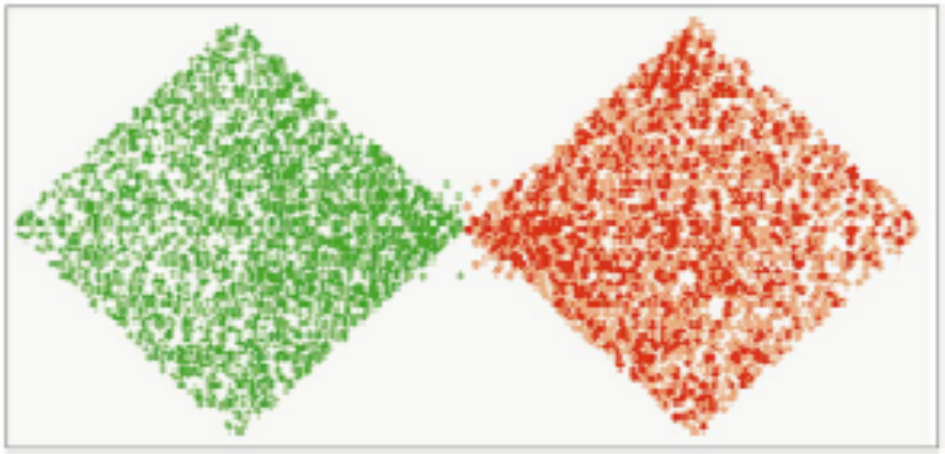
Chameleon: Steps …

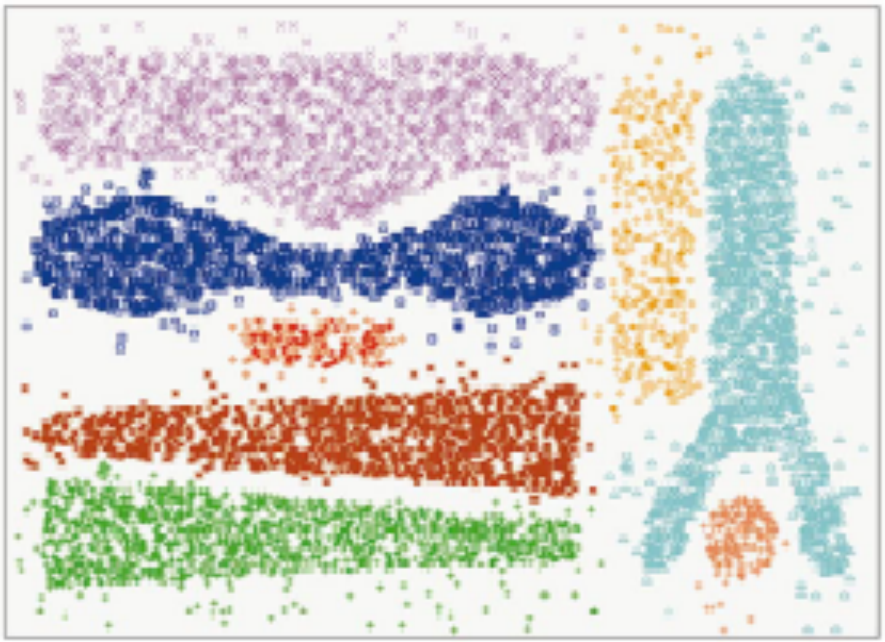
- Phase 2: Use Hierarchical Agglomerative Clustering to merge sub-clusters
  - Two clusters are combined if the *resulting cluster shares certain properties with the constituent clusters*
  - Two key properties used to model cluster similarity:
    - Relative Interconnectivity: Absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters
    - Relative Closeness: Absolute closeness of two clusters normalized by the internal closeness of the clusters

Slide from Tan, Steinbach, Kumar

CHAMELEON (Clustering Complex Objects)







Outline

- Extensions to Hierarchical Clustering
  - BIRCH, CHAMELEON
- Density-based Clustering
  - DBSCAN, OPTICS, DENCLUE
- Grid-Based Clustering
  - STING, CLIQUE
- Probabilistic Clustering
  - EM Clustering
- Clustering High-Dimensional Data
  - Subspace Clustering, Spectral Clustering
- Clustering Graphs and Network Data

Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features
  - discover clusters of arbitrary shape
  - handle noise
  - one scan
  - need density parameter as termination criterion
- Several interesting studies
  - DBSCAN: Ester, et al. KDD ‘96
  - OPTICS: Ankerst, et al., SIGMOD ‘99
  - DENCLUE: Hinneburg and Keim, KDD ‘98
  - CLIQUE: Agrawal, et al., SIGMOD ‘98

Slide from Han, Kamber, and Pei

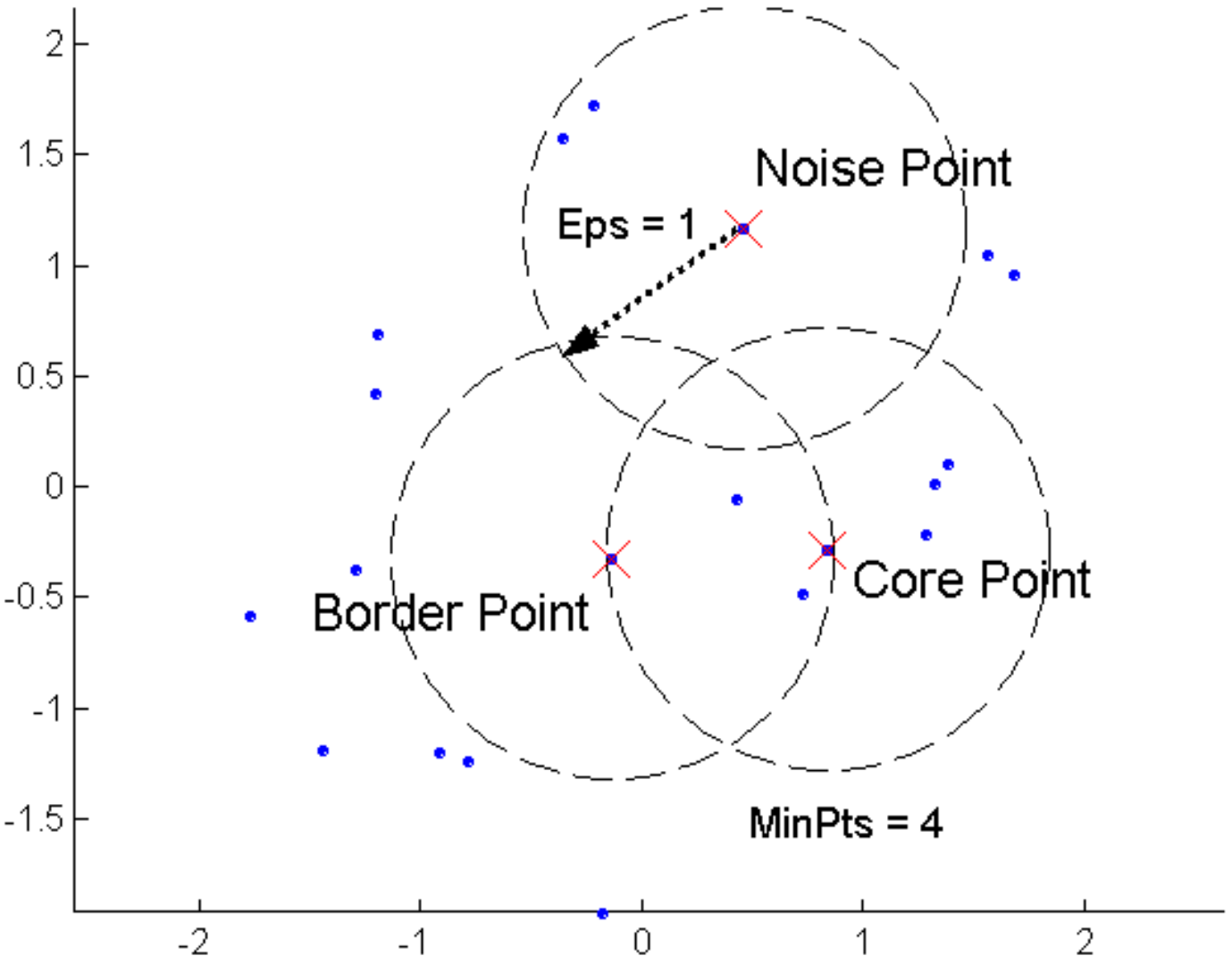
DBSCAN

DBSCAN is a density-based algorithm.

- Density = number of points within a specified radius (Eps)
- A point is a core point if it has more than a specified number of points (MinPts) within Eps
  - These are points that are at the interior of a cluster
- A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A noise point is any point that is not a core point or a border point.

Slide from Tan, Steinbach, Kumar

DBSCAN: Core, Border, and Noise Pts

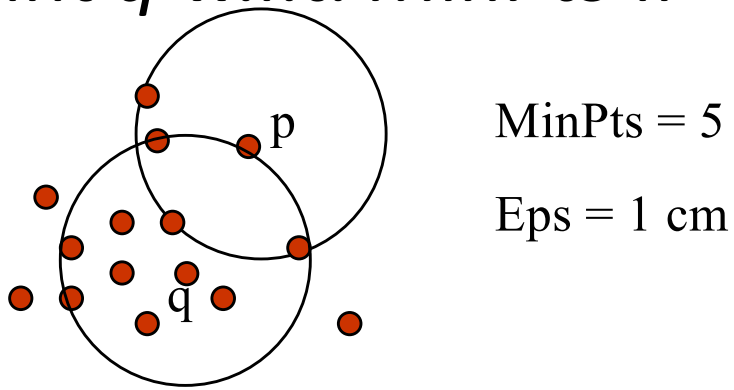


Slide from Tan, Steinbach, Kumar

Density-based clustering: Basic Concepts

- Two parameters:
  - Eps: maximum radius of the neighborhood
  - MinPts: minimum number of points in an Eps-neighborhood of that point
- $N_{eps}(q)$: {p belongs to D | dist(p,q) ≤ Eps}
- Directly density-reachable: A point *p* is directly density-reachable from a point *q* w.r.t. *MinPts* if
  - *p* belongs to $N_{eps}(q)$
  - core point condition:

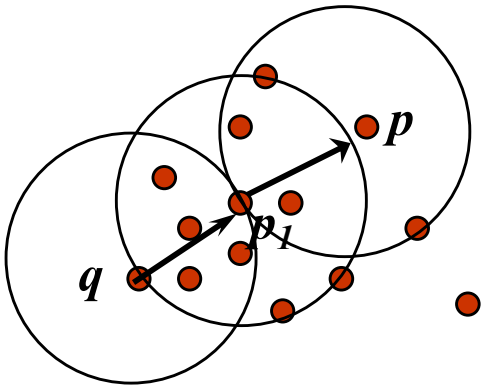
    $|N_{eps}(q)| \geq MinPts$



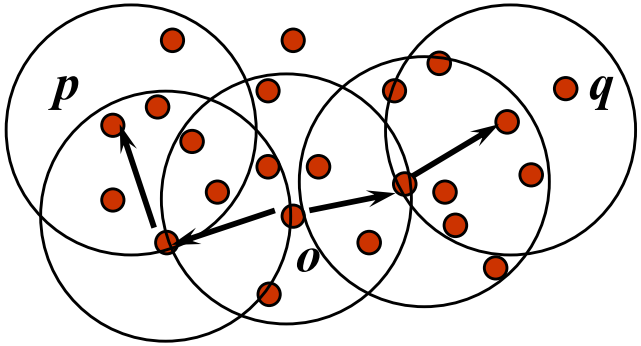
Slide from Han, Kamber, and Pei

Density-Reachable and Density-Connected

- Density-reachable:
  - A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i

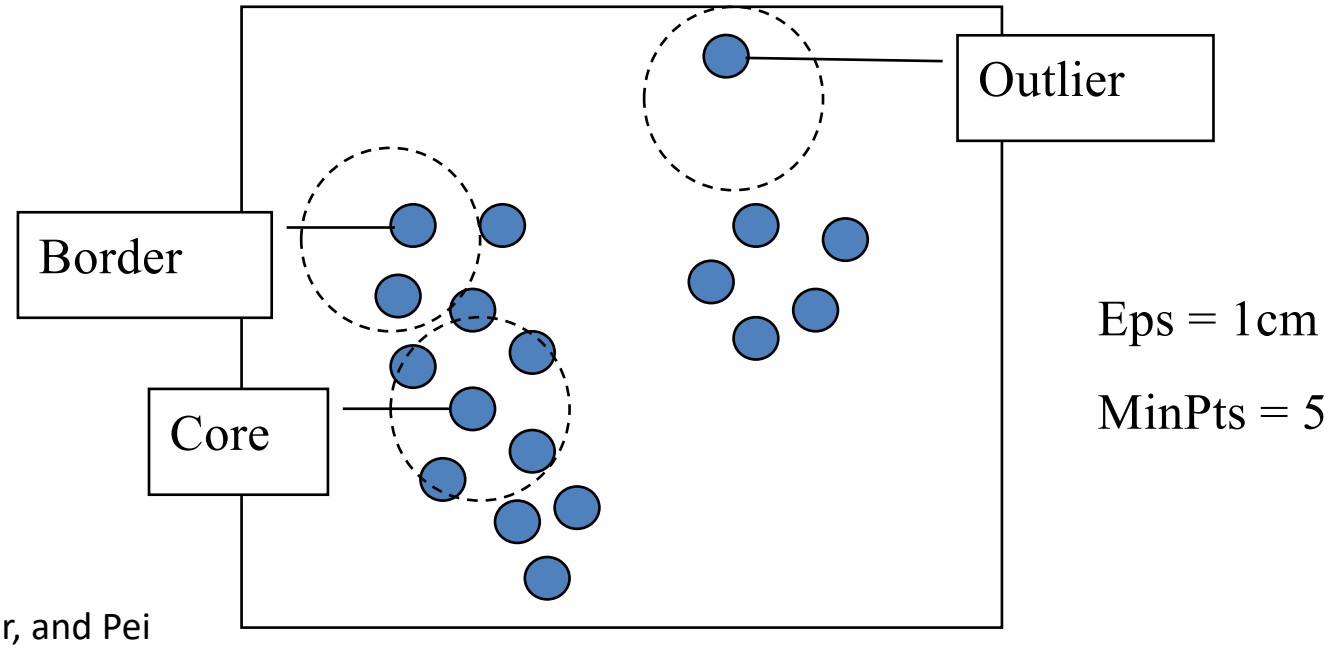


- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



DBSCAN

- DBSCAN: Density-Based Spectral Clustering of Applications with Noise
- Relies on density-based notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



Slide from Han, Kamber, and Pei

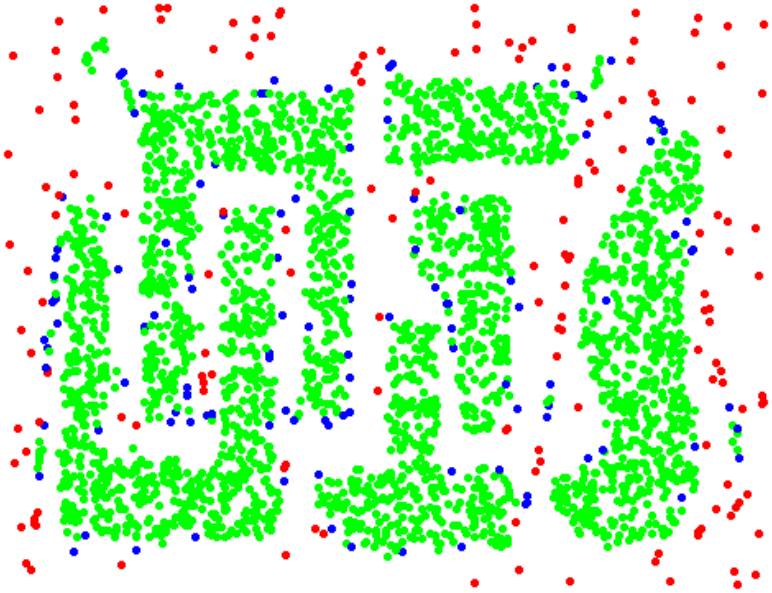
DBSCAN - Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*
- If *p* is a core point, a cluster is formed
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed
- *If a spatial index is used, the computational complexity of DBSCAN is O(nlogn), where n is the number of database objects. Otherwise, the complexity is* $O(n^2)$

Slide from Han, Kamber, and Pei

DBSCAN: Core, Border, and Noise Pts

Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

Slide from Tan, Steinbach, Kumar

DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

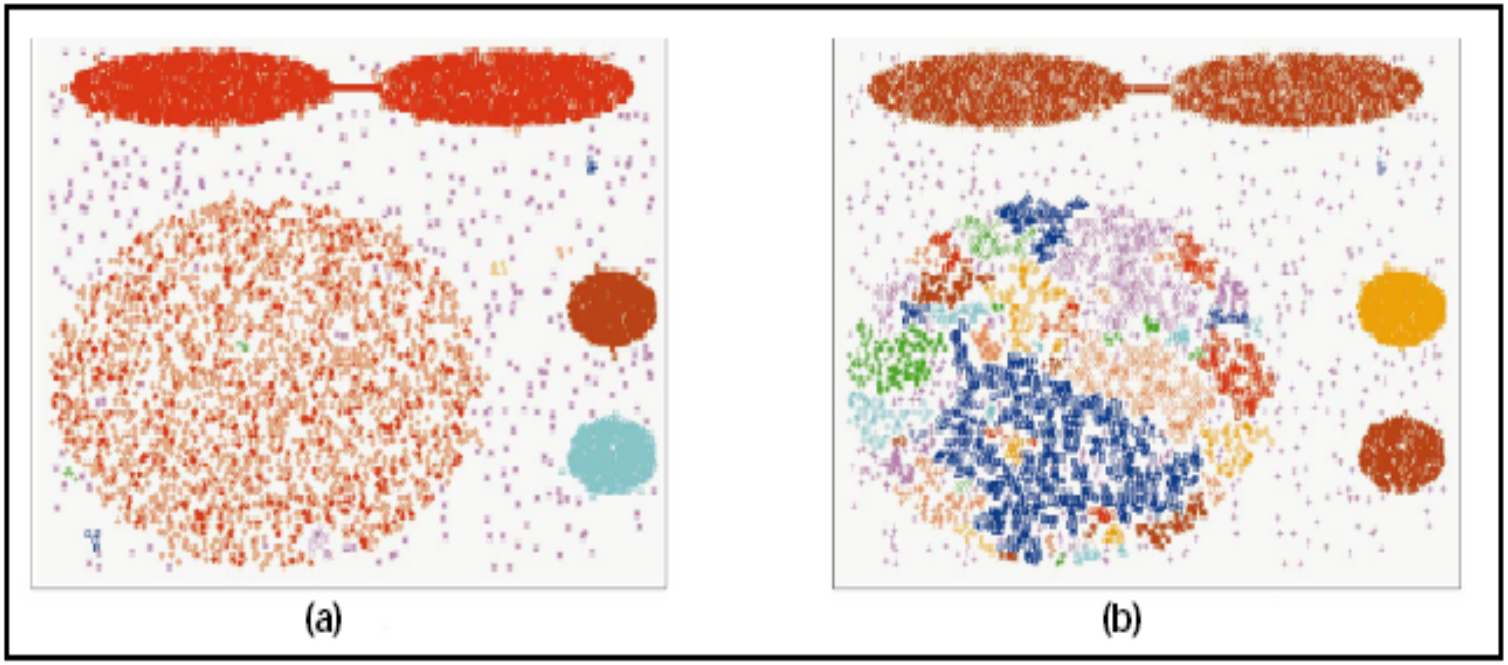
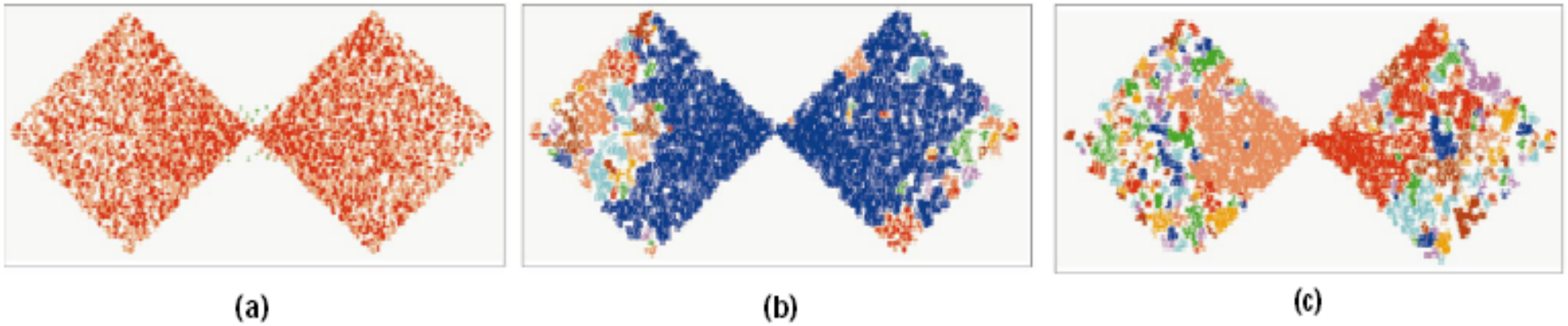


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



Slide from Han, Kamber, and Pei

OPTICS: A Cluster-Ordering Method

- OPTICS: Ordering Points To Identify the Clustering Structure
  - Ankerst, Breunig, Kriegel, and Sander (SIGMOD'99)
- Produces a special order of the database wrt its density-based clustering structure
- This cluster-ordering contains info equiv to the density-based clusterings corresponding to a broad range of parameter settings
- Good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure
- Can be represented graphically or using visualization techniques

Slide from Han, Kamber, and Pei

OPTICS extends DBSCAN

- Index-based:
    - k = number of dimensions
    - N = 20
    - p = 75%
    - M = N(1-p) = 5
  - Complexity: O(*NlogN*)
- Core Distance
- Reachability Distance

Slide from Han, Kamber, and Pei

Core Distance

Definition 5: (core-distance of an object p)

Let p be an object from a database D, let ε be a distance value, let $N_\varepsilon(p)$ be the ε-neighborhood of p, let *MinPts* be a natural number and let *MinPts-distance(p)* be the distance from p to its *MinPts*' neighbor. Then, the *core-distance* of p is defined as $\textit{core-distance}_{\varepsilon,\textit{MinPts}}(p) =$

$$\begin{cases} \textit{UNDEFINED}, \textit{ if } \textit{Card}(N_\varepsilon(p)) < \textit{MinPts} \\ \textit{MinPts-distance}(p), \text{ otherwise} \end{cases}$$

Slide from Han, Kamber, and Pei

Reachability Distance

Definition 6: (reachability-distance object p w.r.t. object o)

Let p and o be objects from a database D, let $N_\varepsilon(o)$ be the ε-neighborhood of o, and let $MinPts$ be a natural number. Then, the *reachability-distance* of p with respect to o is defined as $reachability\text{-}distance_{\varepsilon, MinPts}(p, o) =$

$$\begin{cases} UNDEFINED, \text{ if } \left|N_\varepsilon(o)\right| < MinPts \\ max(core\text{-}distance(o), distance(o, p)), \text{otherwise} \end{cases}$$

Slide from Han, Kamber, and Pei

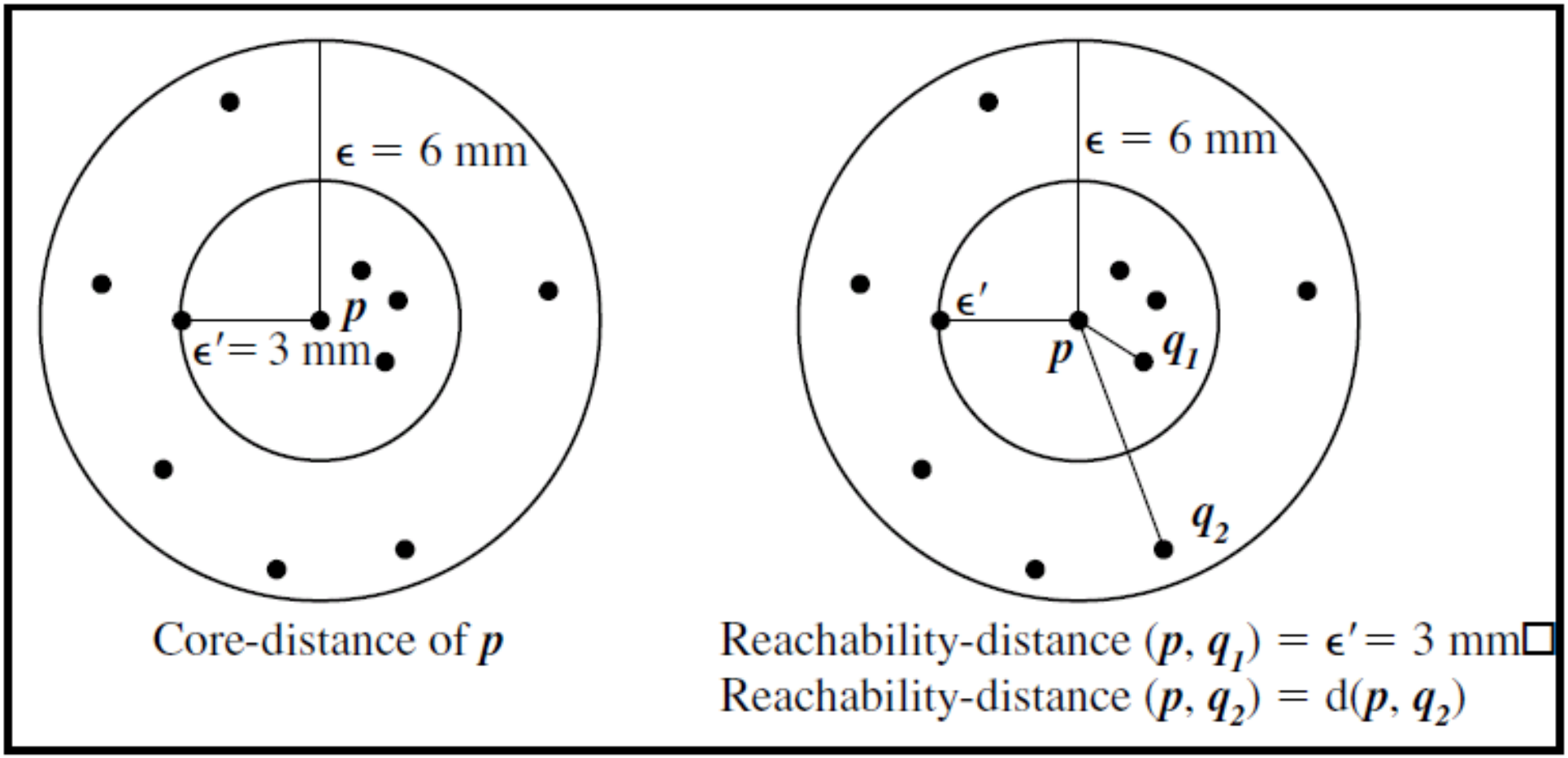


Figure 10.16: OPTICS terminology. Based on [ABKS99].

Slide from Han, Kamber, and Pei

DENCLUE: Using Statistical Density Function

- DENsity-based CLUstEring by Hinneburg & Keim (KDD'98)
- Using statistical density functions:

$$f_{Gaussian}(x,y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

influence of y on x

$$f^D_{Gaussian}(x) = \sum_{i=1}^{N} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

total influence on x

$$\nabla f^D_{Gaussian}(x,x_i) = \sum_{i=1}^{N} (x_i - x) \cdot e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

gradient of x in the direction of x_i

- Major features
  - Solid mathematical foundation
  - Good for data sets with large amounts of noise
  - Allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets
  - Significant faster than existing algorithm (e.g., DBSCAN)
  - But needs a large number of parameters

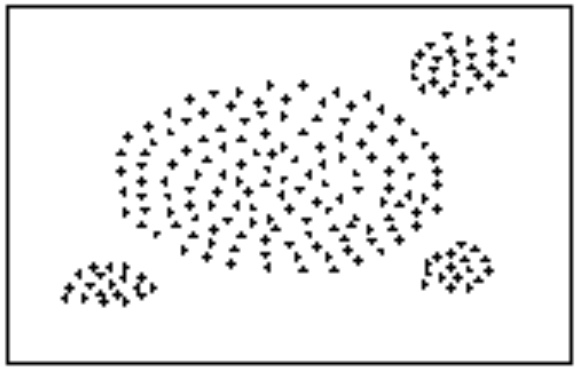
Slide from Han, Kamber, and Pei

DENCLUE

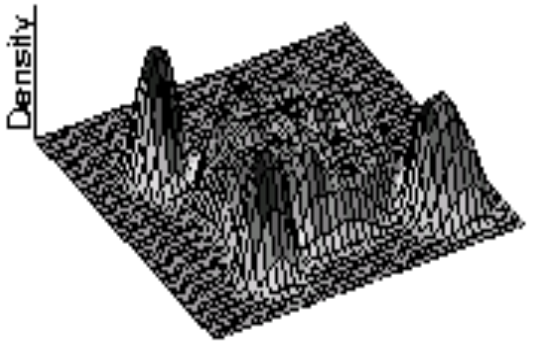
- Uses grid cells but only keeps information about grid cells that do actually contain data points and manages these cells in a tree-based access structure
- Influence function: describes the impact of a data point within its neighborhood
- Overall density of the data space can be calculated as the sum of the influence function of all data points
- Clusters can be determined mathematically by identifying density attractors
- Density attractors are local maximal of the overall density function
- Center defined clusters: assign to each density attractor the points density attracted to it
- Arbitrary shaped cluster: merge density attractors that are connected through paths of high density (> threshold)

Slide from Han, Kamber, and Pei

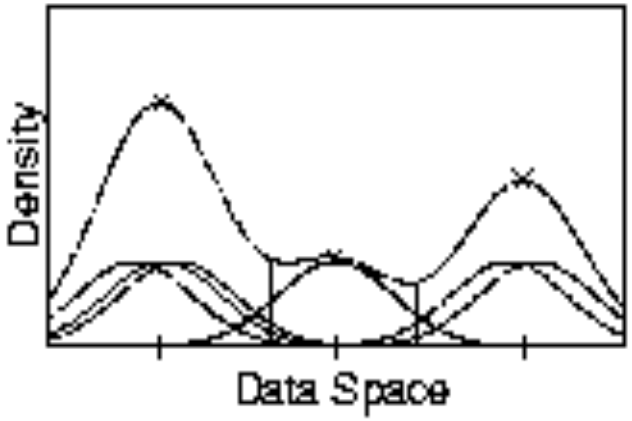
Density Attractor



(a) Data Set

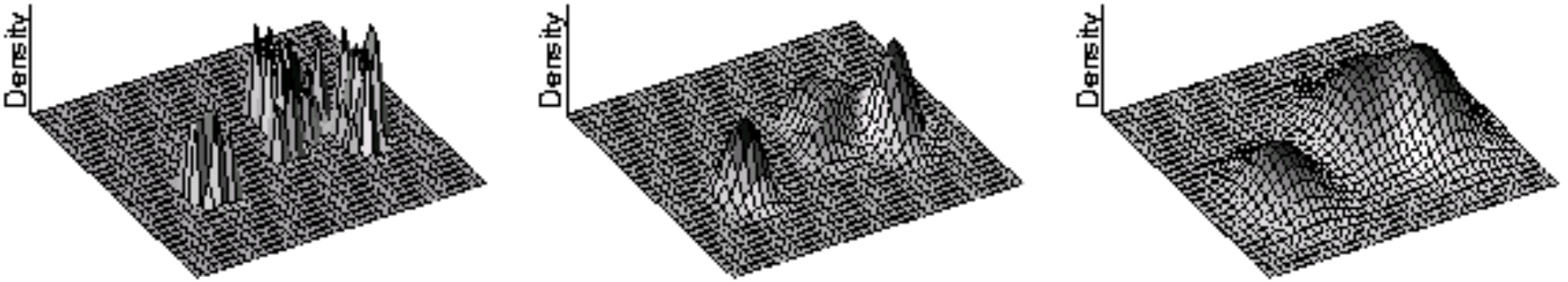


(c) Gaussian



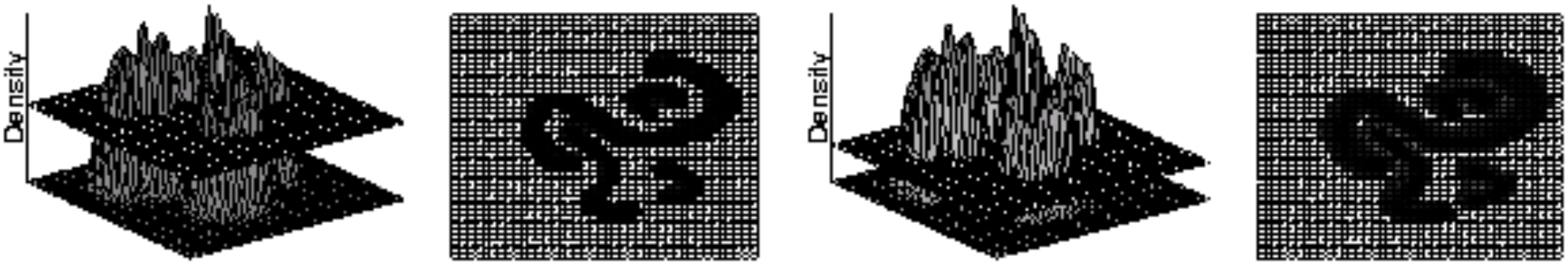
Slide from Han, Kamber, and Pei

Center-Defined and Arbitrary



(a) $\sigma = 0.2$ (b) $\sigma = 0.6$ (d) $\sigma = 1.5$

Figure 3: Example of Center-Defined Clusters for different σ



(a) $\xi = 2$ (b) $\xi = 2$ (c) $\xi = 1$ (d) $\xi = 1$

Figure 4: Example of Arbitray-Shape Clusters for different ξ

Slide from Han, Kamber, and Pei

Outline

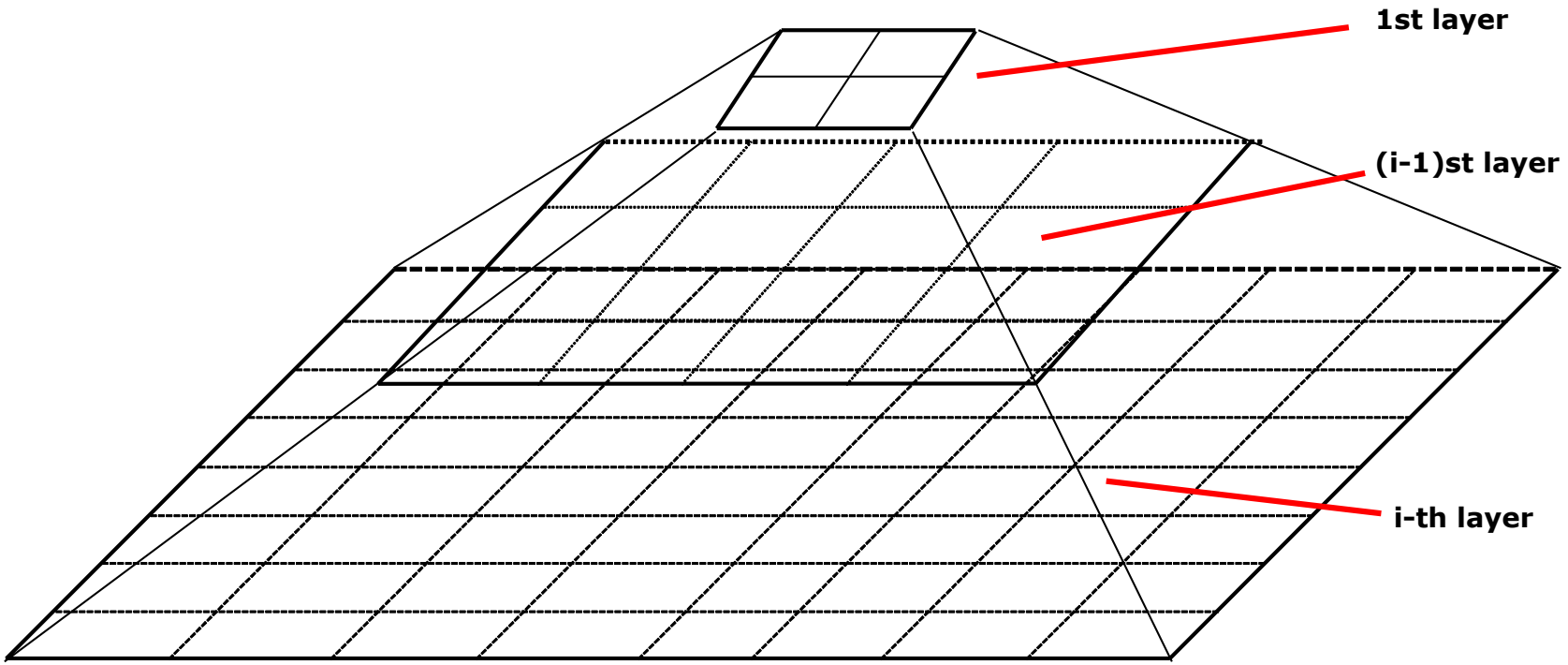
- Extensions to Hierarchical Clustering
  - BIRCH, CHAMELEON
- Density-based Clustering
  - DBSCAN, OPTICS, DENCLUE
- Grid-Based Clustering
  - STING, CLIQUE
- Probabilistic Clustering
  - EM Clustering
- Clustering High-Dimensional Data
  - Subspace Clustering, Spectral Clustering
- Clustering Graphs and Network Data

Grid-based Clustering Method

- Using multi-resolution grid data structure
- Several interesting methods
  - STING (a STatistical INformation Grid approach) by Wang, Yang and Muntz (1997)
  - WaveCluster by Sheikholeslami, Chatterjee, and Zhang (VLDB'98)
    - A multi-resolution clustering approach using wavelet method
  - CLIQUE: Agrawal, et al. (SIGMOD'98)
    - Both grid-based and subspace clustering

STING: A Statistical Information Grid Approach

- Wang, Yang and Muntz (VLDB'97)
- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution



STING Method

- Each cell at a high level is partitioned into a number of smaller cells in the next lower level
- Statistical info of each cell is calculated and stored beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from parameters of lower level cell
  - *count*, *mean*, *s*, *min*, *max*
  - type of distribution—*normal*, *uniform*, etc.
- Use a top-down approach to answer spatial data queries
- Start from a pre-selected layer—typically with a small number of cells
- For each cell in the current level compute the confidence interval

STING Method

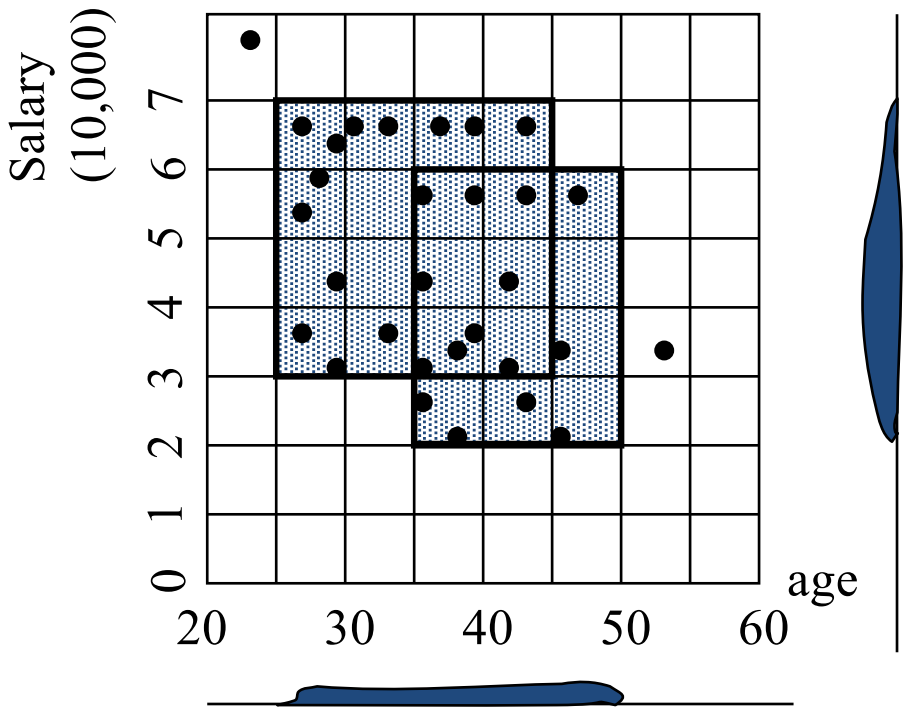
- Remove the irrelevant cells from further consideration
- When finish examining the current layer, proceed to the next lower level
- Repeat this process until the bottom layer is reached
- Advantages:
  - Query-independent, easy to parallelize, incremental update
  - *O(K),* where *K* is the number of grid cells at the lowest level
- Disadvantages:
  - All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected

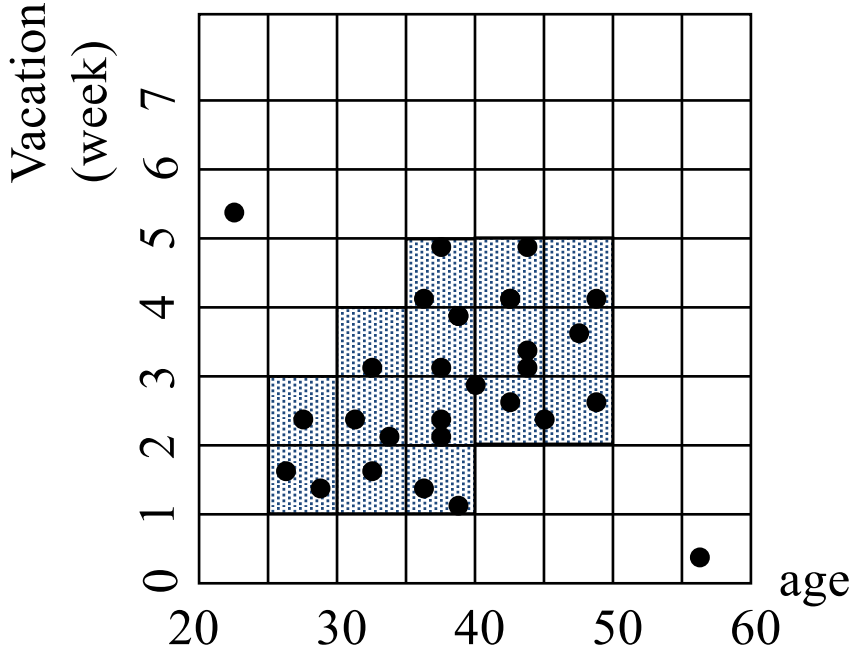
CLIQUE (Clustering In QUEst)

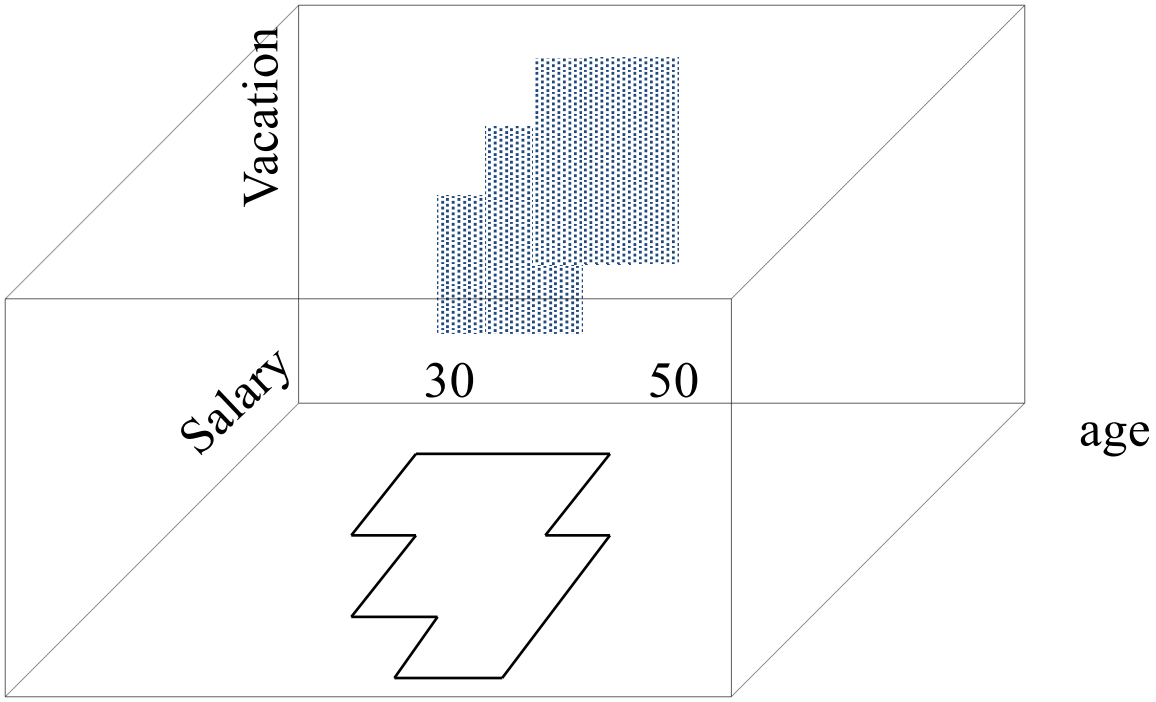
- Agrawal, Gehrke, Gunopulos, Raghavan (SIGMOD'98)
- Automatically identifying subspaces of a high dimensional data space that allow better clustering than original space
- CLIQUE can be considered as both density-based and grid-based
  - It partitions each dimension into the same number of equal length interval
  - It partitions an m-dimensional data space into non-overlapping rectangular units
  - A unit is dense if the fraction of total data points contained in the unit exceeds the input model parameter
  - A cluster is a maximal set of connected dense units within a subspace

CLIQUE – Major Components

- Partition the data space and find the number of points that lie inside each cell of the partition.
- Identify the subspaces that contain clusters using the Apriori principle
- Identify clusters
  - Determine dense units in all subspaces of interests
  - Determine connected dense units in all subspaces of interests.
- Generate minimal description for the clusters
  - Determine maximal regions that cover a cluster of connected dense units for each cluster
  - Determination of minimal cover for each cluster


Salary
(10,000)
0 1 2 3 4 5 6 7
20 30 40 50 60
age


Vacation
(week)
0 1 2 3 4 5 6 7
20 30 40 50 60
age


Vacation
Salary
30
50
age

CLIQUE – Method properties

- Strength
  - *automatically* finds subspaces of the highest dimensionality such that high density clusters exist in those subspaces
  - *insensitive* to the order of records in input and does not presume some canonical data distribution
  - scales *linearly* with the size of input and has good scalability as the number of dimensions in the data increases
- Weakness
  - The accuracy of the clustering result may be degraded at the expense of simplicity of the method

Outline

- Extensions to Hierarchical Clustering
  - BIRCH, CHAMELEON
- Density-based Clustering
  - DBSCAN, OPTICS, DENCLUE
- Grid-Based Clustering
  - STING, CLIQUE
- Probabilistic Clustering
  - EM Clustering
- Clustering High-Dimensional Data
  - Subspace Clustering, Spectral Clustering
- Clustering Graphs and Network Data

Probabilistic Hierarchical Clustering

- Algorithmic hierarchical clustering
  - nontrivial to choose a good distance measure
  - hard to handle missing attribute values
  - optimization goal not clear: heuristic, local search
- Probabilistic hierarchical clustering
  - Use probabilistic models to measure distance between clusters
  - Generative model: regard the set of data objects to be clustered as a sample of the underlying data generation mechanism to be analyzed
  - Easy to understand, same efficiency as algorithmic agglomerative clustering method, can handle partial observed data
- In practice, assume the generative models adopt common distribution functions, e.g., Gaussian or Bernoulli distribution, governed by parameters

- Given a set of 1-D points $X = \{x_1, \ldots, x_n\}$ for clustering analysis & assuming they are generated by a Gaussian distribution:

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- The probability that a point $x_i \in X$ is generated by the model

$$P(x_i|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- The likelihood that X is generated by the model:

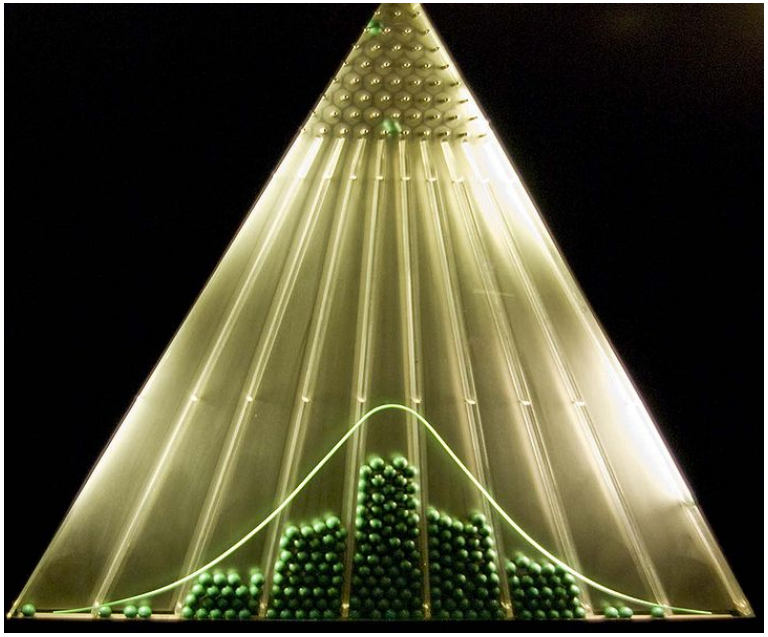
$$L(\mathcal{N}(\mu, \sigma^2) : X) = P(X|\mu, \sigma^2) = \prod_i = 1^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

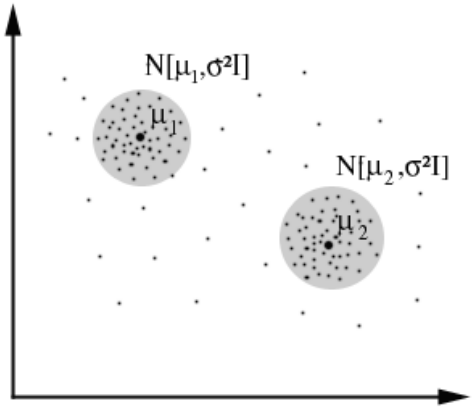
- The task of learning the generative model: find the parameters μ and σ^2 such that

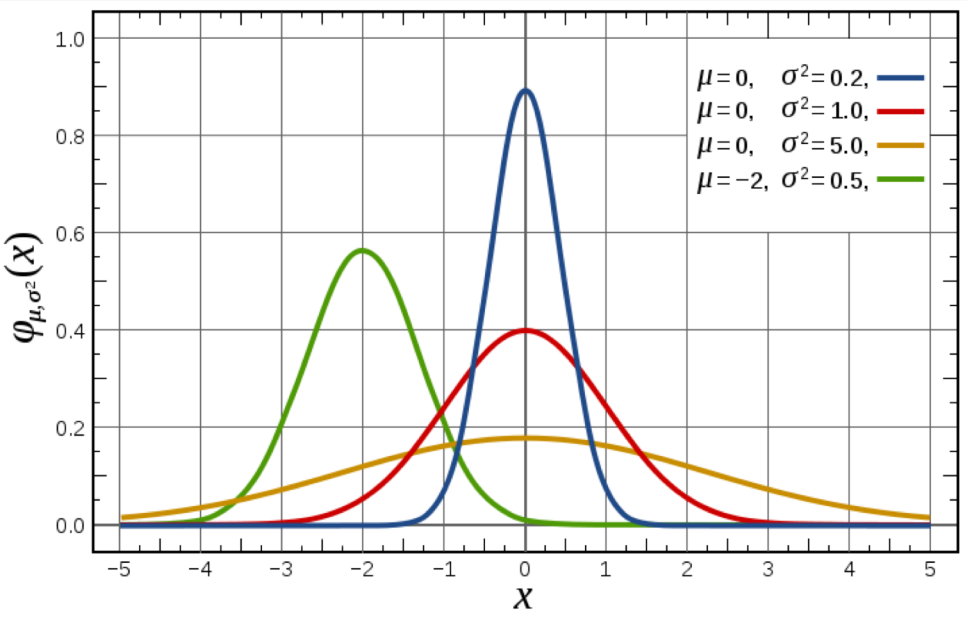
$$\mathcal{N}(\mu_0, \sigma_0^2) = \arg\max\{L(\mathcal{N}(\mu, \sigma^2) : X)\}$$

Maximum likelihood

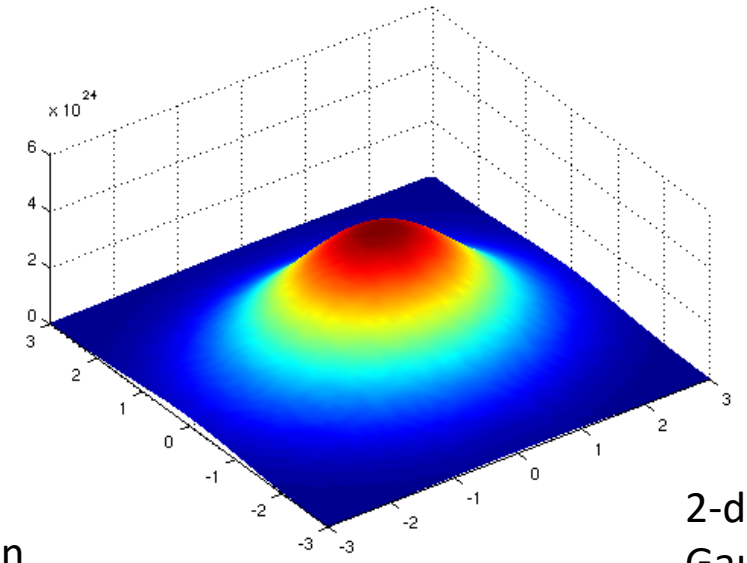
Gaussian Distribution







1-d Gaussian



2-d Gaussian

From wikipedia and http://home.dei.polimi.it

Probabilistic Hierarchical Clustering

- For a set of objects partitioned into *m* clusters $C_1, \ldots, C_m$, the quality can be measured by,

$$Q(\{C_1, \ldots, C_m\}) = \prod_{i=1}^{m} P(C_i)$$

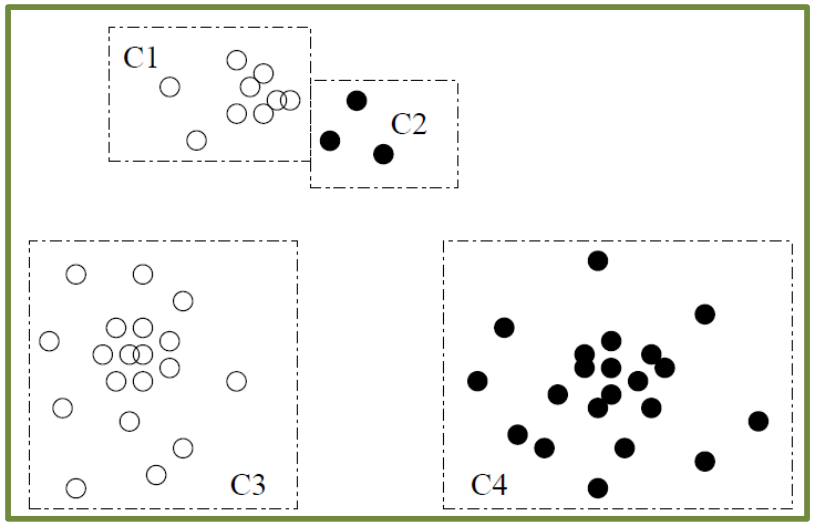
  where *P*() is the maximum likelihood

- If we merge two clusters C_{j1} and C_{j2} into a cluster $C_{j1} \cup C_{j2}$, then, the change in quality of the overall clustering is

$$Q((\{C_1, \ldots, C_m\} - \{C_{j_1}, C_{j_2}\}) \cup \{C_{j_1} \cup C_{j_2}\}) - Q(\{C_1, \ldots, C_m\})$$

$$= \frac{\prod_{i=1}^{m} P(C_i) \cdot P(C_{j_1} \cup C_{j_2})}{P(C_{j_1}) P(C_{j_2})} - \prod_{i=1}^{m} P(C_i)$$

$$= \prod_{i=1}^{m} P(C_i) \left(\frac{P(C_{j_1} \cup C_{j_2})}{P(C_{j_1}) P(C_{j_2})} - 1 \right)$$



- Distance between clusters C_1 and C_2:

$$dist(C_i, C_j) = -\log \frac{P(C_1 \cup C_2)}{P(C_1) P(C_2)}$$

Algorithm

- Algorithm: Progressively merge points and clusters

  Input: $D = \{o_1, ..., o_n\}$: a data set containing n objects

  Output: A hierarchy of clusters

  Method

  Create a cluster for each object $C_i = \{o_i\}$, $1 \le i \le n$;

  For i = 1 to n {

  Find pair of clusters C_i and C_j such that

  $C_i, C_j = \text{argmax}_{i \neq j} \{\log (P(C_i \cup C_j)/(P(C_i)P(C_j))\}$;

  If $\log (P(C_i \cup C_j)/(P(C_i)P(C_j)) > 0$ then merge C_i and C_j

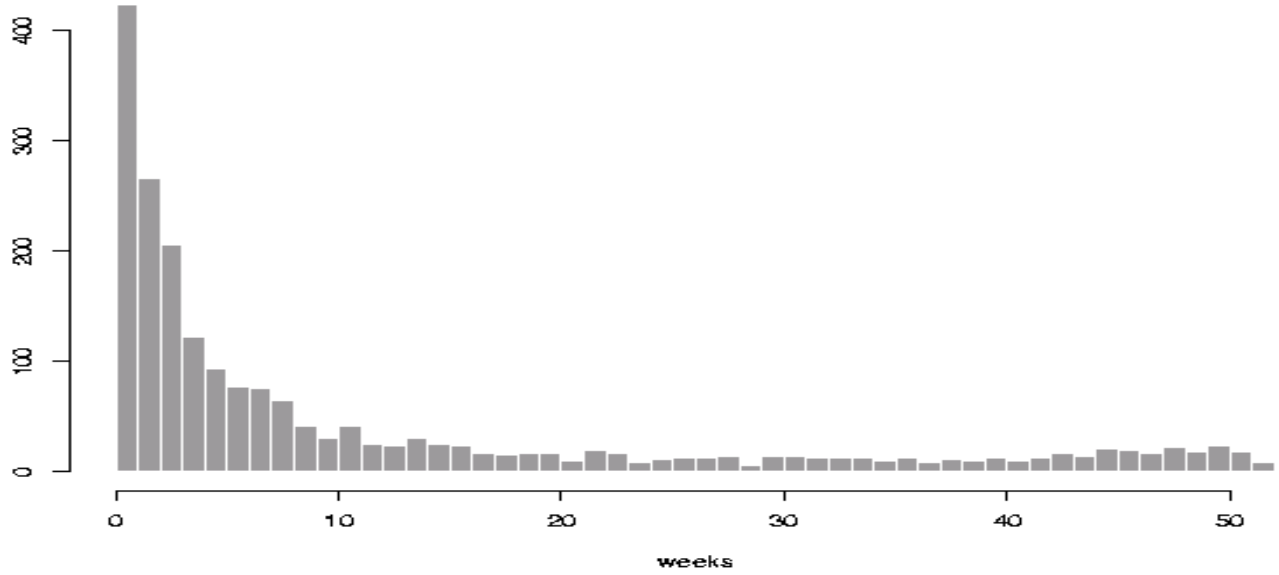
  }

Estimating Probability Densities

- Using probability densities is one way to describe data
- Finite mixtures of probability densities can be viewed as clusters
- Because we have a probability model, log-likelihood can be used to evaluate:

$$S_L(\theta) = -\sum_{i=1}^{n} \log f(x_i)$$

Mixture Models



$$f(x) = p\frac{(\lambda_1)^x e^{-\lambda_1}}{x!} + (1-p)\frac{(\lambda_2)^{52-x} e^{-\lambda_2}}{(52-x)!}$$

- Two stage model:
  - Assign data points to clusters
  - Assess fit of the model

Model-based Clustering

- A set C of *k* probabilistic clusters $C_1, \ldots, C_k$ with probability density functions $f_1, \ldots, f_k$ respectively, and their probabilities $w_1, \ldots, w_k$
- Probability of an object o generated by cluster C_j is $P(o|C_j) = \omega_j f_j(o)$
- Probability of *o* generated by the set of clusters **C** is $P(o|\boldsymbol{C}) = \sum_{j=1}^{k} \omega_j f_j(o)$
- The generation of the data is (each point ind.)

$$P(D|\boldsymbol{C}) = \prod_{i=1}^{n} P(o_i|\boldsymbol{C}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j f_j(o_i)$$

- Task: find a set C of k cluster to maximize P(D|C)

Model-Based Clustering

- Task: Find a set C of k probabilistic clusters s.t. $P(D|\boldsymbol{C})$ is maximized
- However, maximizing $P(D|\boldsymbol{C})$ is often intractable since the probability density function of a cluster can take an arbitrarily complicated form
- To make it computationally feasible (as a compromise), assume the probability density functions being some parameterized distributions

Univariate Gaussian Mixture Model

- O = {o_1, …, o_n} (n observed objects), Θ = {θ_1, …, θ_k} (parameters of the k distributions), and $P_j(o_i | \theta_j)$ is the probability that o_i is generated from the j-th distribution using parameter θ_j, we have

$$P(o_i|\Theta) = \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j) \qquad P(\mathbf{O}|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j)$$

- Univariate Gaussian mixture model
  - Assume the probability density function of each cluster follows a 1-d Gaussian distribution. Suppose that there are k clusters.
  - The probability density function of each cluster are centered at μ_j with standard deviation σ_j, θ_j, = (μ_j, σ_j), we have

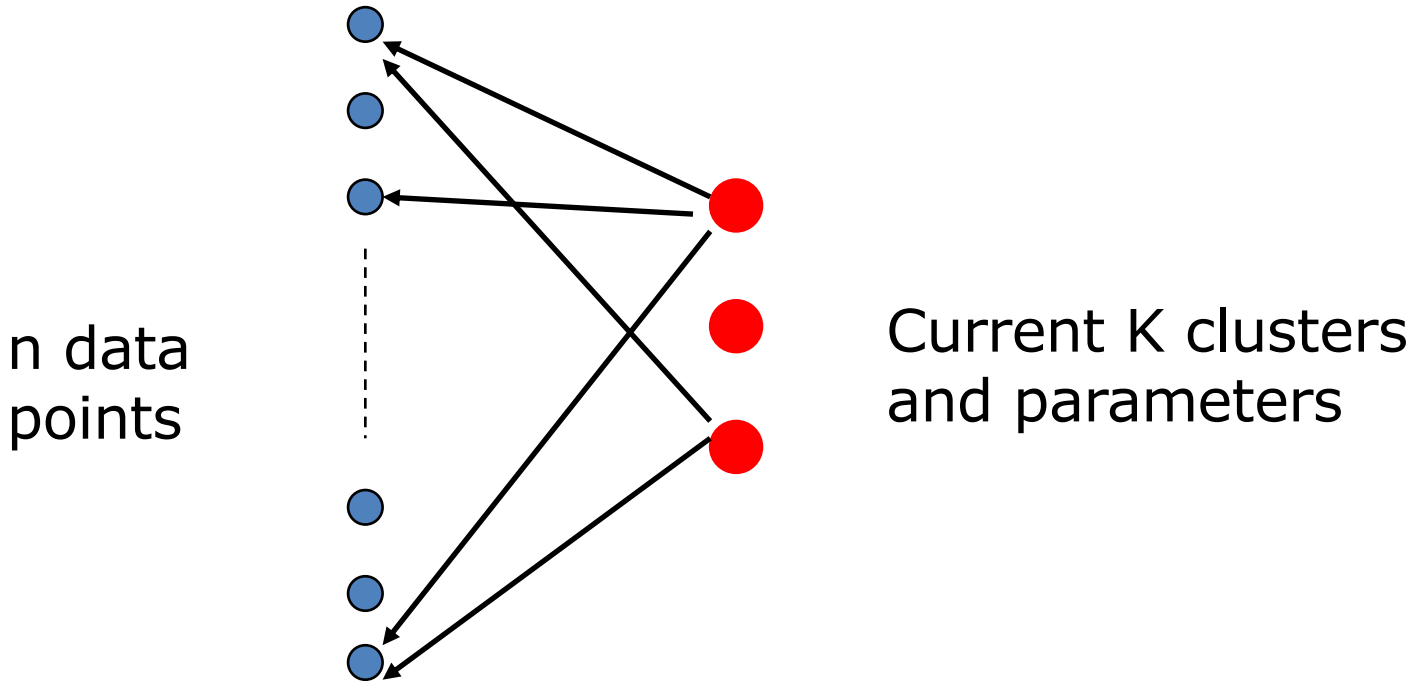
$$P(o_i|\Theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}} \qquad P(o_i|\Theta) = \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

$$P(\mathbf{O}|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

Mixture Models and EM

- How do we find the models to mix?
- EM (Expectation / Maximization)
  - common technique that converges to a solution for finding mixture models
- Assume multivariate normal components.
- EM approach
  - take an initial solution
  - calculate the probability that each point comes from each component and assign it (E-step)
  - re-estimate parameters for the components based on the new assignments (M-step)
  - repeat until convergence

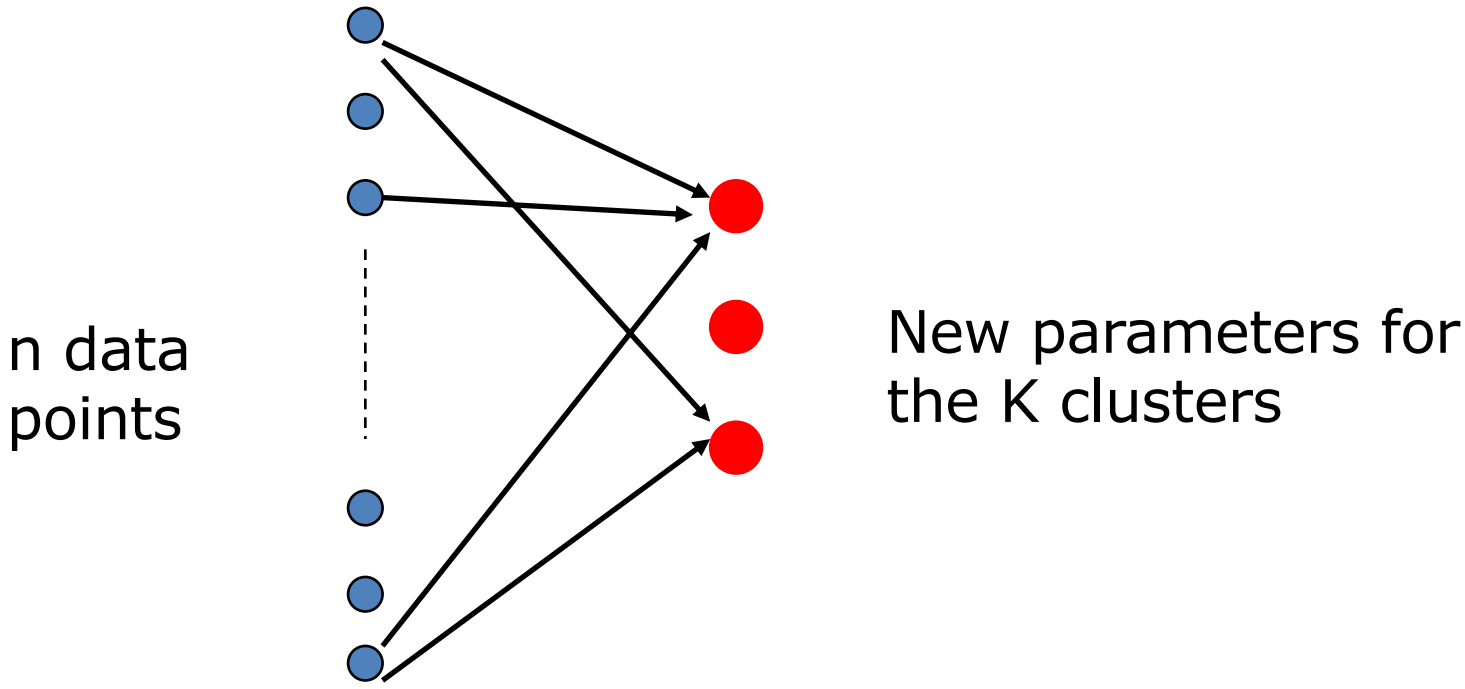
The E (Expectation) Step



E step: Compute p(data point i is in group k)

- **Expectation Step** (E-step): Given the current cluster centers, each object is assigned to the cluster whose center is closest to the object: An object is *expected to belong to the closest cluster*

The M (Maximization) Step



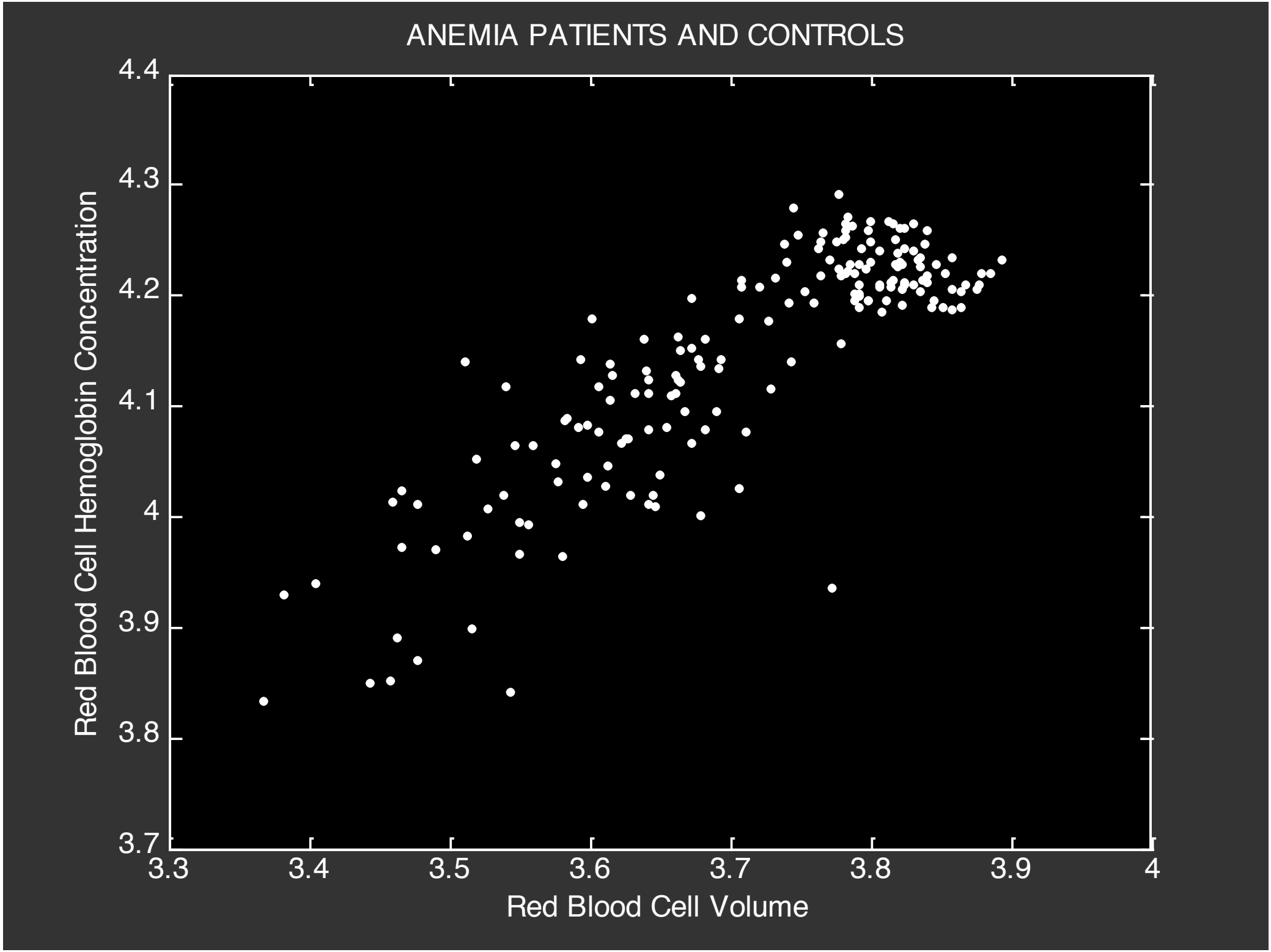
M step: Compute θ, given n data points and memberships

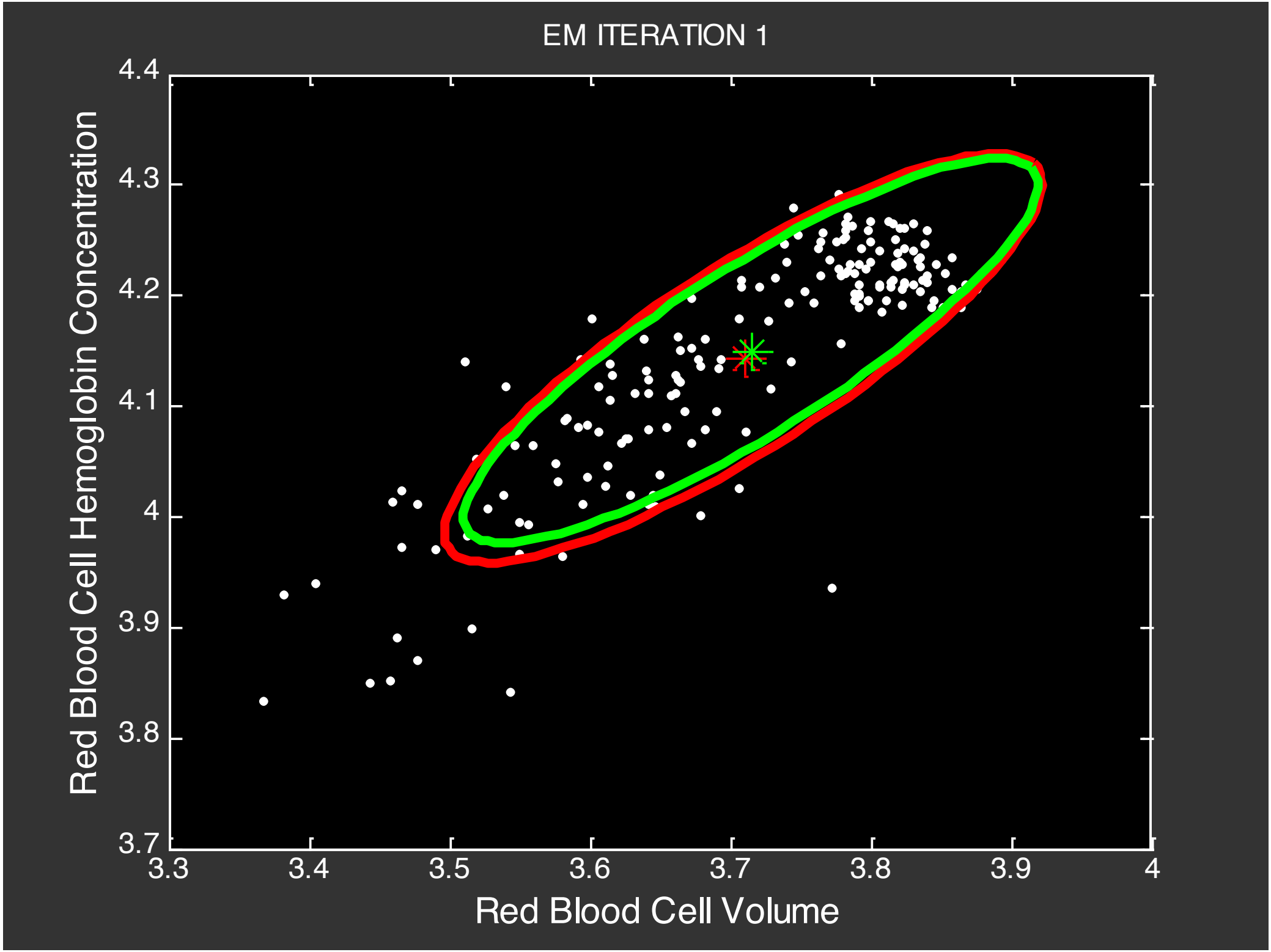
- **Maximization Step** (M-step): Given the cluster assignment, for each cluster, the algorithm *adjusts the center* so that *the sum of distance* from the objects assigned to this cluster and the new center is minimized

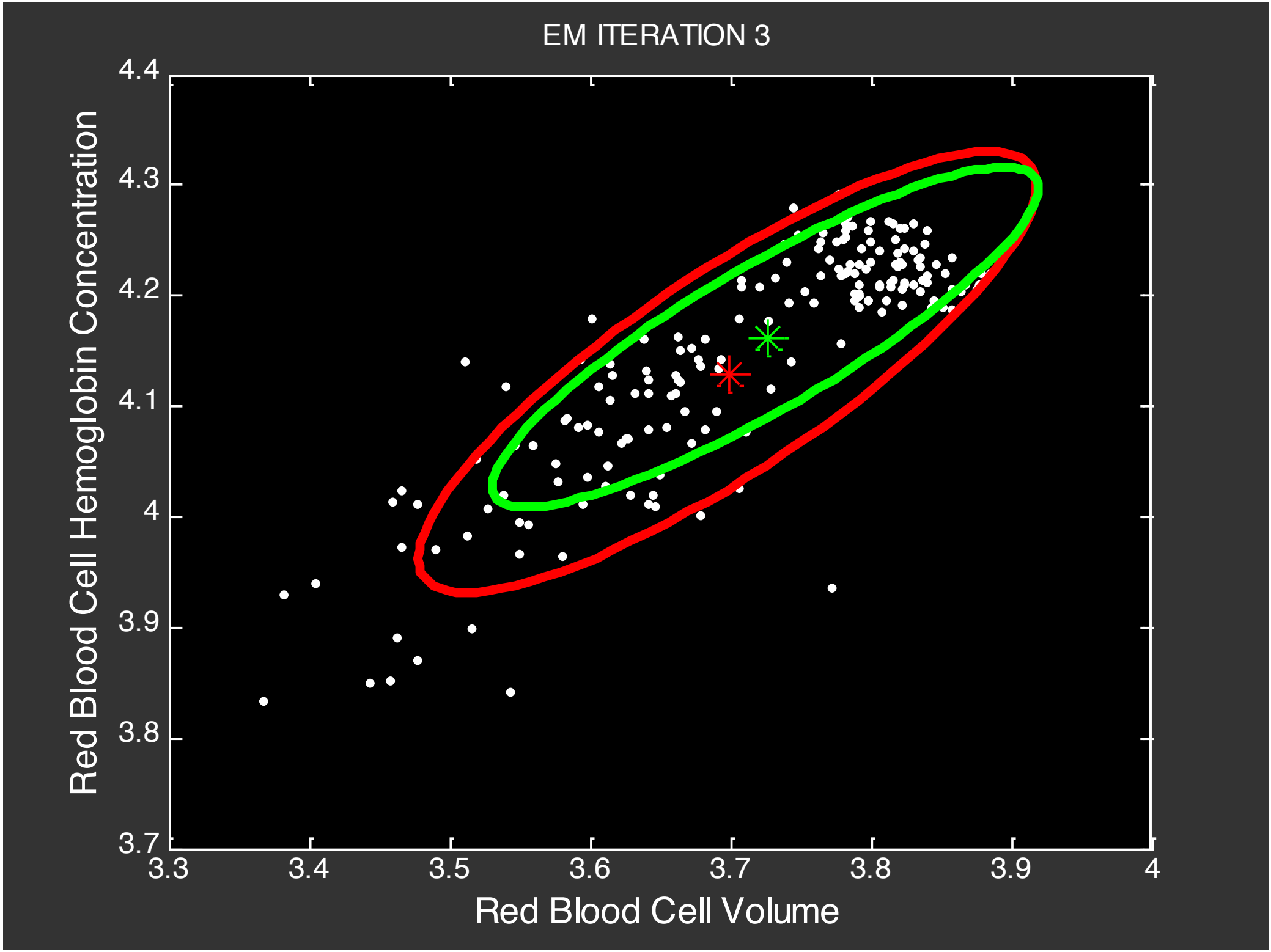
Mixtures and EM Learning

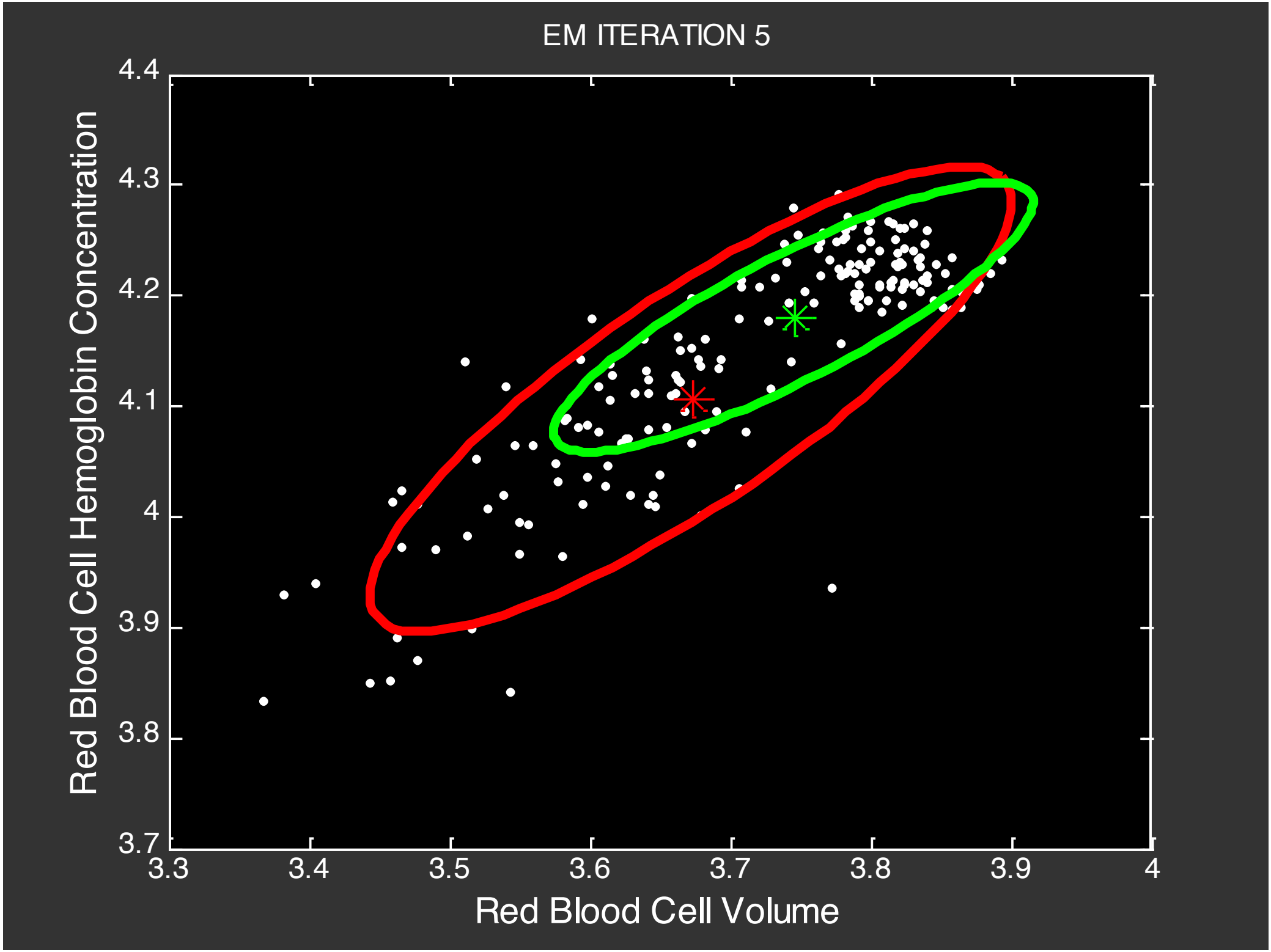
- Probabilistic assignment to clusters … not a partition

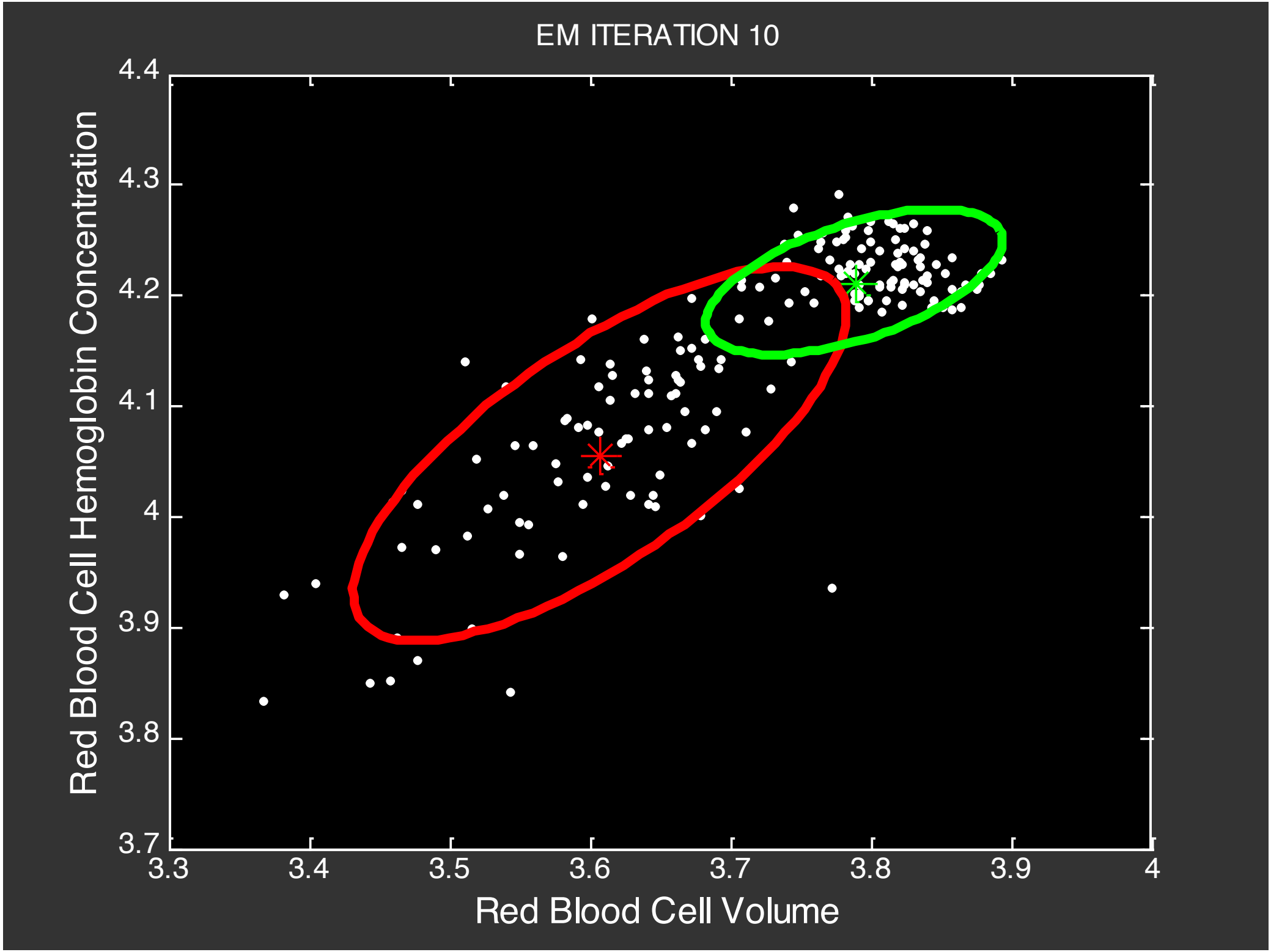
- K-means is a special case of EM
  - Gaussian mixtures with isotropic (diagonal, equi-variance)
  - Approximate the E-step by choosing most likely cluster (instead of using membership probabilities)
  - EM can be used more broadly to estimate generic distributions

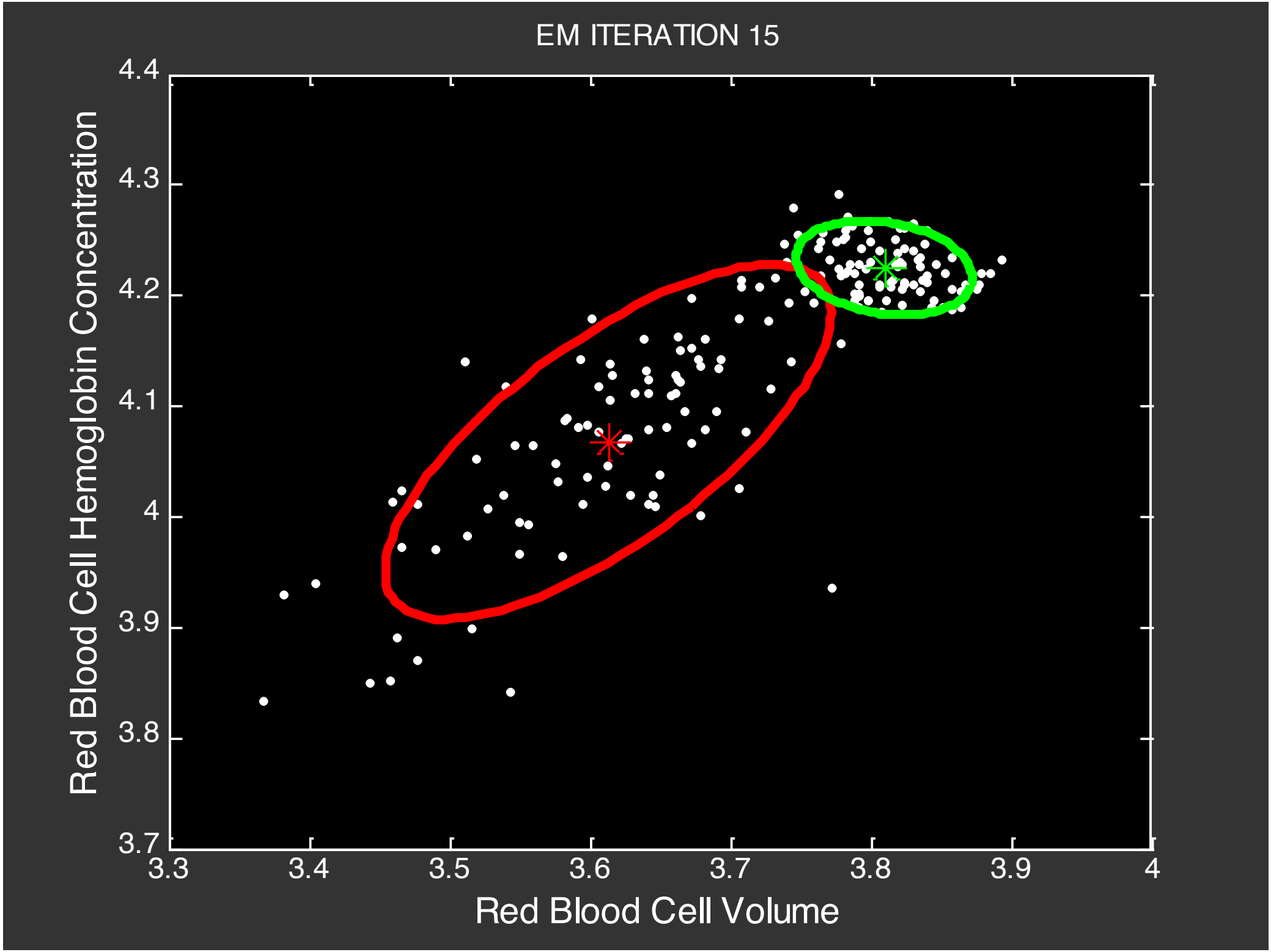

ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

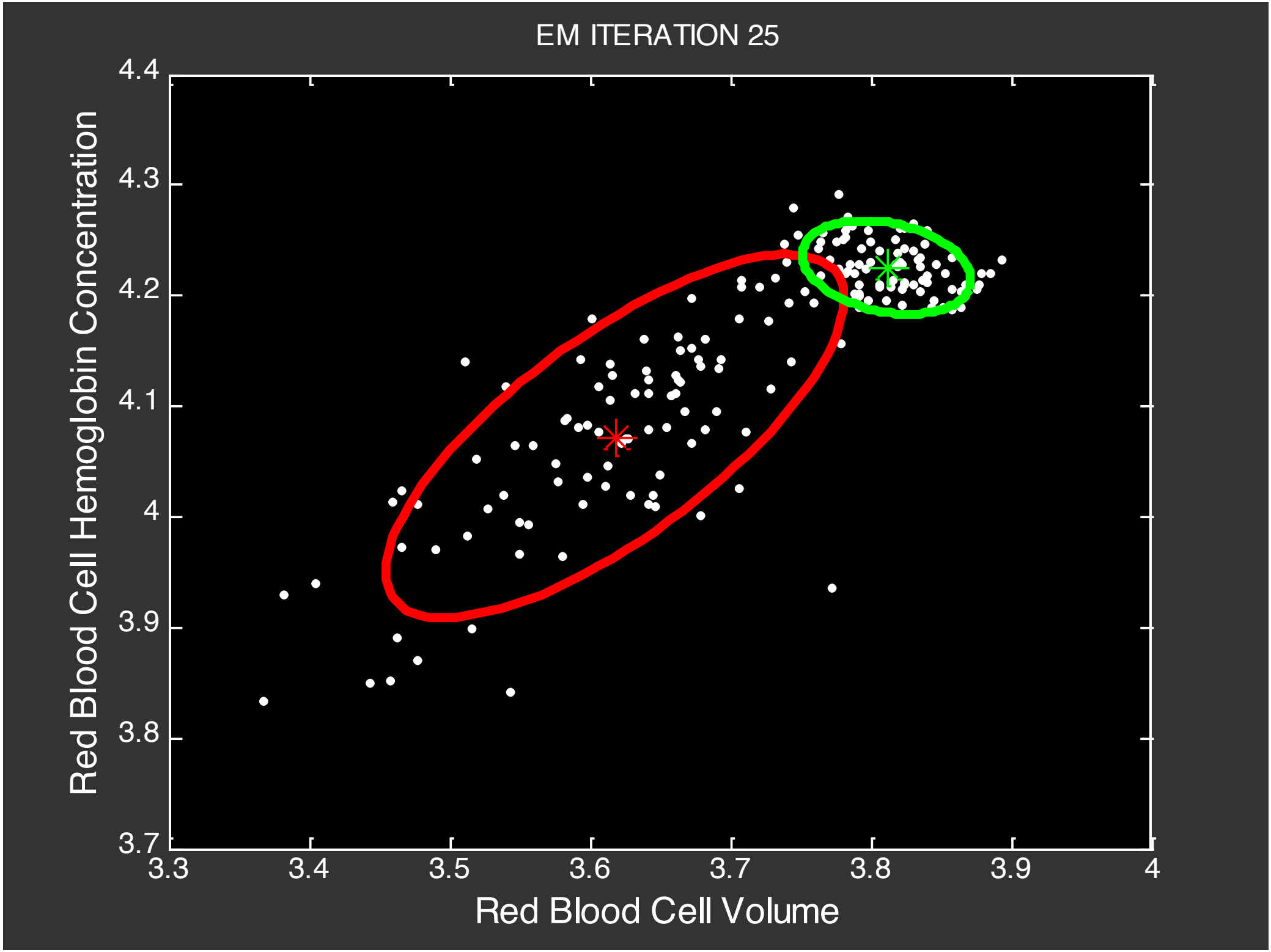

EM ITERATION 1
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

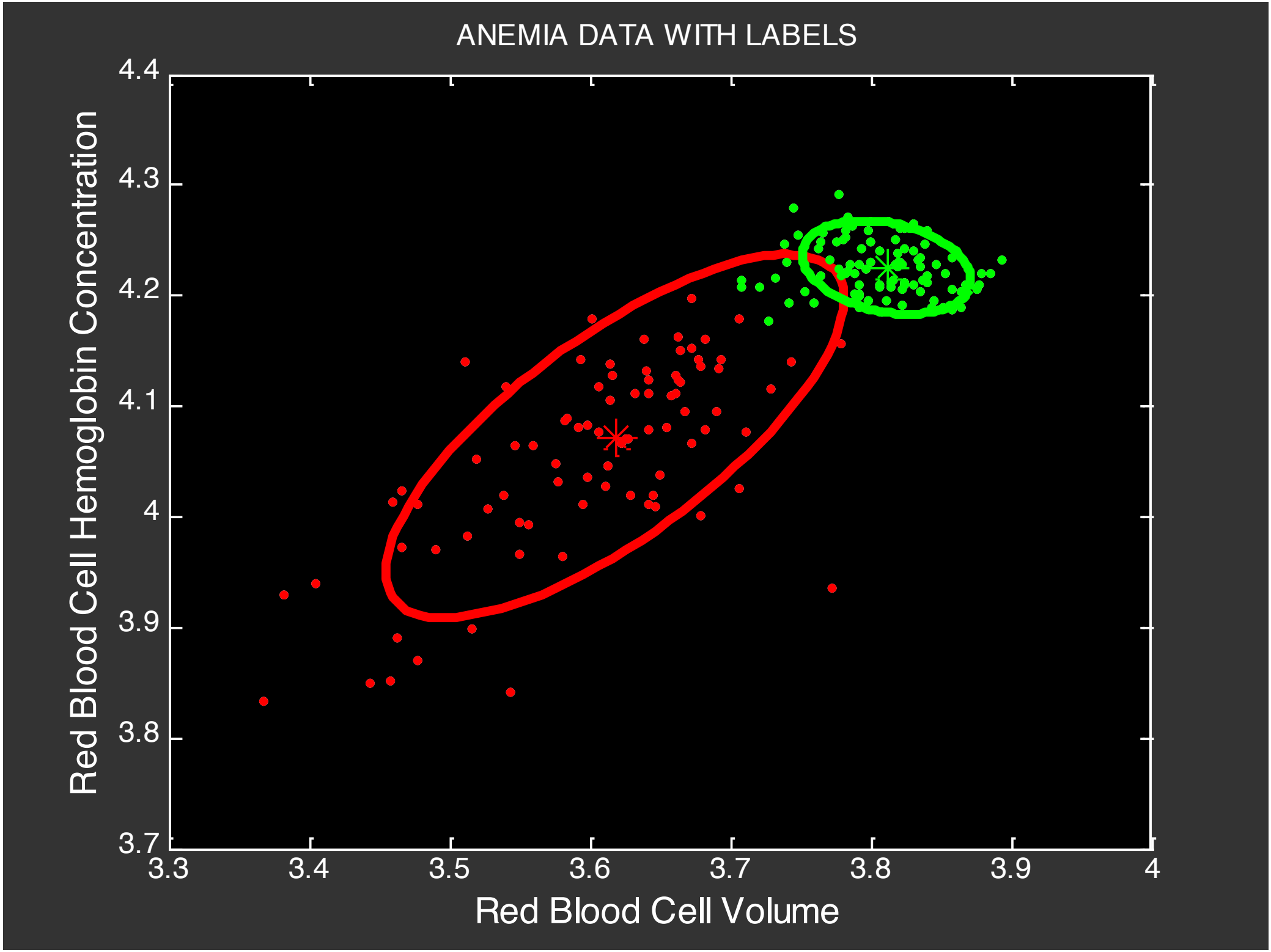

EM ITERATION 3
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

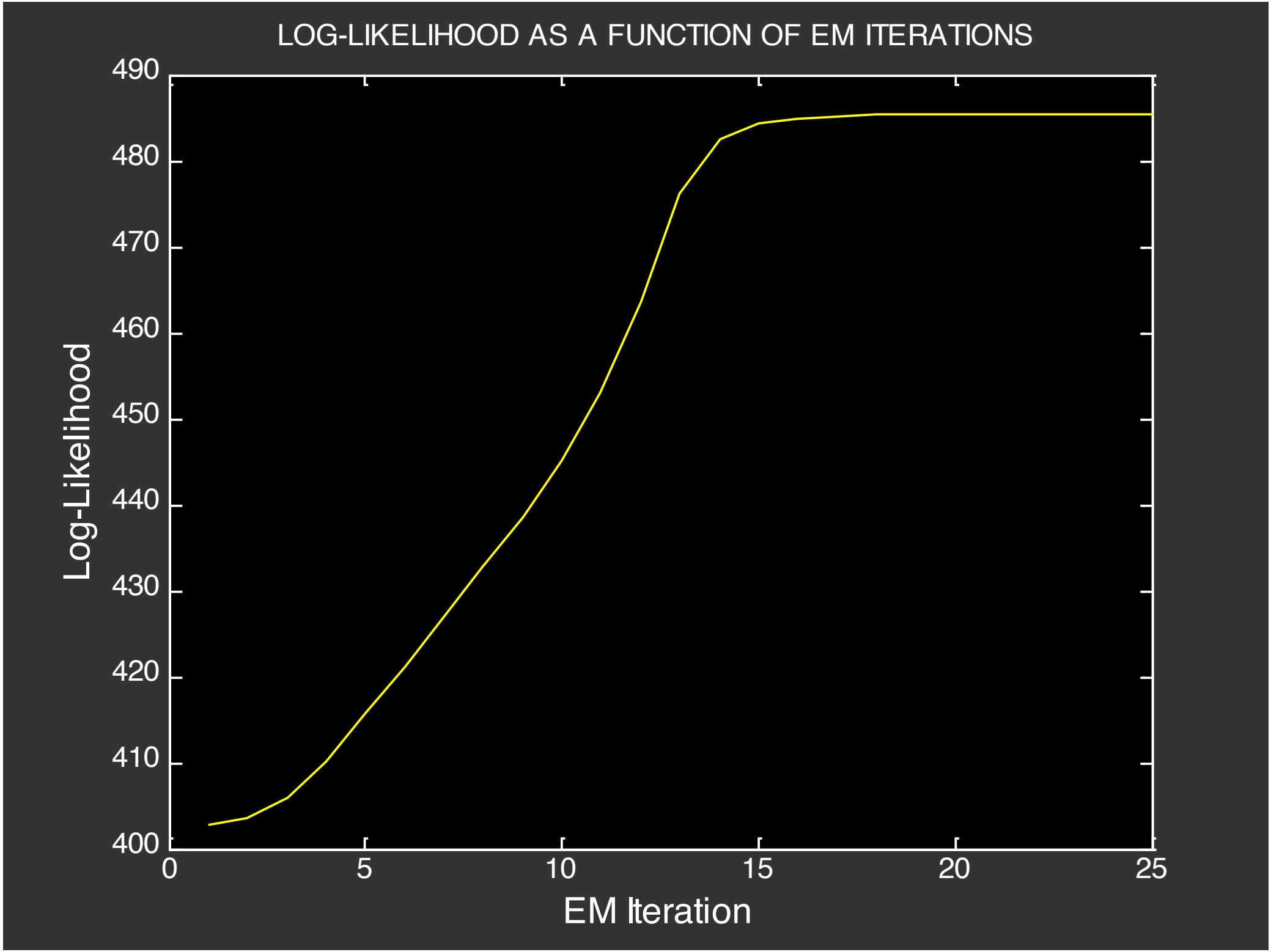

EM ITERATION 5
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 10
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 15
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 25
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


ANEMIA DATA WITH LABELS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


LOG-LIKELIHOOD AS A FUNCTION OF EM ITERATIONS
Log-Likelihood
EM Iteration
490
480
470
460
450
440
430
420
410
400
0
5
10
15
20
25

Selecting K in mixture models

- Can not just choose K that maximizes likelihood
  - Likelihood is always larger for larger K
- Model selection alternatives for choosing k:
  - in-sample: penalize for complexity
    - BIC: Bayesian information criterion
    - easy to implement, asymptotically correct
  - Bayesian: compute posterior p(k | data)
    - calculation may be difficult, includes calculating P(data | k)
  - Out-of-sample: cross – validation
    - score models by likelihood on test data
    - can be noisy on small data (sensitive to outliers)

Summary Mixture Models

- Strength
  - Mixture models are more general than partitioning and fuzzy clustering
  - Clusters can be characterized by a small number of parameters
  - The results may satisfy the statistical assumptions of the generative models
- Weakness
  - Converge to local optimal
  - Computationally expensive if the number of distributions is large or the data set contain very few observed data points
  - Need large data sets
  - Hard to estimate number of clusters

Outline

- Extensions to Hierarchical Clustering
  - BIRCH, CHAMELEON
- Density-based Clustering
  - DBSCAN, OPTICS, DENCLUE
- Grid-Based Clustering
  - STING, CLIQUE
- Probabilistic Clustering
  - EM Clustering
- Clustering High-Dimensional Data
  - Subspace Clustering, Spectral Clustering
- Clustering Graphs and Network Data

Clustering High-Dimensional Data

- Clustering high-dimensional data major challenges
  - many irrelevant dimensions may mask clusters
  - distance measure become meaningless
  - clusters may exist only in some subspaces
- Methods
  - Subspace-clustering: search for clusters in subspaces of high-dimensional space
  - Dimensionality reduction methods

Distance Measures in High-Dimensional Data

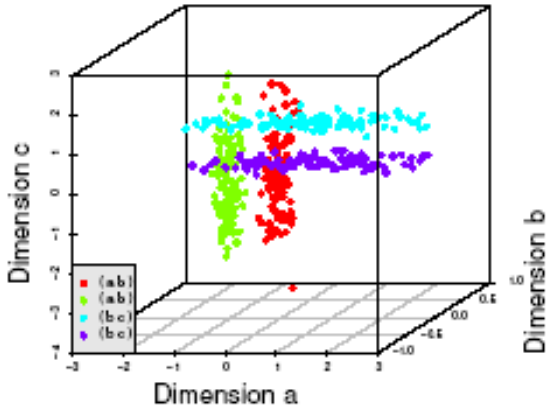
- Traditional distance measures may be dominated by noise in high dimensions
- Which pairs of customers are more similar?

Customer	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Ada	1	0	0	0	0	0	0	0	0	0
Bob	0	0	0	0	0	0	0	0	0	1
Cathy	1	0	0	0	1	0	0	0	0	1

- By Euclidean distance

$$dist(\text{Ada}, \text{Bob}) = dist(\text{Bob}, \text{Cathy}) = dist(\text{Ada}, \text{Cathy}) = \sqrt{2}$$

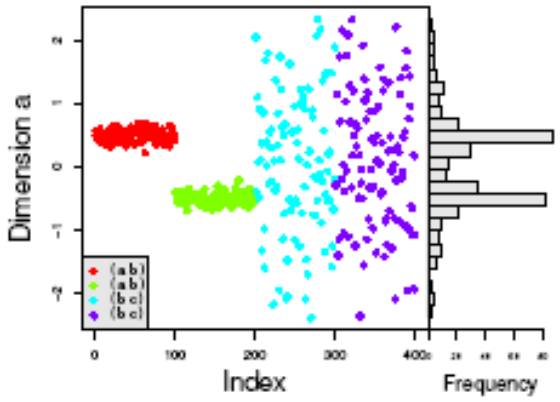
- Clustering should not only consider dimensions by also attributes
  - feature transformation: PCA, SVD
  - feature selection: find a subspace with good clusters



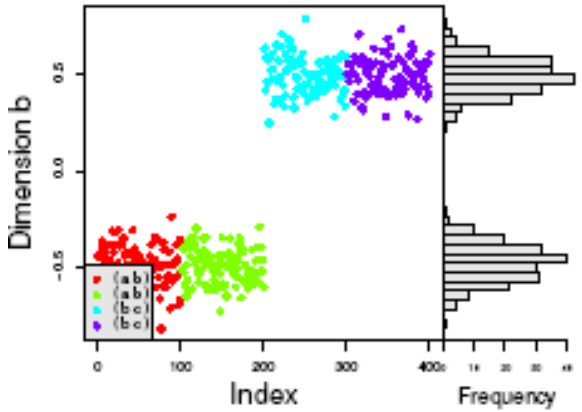
Why Subspace Clustering?

(adapted from Parsons et al. SIGKDD Explorations 2004)

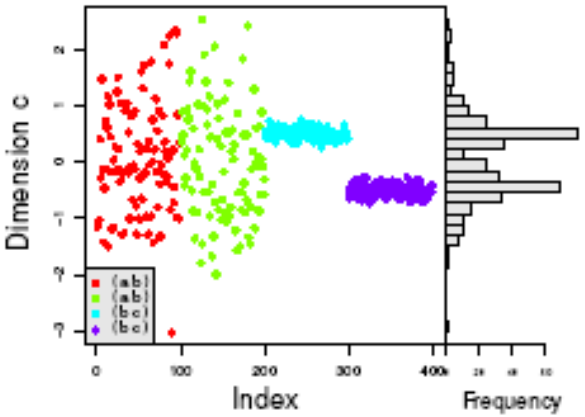
- Clusters may exist only in some subspaces
- Subspace-clustering: find clusters in all the subspaces



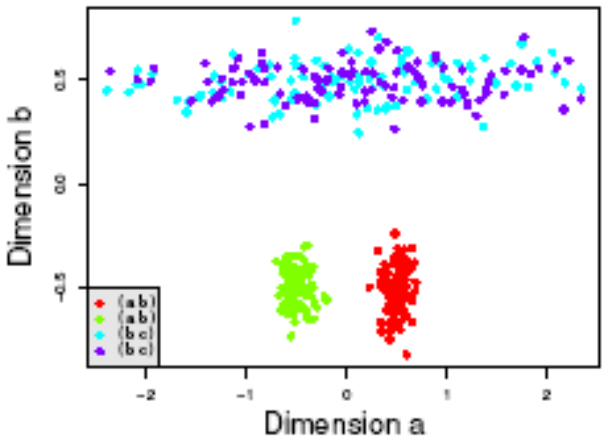
(a) Dimension a



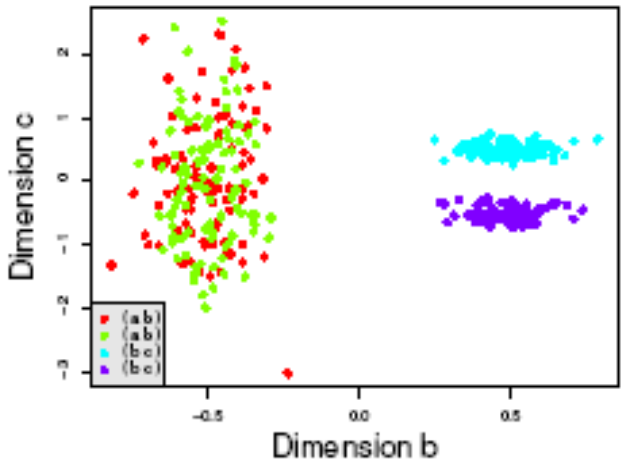
(b) Dimension b



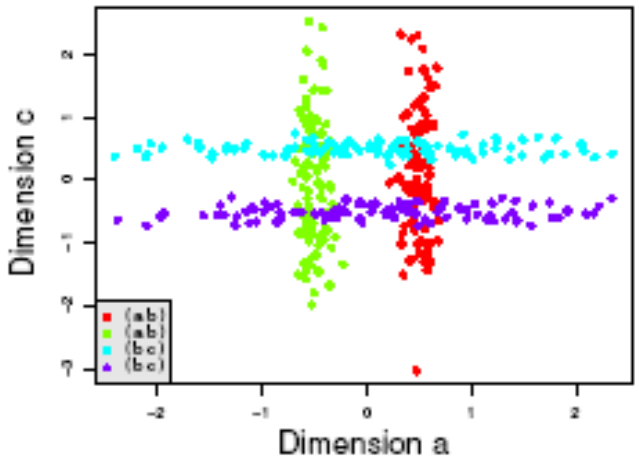
(c) Dimension c



(a) Dims a & b



(b) Dims b & c



(c) Dims a & c

Subspace Clustering Methods

- Subspace search methods
  - bottom-up approaches
  - top-down approaches
- Correlation-based clustering methods
  - PCA based methods
- Bi-clustering methods
  - optimization-based methods
  - enumeration methods

Subspace Clustering I: Subspace Search

- Bottom-up approaches
  - start from a low-D subspace and search higher-D subspaces only when there may be clusters in such subspaces
  - various pruning techniques to reduce the number of higher-D subspaces to be searched
  - Ex. CLIQUE
- Top-down approaches
  - start from full space and search smaller subspaces recursively
  - effective only if the locality assumption holds: subspace of a cluster can be determined by the local neighborhood
  - Ex. PROCLUS (Agrawal, et al. '99)

Subspace Clustering II: Correlation-based

- subspace search method: similarity based on distance or density
- correlation-based method: based on advanced correlation models
- Ex. PCA-based approach
  - Apply PCA (principal component analysis) to derive set of new, uncorrelated dimensions
  - Then, perform clusters in new spaces
- Other space transformations
  - Hough transform
  - Fractal dimensions

Subspace Clustering III: Bi-Clustering

- Bi-clustering: cluster both objects and attributes simultaneously (treat objects and attributes in symmetric way)
- Four requirements:
  - only a small set of objects participate in a cluster
  - a cluster only involves a small number of attributes
  - an object may participate in multiple clusters, or does not participate in any cluster at all
  - an attribute may be involved in multiple clusters, or is not involved in any cluster at all

Types of Bi-clusters

- Let A = {a_1, ..., a_n} be a set of genes, B = {b_1, …, b_n} a set of conditions
- A bi-cluster: A submatrix where genes and conditions follow some consistent patterns
- 4 types of bi-clusters (ideal cases)
  - Bi-clusters with constant values:
    - for any *i* in *I* and *j in J*, $e_{ij} = c$

10	10	10	10	10
20	20	20	20	20
50	50	50	50	50
0	0	0	0	0

  - Bi-clusters with constant values on rows:
    - $e_{ij} = c + \alpha_i$
    - Also, it can be constant values on columns

10	50	30	70	20
20	60	40	80	30
50	90	70	110	60
0	40	20	60	10

  - Bi-clusters with *coherent values* (aka. *pattern-based clusters*)
    - $e_{ij} = c + \alpha_i + \beta_j$
  - Bi-clusters with *coherent* evolutions on rows
    - $e_{ij}\ (e_{i1j1} - e_{i1j2})(e_{i2j1} - e_{i2j2}) \geq 0$
    - i.e., only interested in the up- or down- regulated changes across genes or conditions without constraining on the exact values

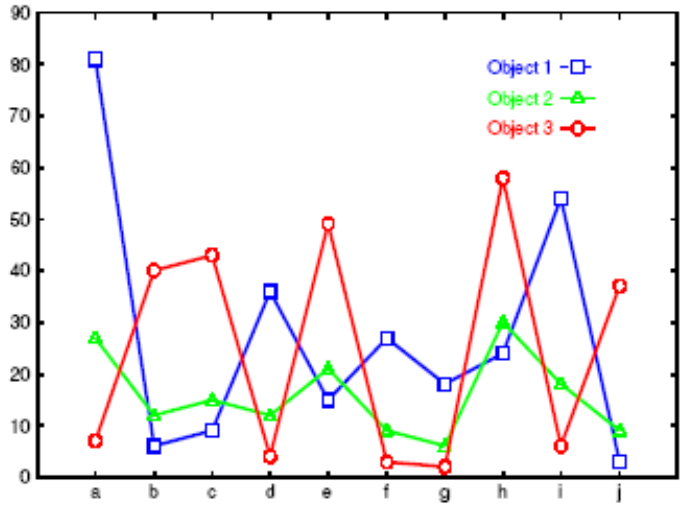
10	50	30	70	20
20	100	50	1000	30
50	100	90	120	80
0	80	20	100	10

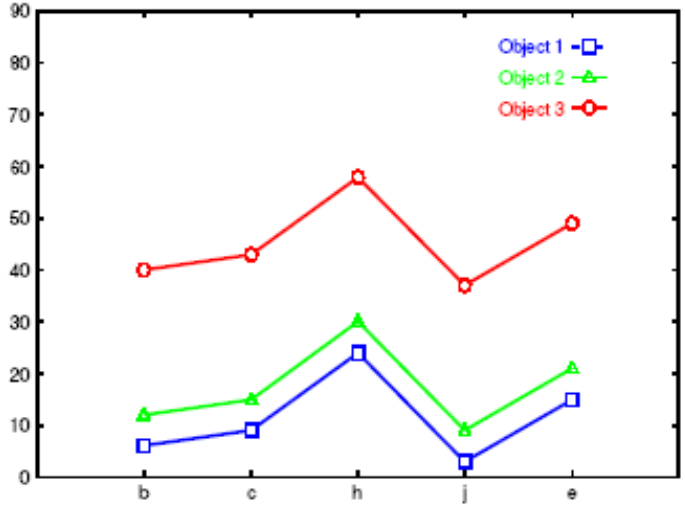
Bi-Clustering Methods

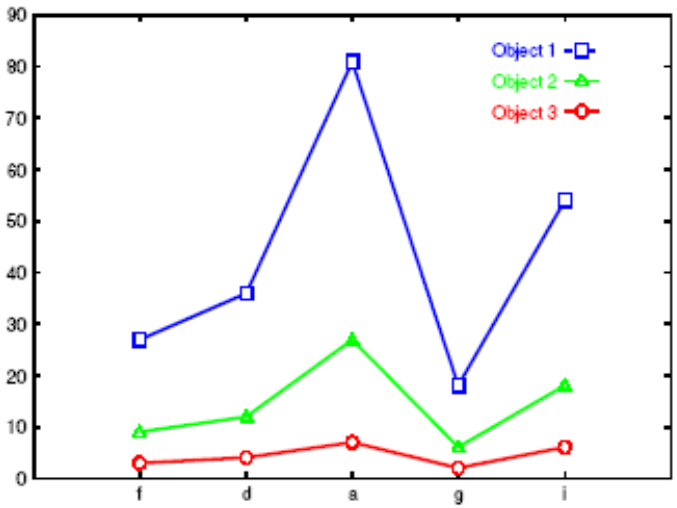
- Real-world data is noisy: Try to find approximate bi-clusters
- Methods: Optimization-based methods vs. enumeration methods
- Optimization-based methods
  - Try to find a submatrix at a time that achieves the best significance as a bi-cluster
  - Due to the cost in computation, greedy search is employed to find local optimal bi-clusters
  - Ex. δ-Cluster Algorithm (Cheng and Church, ISMB'2000)
- Enumeration methods
  - Use a tolerance threshold to specify the degree of noise allowed in the bi-clusters to be mined
  - Then try to enumerate all submatrices as bi-clusters that satisfy the requirements
  - Ex. δ-pCluster Algorithm (H. Wang et al.' SIGMOD'2002, MaPle: Pei et al., ICDM'2003)

Bi-Clustering for Microarray Analysis

- Left figure: Micro-array “raw” data shows 3 genes and their values in a multi-D space: Difficult to find their patterns
- Right two: Some subsets of dimensions form nice shift and scaling patterns
- No globally defined similarity/distance measure
- Clusters may not be exclusive
  - An object can appear in multiple clusters







Bi-Clustering (I): δ-Bi-Cluster

- For a submatrix *I x J,* the mean of the *i*-th row: $e_{iJ} = \frac{1}{|J|}\sum_{j \in J} e_{ij}$
  - The mean of the *j*-th column: $e_{Ij} = \frac{1}{|I|}\sum_{i \in I} e_{ij}$
  - The mean of all elements in the submatrix is

$$e_{IJ} = \frac{1}{|I||J|}\sum_{i \in I, j \in J} e_{ij} = \frac{1}{|I|}\sum_{i \in I} e_{iJ} = \frac{1}{|J|}\sum_{j \in J} e_{Ij}$$

  - The quality of the submatrix as a bi-cluster can be measured by the *mean squared residue* value

$$H(I \times J) = \frac{1}{|I||J|}\sum_{i \in I, j \in J} (e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2$$

- A submatrix *I x J* is **δ-bi-cluster** if $H(I \times J) \leq \delta$ where $\delta \geq 0$ is a threshold. When $\delta = 0$, *I x J* is a perfect bi-cluster with coherent values. By setting $\delta > 0$, a user can specify the tolerance of average noise per element against a perfect bi-cluster
  - residue(e_{ij}) = $e_{ij} - e_{iJ} - e_{Ij} + e_{IJ}$

Bi-Clustering (I): δ-Bi-Cluster

- **Maximal δ-bi-cluster** is a **δ**-bi-cluster *I x J* such that there does not exist another **δ**-bi-cluster *I' x J' which* contains *I x J*
- Computing is costly: Use heuristic greedy search to obtain local optimal clusters
- Two phase computation: deletion phase and additional phase
- Deletion phase: Start from the whole matrix, iteratively remove rows and columns while the mean squared residue of the matrix is over **δ**
  - At each iteration, for each row/*column*, compute the *mean squared residue:*

$$d(i) = \frac{1}{|J|}\sum_{j \in J}(e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2 \qquad d(j) = \frac{1}{|I|}\sum_{i \in I}(e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2$$

  - Remove the row or column of the largest mean squared residue
- Addition phase:
  - Expand iteratively the **δ**-bi-cluster *I x J* obtained in the deletion phase as long as the **δ**-bi-cluster requirement is maintained
  - Consider all the rows/columns not involved in the current bi-cluster *I x J* by calculating their mean squared residues
  - A row/column of the smallest mean squared residue is added into the current **δ**-bi-cluster
- It finds only one **δ**-bi-cluster, thus needs to run multiple times: replacing the elements in the output bi-cluster by random numbers

Bi-Clustering (II): δ-pCluster

- Enumerating all bi-clusters (δ-pClusters) [H. Wang, et al., Clustering by pattern similarity in large data sets. SIGMOD' 02]
- Since a submatrix *I x J* is a bi-cluster with (perfect) coherent values iff $e_{i1j1} - e_{i2j1} = e_{i1j2} - e_{i2j2}$. For any 2 *x* 2 submatrix of *I x J*, define *p-score*

$$p\text{-score}\begin{pmatrix} e_{i_1j_1} & e_{i_1j_2} \\ e_{i_2j_1} & e_{i_2j_2} \end{pmatrix} = |(e_{i_1j_1} - e_{i_2j_1}) - (e_{i_1j_2} - e_{i_2j_2})|$$

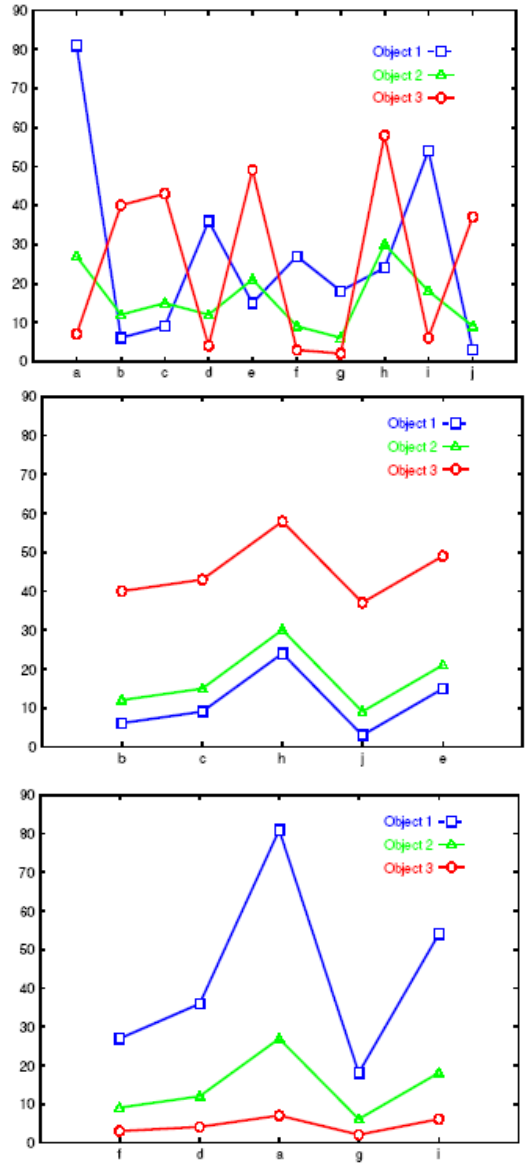
- A submatrix *I x J* is a **δ-pCluster** (pattern-based cluster) if the *p*-score of every 2 *x* 2 submatrix of *I x J* is at most δ, where δ ≥ 0 is a threshold specifying a user's tolerance of noise against a perfect bi-cluster
- The *p*-score controls the noise on every element in a bi-cluster, while the mean squared residue captures the average noise
- **Monotonicity**: If *I* x *J* is a δ-pClusters, every *x* x *y* (*x,y* ≥ 2) submatrix of *I* x *J* is also a δ-pClusters.
- A δ-pCluster is **maximal** if no more row or column can be added into the cluster and retain δ-pCluster: We only need to compute all maximal δ-pClusters.

MaPle: Enumeration of δ-pClusters

- Pei et al., MaPle: Efficient enumerating all maximal δ-pClusters. ICDM'03
- Framework: Same as pattern-growth in frequent pattern mining (based on the downward closure property)
- For each condition combination J, find the maximal subsets of genes I such that I x J is a δ-pClusters
  - If I x J is not a submatrix of another δ-pClusters
  - then I x J is a maximal δ-pCluster.
- Algorithm is very similar to mining frequent closed itemsets
- Additional advantages of δ-pClusters:
  - Due to averaging of δ-cluster, it may contain outliers but still within δ-threshold
  - Computing bi-clusters for scaling patterns, take logarithmic on

$$\frac{d_{xa} / d_{ya}}{d_{xb} / d_{yb}} < \delta$$

will lead to the p-score form



Dimensionality Reduction Methods

- Dimensionality reduction: In some situations, it is more effective to construct a new space instead of using some subspaces of the original data
- Dimensionality reduction methods
  - Feature selection and extraction: But may not focus on clustering structure finding
  - Spectral clustering: Combining feature extraction and clustering (i.e., use the *spectrum* of the similarity matrix of the data to perform dimensionality reduction for clustering in fewer dimensions)
    - Normalized Cuts (Shi and Malik, CVPR'97 or PAMI'2000)
    - The Ng-Jordan-Weiss algorithm (NIPS'01)

Spectral Clustering: Ng-Jordan-Weiss (NJW) Method

- Given a set of objects $o_1, \ldots, o_n$, and the distance between each pair of objects, $dist(o_i, o_j)$, find the desired number k of clusters
- Calculate an affinity matrix W, where σ is a scaling parameter that controls how fast the affinity W_{ij} decreases as $dist(o_i, o_j)$ increases. In NJW, set $W_{ij} = 0$

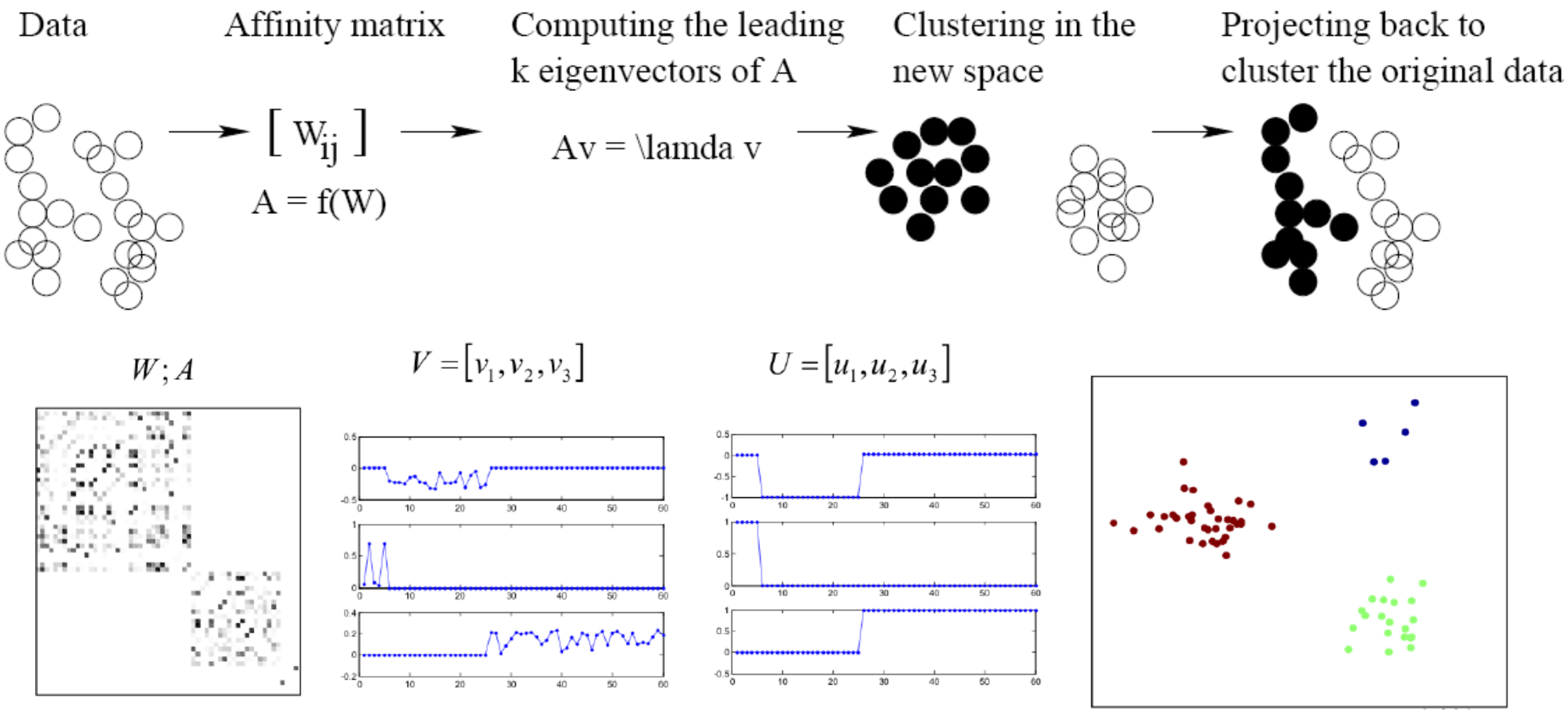
$$W_{ij} = e^{-\frac{dist(o_i,o_j)}{\sigma^2}}$$

- Derive a matrix A = *f*(W). NJW defines a matrix D to be a diagonal matrix s.t. D_{ii} is the sum of the i-th row of W, i.e.,

$$D_{ii} = \sum_{j=1}^{n} W_{ij}$$

  Then, A is set to $A = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$
- A spectral clustering method finds the k leading eigenvectors of A
  - A vector v is an eigenvector of matrix A if Av = λv, where λ is the corresponding eigen-value
- Using the k leading eigenvectors, project the original data into the new space defined by the k leading eigenvectors, and run a clustering algorithm, such as *k*-means, to find k clusters
- Assign the original data points to clusters according to how the transformed points are assigned in the clusters obtained

Spectral Clustering



- Spectral clustering: Effective in tasks like image processing
- Scalability challenge: Computing eigenvectors on a large matrix is costly
- Can be combined with other clustering methods, such as bi-clustering

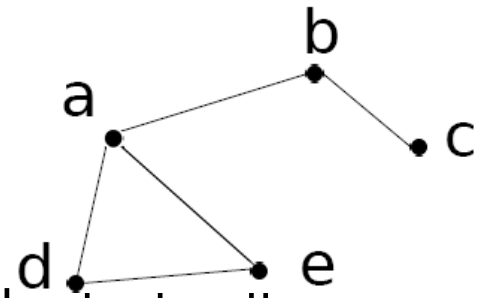
Outline

- Extensions to Hierarchical Clustering
  - BIRCH, CHAMELEON
- Density-based Clustering
  - DBSCAN, OPTICS, DENCLUE
- Grid-Based Clustering
  - STING, CLIQUE
- Probabilistic Clustering
  - EM Clustering
- Clustering High-Dimensional Data
  - Subspace Clustering, Spectral Clustering
- Clustering Graphs and Network Data

Clustering Graphs

- Applications
  - Bi-partite graphs, e.g., customers and products, authors and conferences
  - Web search engines, e.g., click through graphs and Web graphs
  - Social networks, friendship/coauthor graphs
- Similarity measures
  - Geodesic distances
  - Distance based on random walk (SimRank)
- Graph clustering methods
  - Minimum cuts: FastModularity (Clauset, Newman & Moore, 2004)
  - Density-based clustering: SCAN (Xu et al., KDD'2007)

Similarity Measure: Geodesic Dist.



- Geodesic distance (A, B): length (i.e., # of edges) of the shortest path between A and B (if not connected, defined as infinite)
- **Eccentricity** of v, eccen(v): The largest geodesic distance between v and any other vertex $u \in V - \{v\}$.
  - E.g., eccen(a) = eccen(b) = 2; eccen(c) = eccen(d) = eccen(e) = 3
- **Radius** of graph G: The minimum eccentricity of all vertices, i.e., the distance between the "most central point" and the "farthest border"
  - $r = \min_{v \in V} eccen(v)$
  - E.g., radius (g) = 2
- **Diameter** of graph G: The maximum eccentricity of all vertices, i.e., the largest distance between any pair of vertices in G
  - $d = \max_{v \in V} eccen(v)$
  - E.g., diameter (g) = 3
- A **peripheral vertex** is a vertex that achieves the diameter.
  - E.g., Vertices c, d, and e are peripheral vertices

SimRank

- Similarity based on Random Walk and Structural Context
- In a directed graph G = (V,E),
  - *individual in-neighborhood* of v: I(v) = {u | (u, v) ∈ E}
  - *individual out-neighborhood* of v: O(v) = {w | (v, w) ∈ E}
- Similarity in SimRank: $$s(u,v) = \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x,y)$$
  - Initialization: $$s_0(u,v) = \begin{cases} 0 & \text{if } u \neq v \\ 1 & \text{if } u = v \end{cases}$$
  - Then we can compute s_{i+1} from s_i based on the definition
- Similarity based on random walk: in a strongly connected component
  - Expected distance: $$d(u,v) = \sum_{t:u \rightsquigarrow v} P[t] l(t)$$
  - Expected meeting distance: $$m(u,v) = \sum_{t:(u,v) \rightsquigarrow (x,x)} P[t] l(t)$$
  - Expected meeting probability: $$p(u,v) = \sum_{t:(u,v) \rightsquigarrow (x,x)} P[t] C^{l(t)}$$

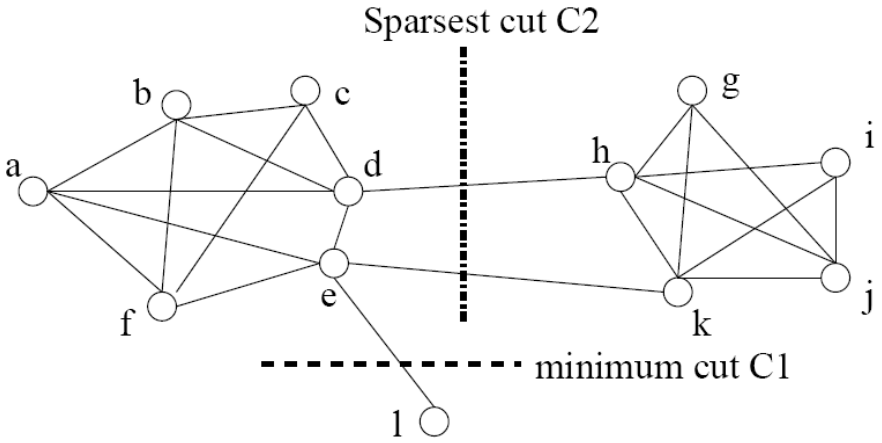
$$P[t] = \begin{cases} \prod_{i=1}^{k-1} \frac{1}{|O(w_i)|} & \text{if } l(t) > 0 \\ 0 & \text{if } l(t) = 0. \end{cases}$$

P[t] is the probability of the tour

Graph Clustering: Sparsest Cut

- G = (V,E). The *cut set* of a cut is the set of edges {(u, v) ∈ E | u ∈ S, v ∈ T } and S and T are in two partitions
- *Size* of the cut: # of edges in the cut set
- Min-cut (e.g., C_1) is not a good partition
- A better measure: **Sparsity**:

$$\Phi = \frac{\text{the size of the cut}}{\min\{|S|, |T|\}}$$



- A cut is ***sparsest*** if its sparsity is not greater than that of any other cut
- Ex. Cut C2 = ({a, b, c, d, e, f, l}, {g, h, i, j, k}) is the sparsest cut
- For k clusters, the **modularity** of a clustering assesses the quality of the clustering:

- The *modularity* of a clustering of a graph is the difference between the fraction of all edges that fall into individual clusters and the fraction that would do so if the graph vertices were randomly connected
- The optimal clustering of graphs maximizes the modularity

Graph Clustering: Find Good Cuts

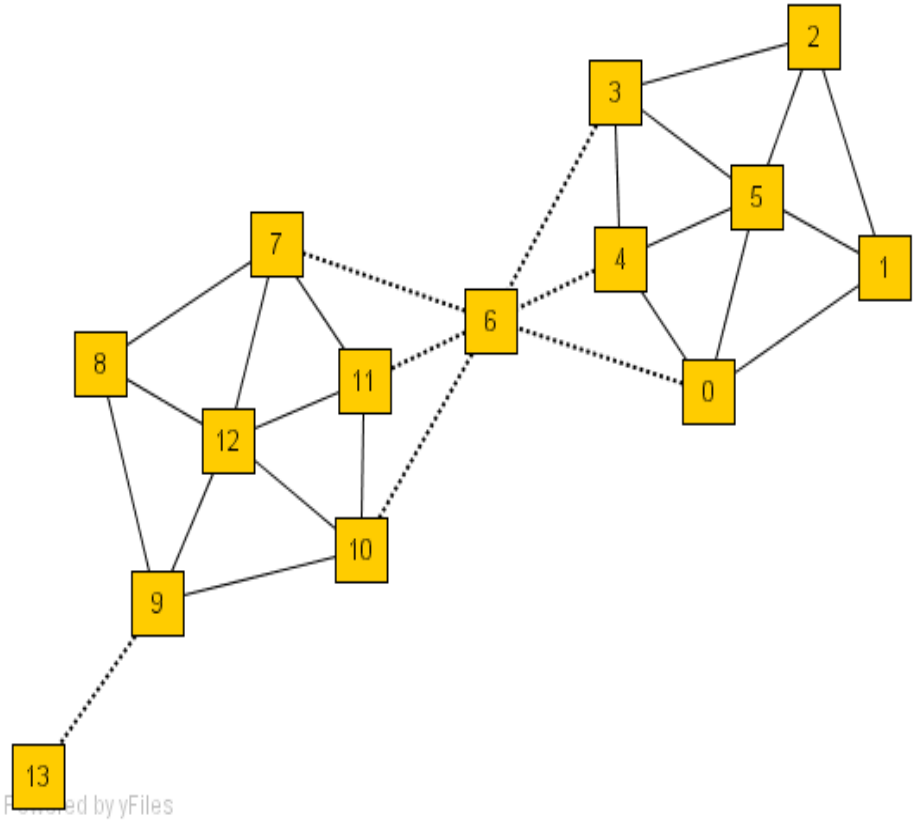
- High computational cost
  - Many graph cut problems are computationally expensive
  - The sparsest cut problem is NP-hard
  - Need to tradeoff between efficiency/scalability and quality
- Sophisticated graphs
  - May involve weights and/or cycles.
- High dimensionality
  - A graph can have many vertices. In a similarity matrix, a vertex is represented as a vector (a row in the matrix) whose dimensionality is the number of vertices in the graph
- Sparsity
  - A large graph is often sparse, meaning each vertex on average connects to only a small number of other vertices
  - A similarity matrix from a large sparse graph can also be sparse

Graph Clustering

- Two approaches for clustering graph data
  - Use *generic clustering methods* for high-dimensional data
  - *Designed specifically for clustering graphs*
- Using clustering methods for high-dimensional data
  - Extract a similarity matrix from a graph using a similarity measure
  - A generic clustering method can then be applied on the similarity matrix to discover clusters
  - Ex. Spectral clustering: approximate optimal graph cut solutions
- Methods specific to graphs
  - Search the graph to find well-connected components as clusters
  - Ex. SCAN (Structural Clustering Algorithm for Networks)
    - X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, “SCAN: A Structural Clustering Algorithm for Networks”, KDD'07

SCAN: density-based network clustering

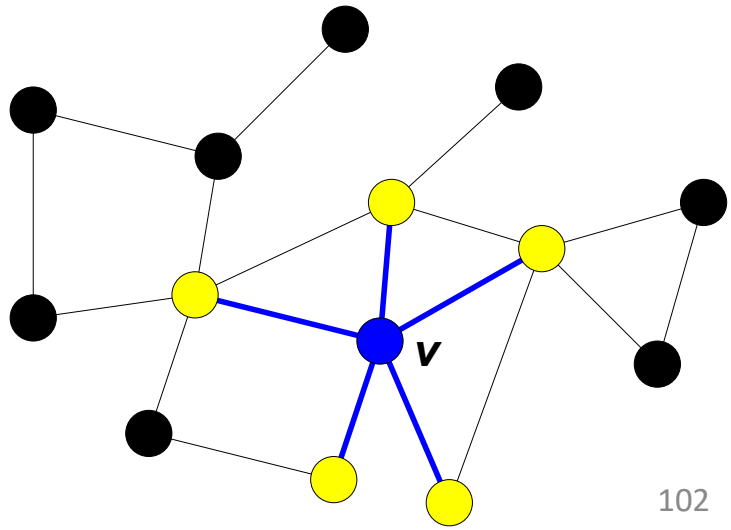
- How many clusters?
- What size should they be?
- What is the best partitioning?
- Should some points be segregated?



Application: Given information of who associates with whom, could one identify clusters of individuals with common interests, or special relationships (friends, family, etc.)

Social Network Model

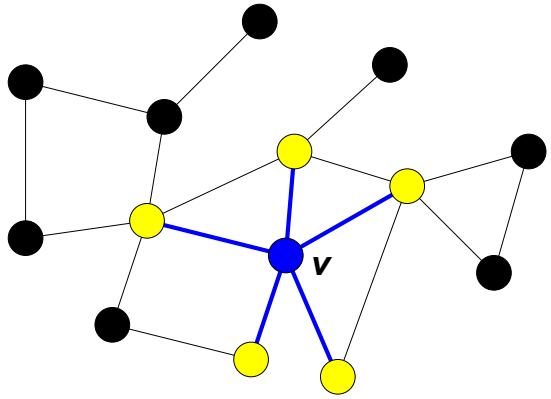
- Cliques, hubs and outliers
  - Individuals in a tight social group, or clique, know many of the same people, regardless of the size of the group
  - Individuals who are hubs know many people in different groups but belong to no single group. Politicians, for example bridge multiple groups
  - Individuals who are outliers reside at the margins of society. Hermits, for example, know few people and belong to no group
- The Neighborhood of a Vertex
  - immediate neighborhood of a vertex (the set of people that an individual knows)



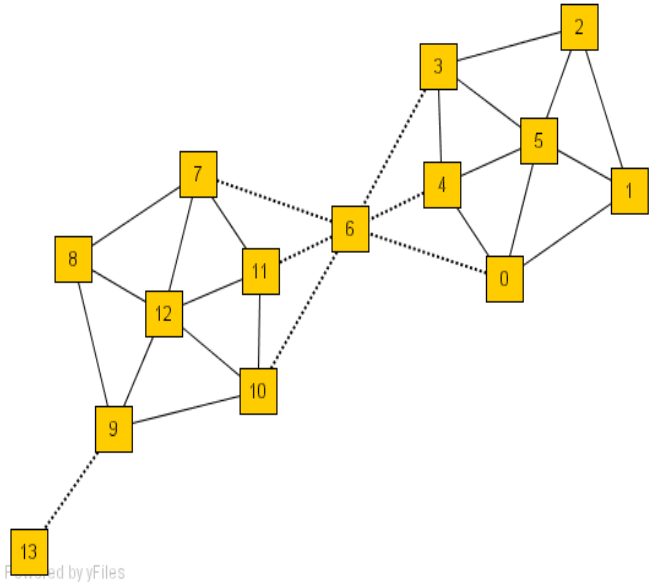
Structure Similarity

- The desired features tend to be captured by a measure we call Structural Similarity

$$\sigma(v,w) = \frac{|\Gamma(v) \bigcap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$



- Structural similarity is large for members of a clique and small for hubs and outliers



Structural Connectivity

- Neighborhood: $N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$
- Core: $CORE_{\varepsilon,\mu}(v) \Leftrightarrow \mid N_\varepsilon(v) \mid \geq \mu$
- Direct structure reachable:

$$DirRECH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_\varepsilon(v)$$

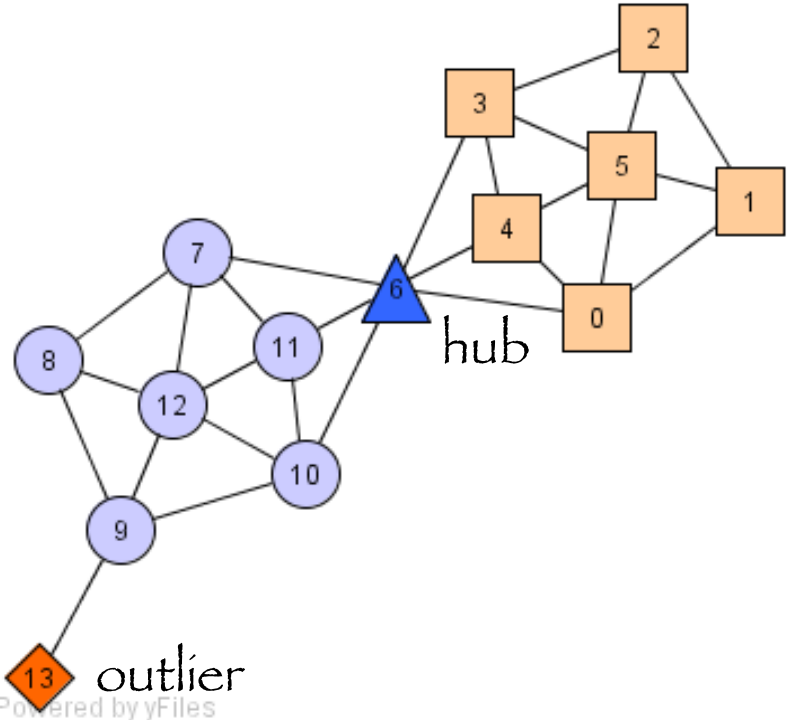
- Structure reachable: transitive closure of direct structure reachability
- Structure connected:

$$CONNECT_{\varepsilon,\mu}(v,w) \Leftrightarrow \exists u \in V : RECH_{\varepsilon,\mu}(u,v) \wedge RECH_{\varepsilon,\mu}(u,w)$$

[1] M. Ester, H. P. Kriegel, J. Sander, & X. Xu (KDD'96) "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases

Structure-Connected Clusters

- Structure-connected cluster C
  - Connectivity: $\forall v, w \in C : CONNECT_{\varepsilon,\mu}(v, w)$
  - Maximality: $\forall v, w \in V : v \in C \wedge REACH_{\varepsilon,\mu}(v, w) \Rightarrow w \in C$
- Hubs:
  - Not belong to any cluster
  - Bridge to many clusters
- Outliers:
  - Not belong to any cluster
  - Connect to less clusters



Summary of Clustering Methods

- Clustering: group data so that object similarity is high within clusters but low across clusters
- Partitioning Methods
  - k-means, k-medoids
- Hierarchical Methods
  - Agglomerative and divisive methods, Birch, Chameleon
- Density-based Methods
  - DBScan, Optics, and DenClue
- Grid-Based Methods
  - STING and CLIQUE

Summary of Clustering Methods

- Probability Model-Based Clustering
  - Probability-model-based clustering
  - The EM algorithm
- Clustering High-Dimensional Data
  - Subspace clustering: bi-clustering methods
  - Dimensionality reduction: Spectral clustering
- Clustering Graphs and Network Data
  - Graph clustering: min-cut vs. sparsest cut
  - High-dimensional clustering methods
  - Graph-specific clustering methods, e.g., SCAN