Clustering/Segmentation

Read: JW Ch 9, 10

(also HTF 14.3, B 9)

Data Representation and Goal

- Goal: Partition objects into a (possibly unspecified) number of groups or clusters
  - Without access to "Y"
- Want objects in same group to be similar; and objects in different groups to be quite different

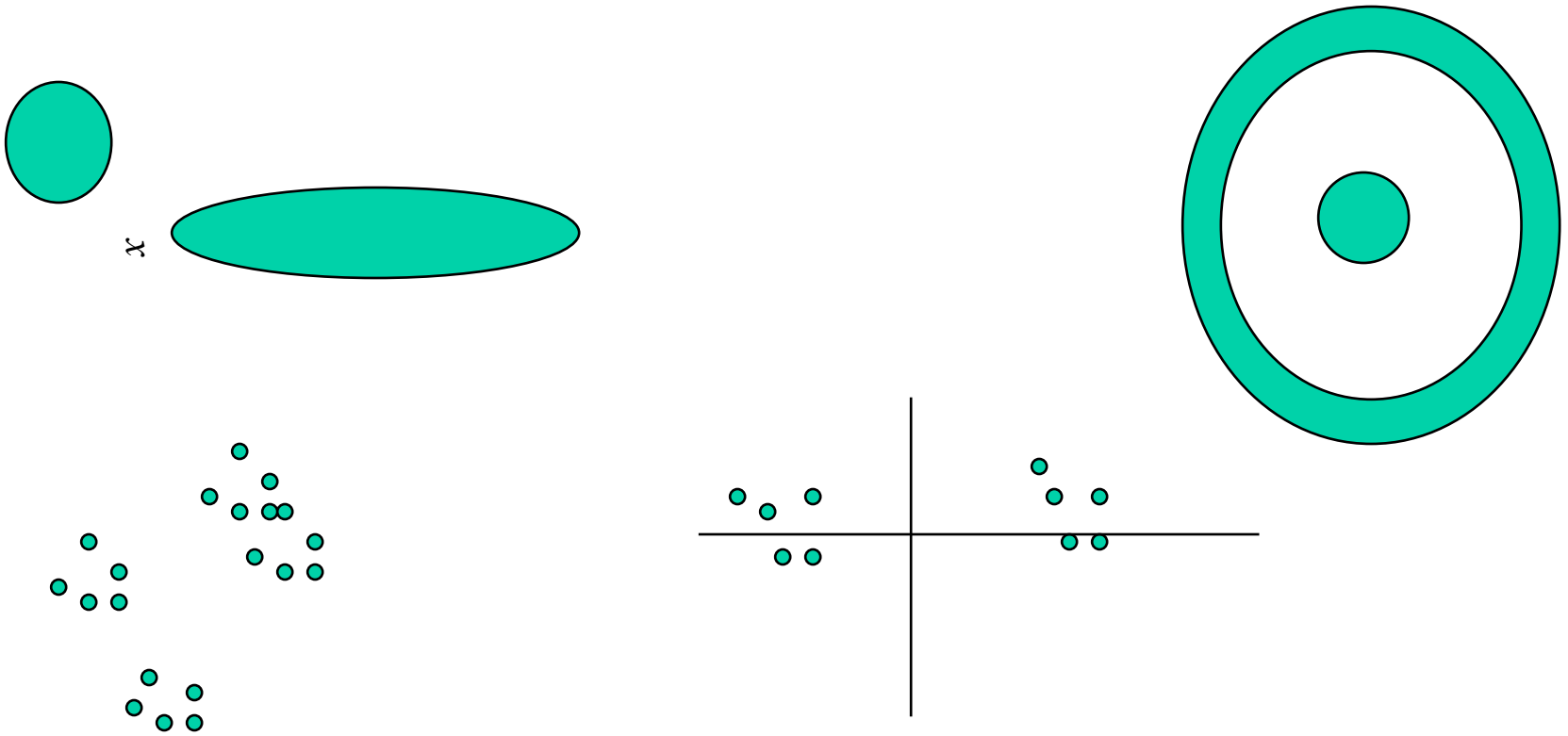
Object Representation:

- Objects are points in some feature space
- A similarity measure is known for each pair of objects

Can convert between representations.

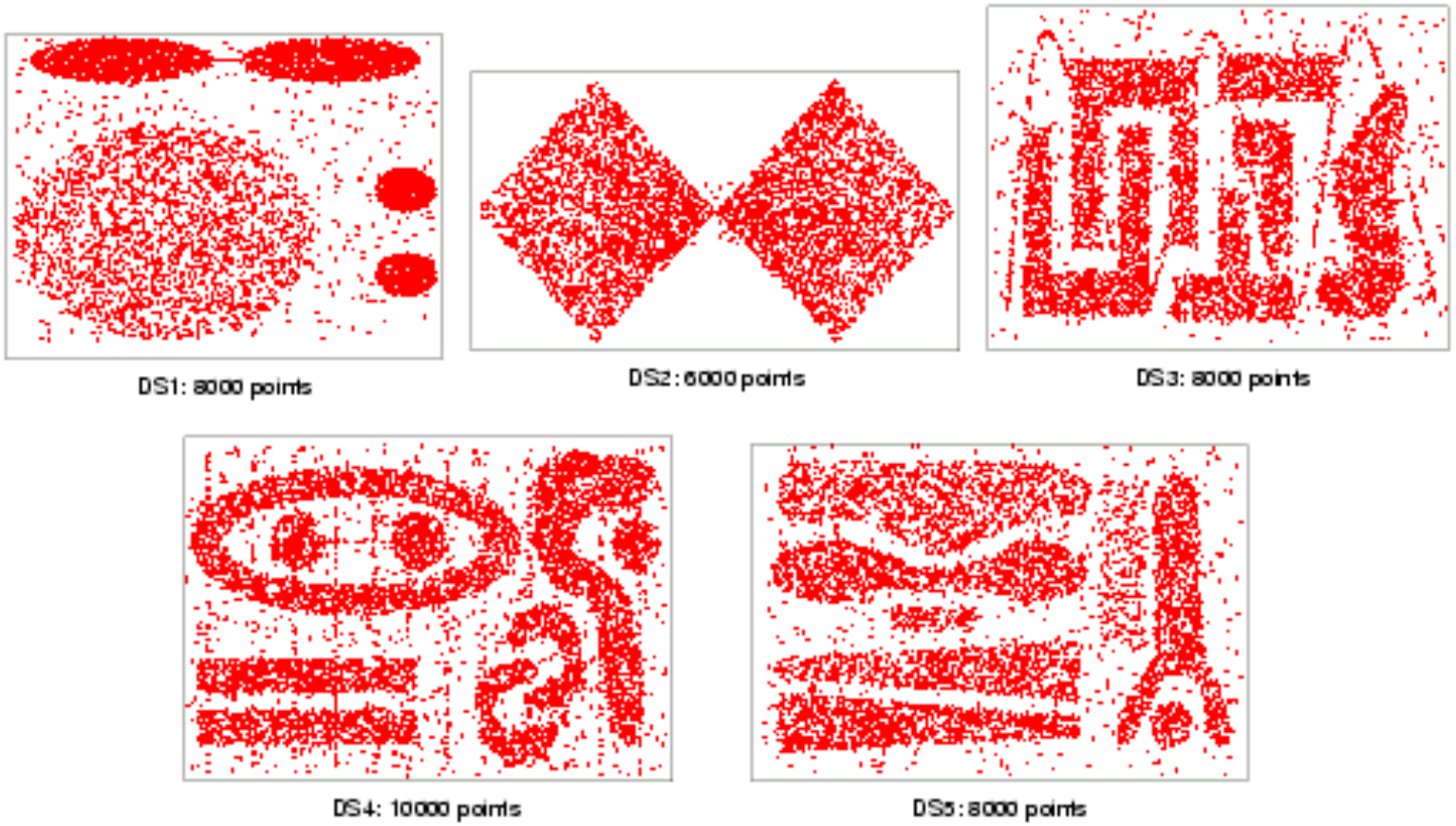
Difficulty of Problem

- Difficult even for 2-D data in Euclidean space!



Various Shapes

- Different sizes, shapes, non-convex, noisy,…..



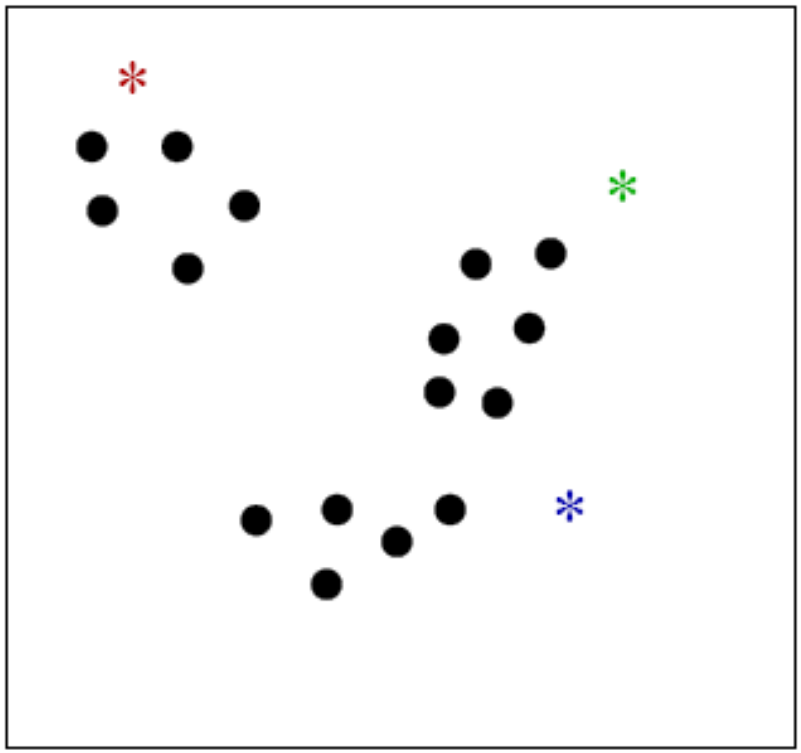
Partitional Clustering: K-Means Algo

- K = # of clusters (given); one "mean" per cluster
- Interval data

- Initialize means (e.g. by picking k samples at random)
- Iterate:

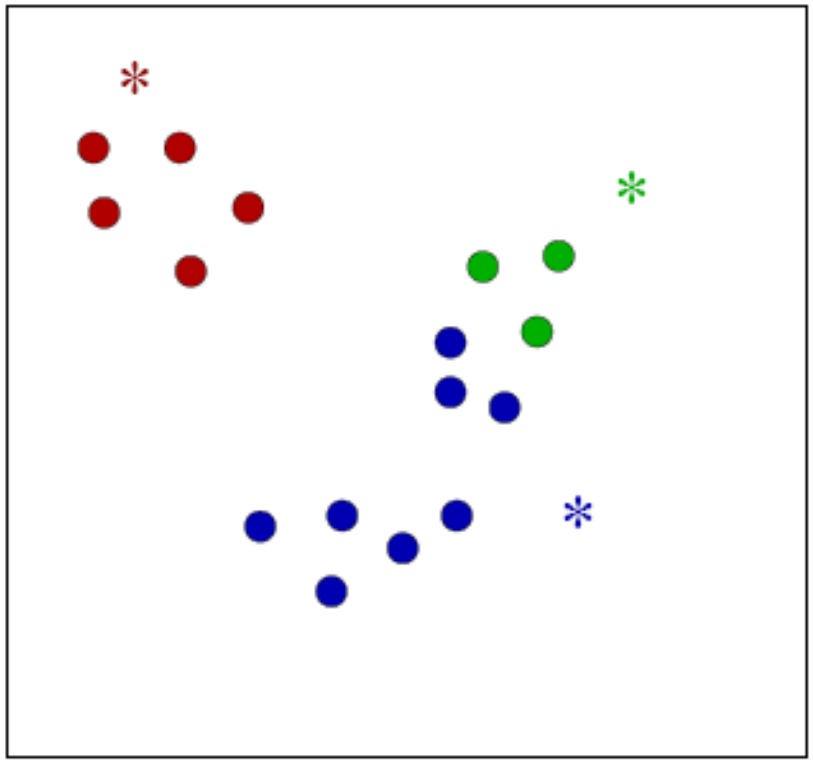
(1) assign each point to nearest mean

(2) move "mean" to center of its cluster.

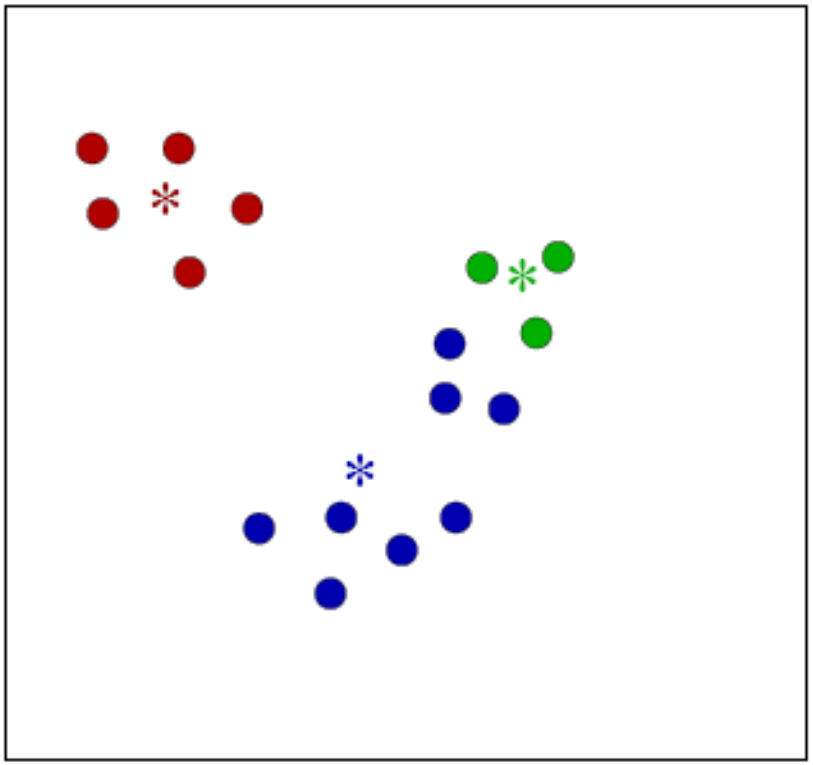


Initialize representatives ("means")

Assignment Step; Means Update



Assign to nearest representative



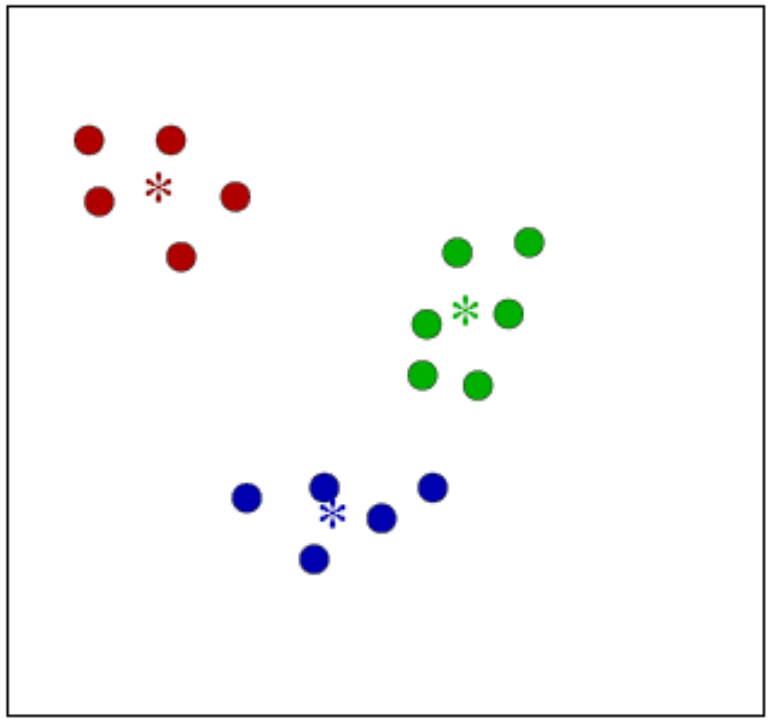
Re-estimate means

Visualization http://tech.nitoyon.com/en/blog/2013/11/07/k-means/

Convergence after another iteration

Complexity:

O(k . n . # of iterations)



The objective function is

$$\min_{\{\mu_1,\cdots,\mu_k\}} \sum_{h=1} \sum_{\mathbf{x}\in\mathcal{X}_h} \|\mathbf{x}-\boldsymbol{\mu}_h\|^2$$

Hierarchical Agglomerative Clustering (HAC)

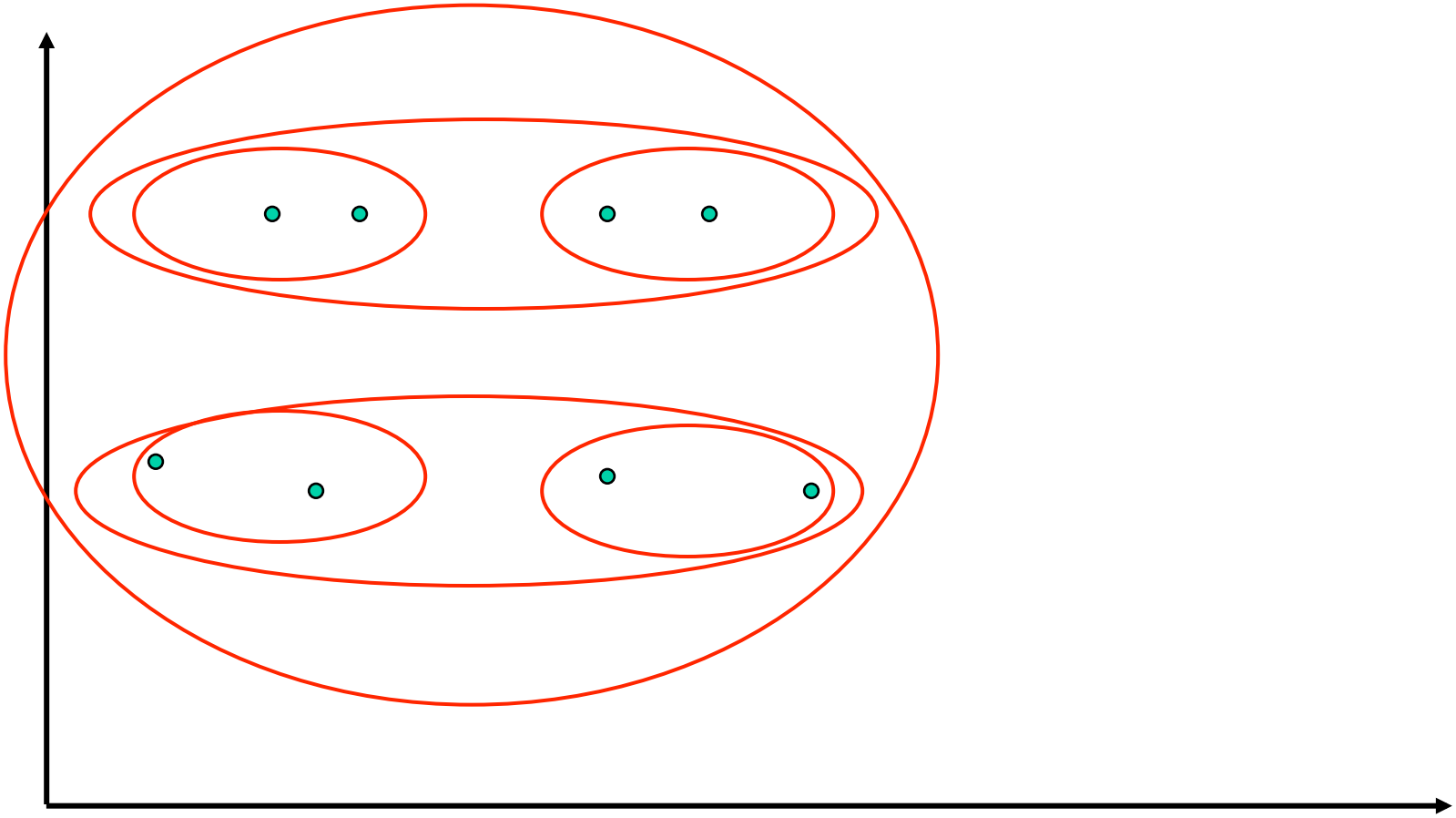
- Start with all instances in their own cluster.
- Until there is only one cluster:
  - Among the current clusters, determine the two
  - clusters, c_i and c_j, that are most similar
  - Replace c_i and c_j with a single cluster $c_i \cup c_j$

- Computational Complexity:O(n^2 log n)

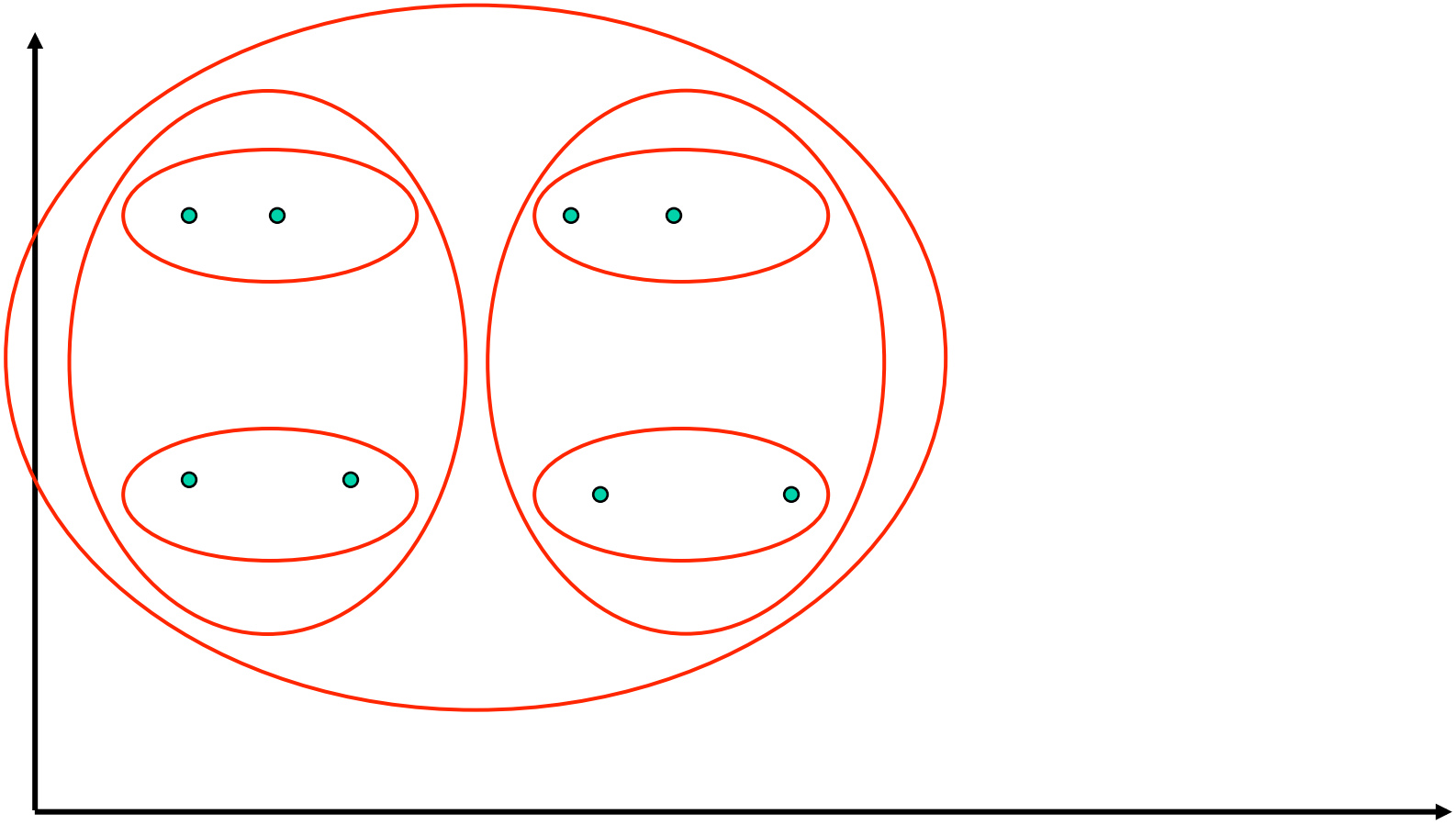
HAC choices

- How to compute similarity of two clusters each possibly containing multiple instances?
  - Single Link (aka nearest neighbor): Similarity of two most similar members.
    - SLINK: online version (Sibson, 73)
  - Complete Link (aka farthest neighbor): Similarity of two least similar members.
  - Group Average: Average similarity between members.
    - Can still compute cosine similarity in constant time if all docs unit length, and sum of docs in cluster is stored!!
  - Ward's Method:
    - minimize increase in variance = $m_i\, m_j\, d^2_{ij} / (m_i + m_j)$
      - m = # of objects
      - Tends to produce homogeneous clusters, symmetric hierarchy
      - But sensitive to outliers/ elongated clusters

Single Link Example

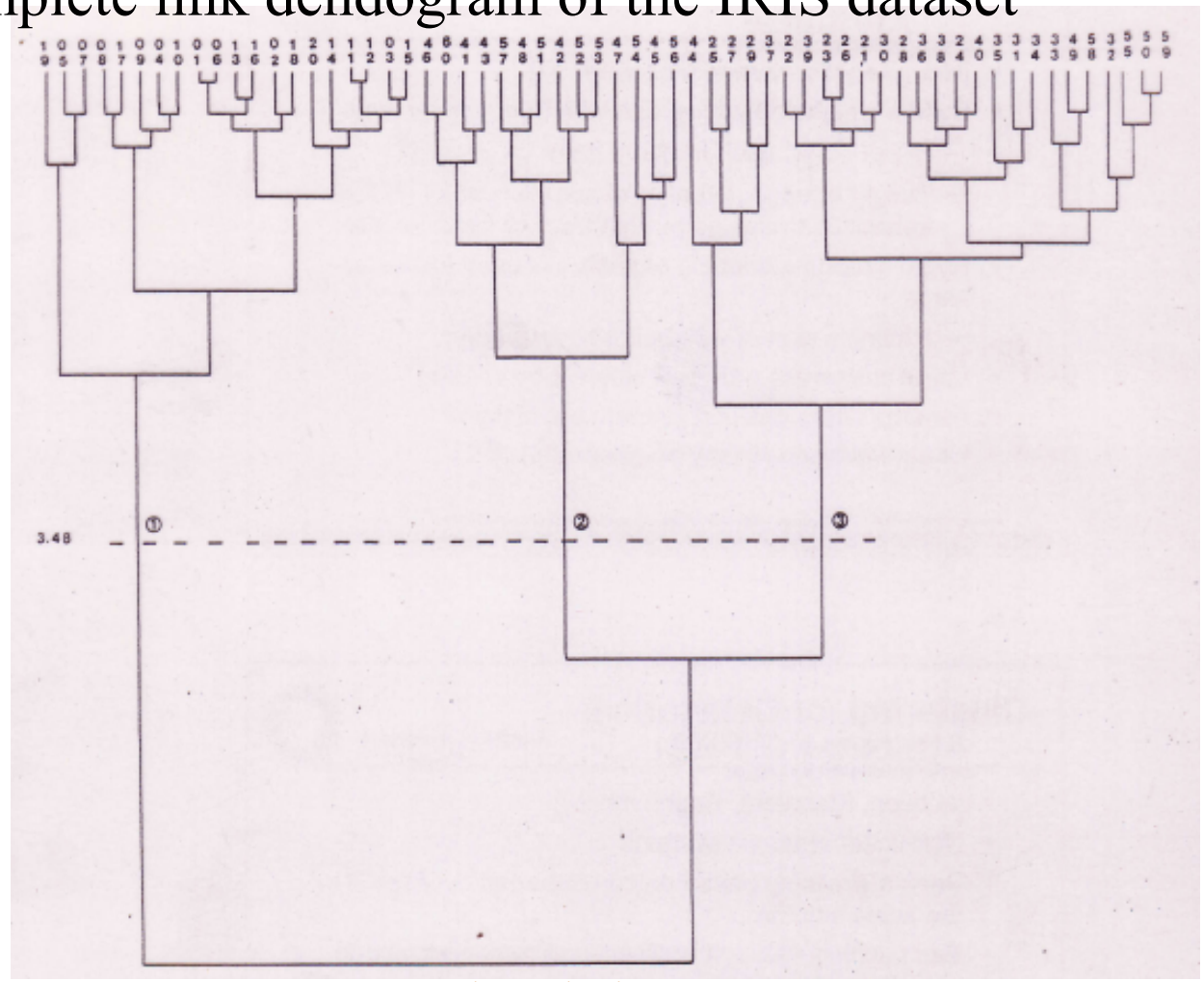


Complete Link Example



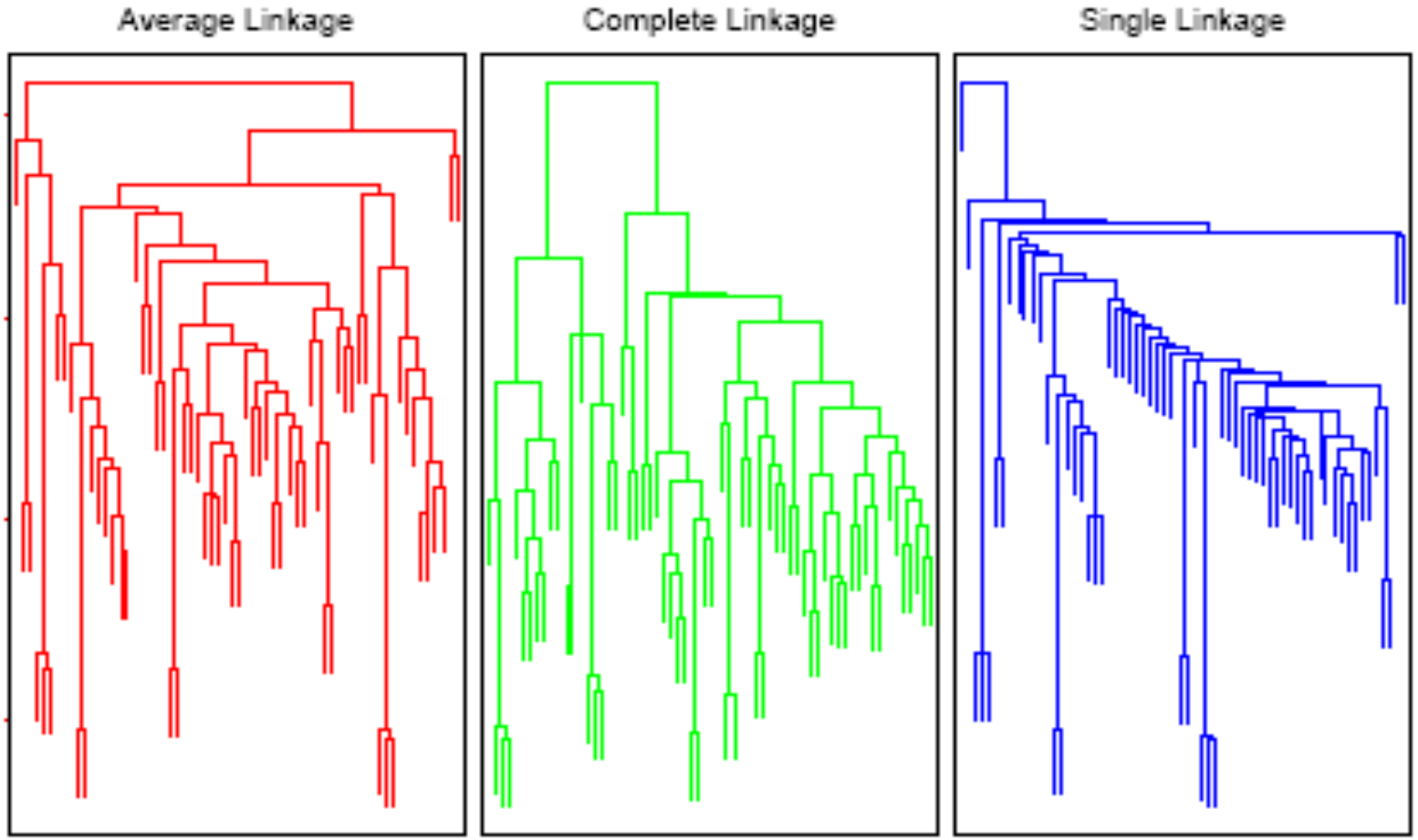
Dendograms and Model Selection

- Complete link dendogram of the IRIS dataset



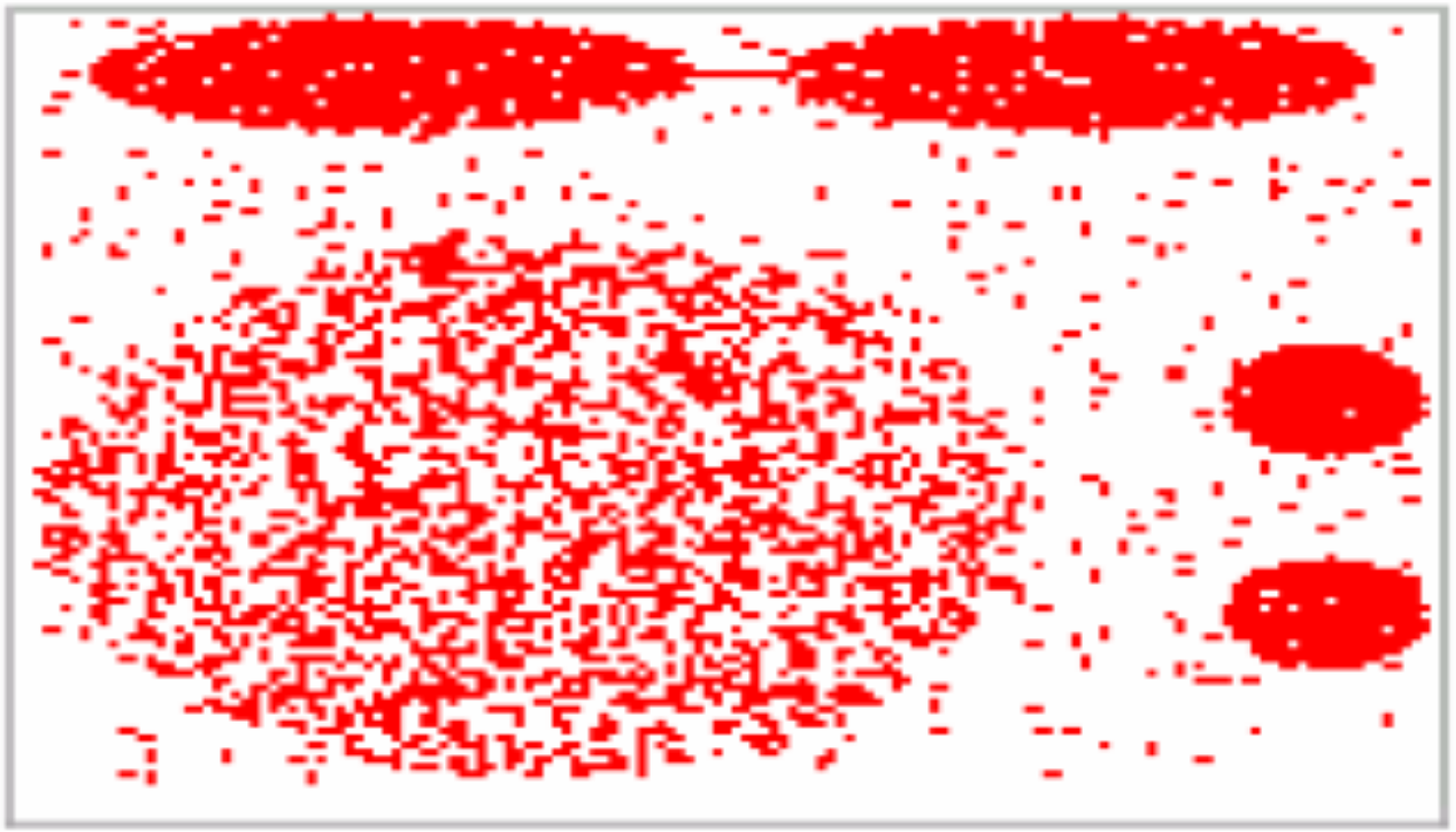
Visualizing Hierarchical Clustering: Dendograms

- Not a graphical summary but a clustering structure *imposed* by a specific algorithm



Points of Failure

- Where will (i) k-means, (ii) single-link, (iii) complete-link FAIL?



Soft Clustering Techniques

- In soft clustering techniques, a data point can belong (with different amounts of membership level) to more than one cluster.
  - Fuzzy k-means
  - Fitting a Mixture Model ("EM clustering")
    - Each component of mixture is a unimodel distribution, (e.g. Gaussian) corresponding to a cluster
    - Use a suitable method such as "EM" to fit the mixture model to the data
      - Yields membership of each point to each mixture component
      - Very general, e.g. mixture of Hidden Markov Models, etc.

Expectation-Maximization (EM)

- Dempster, Laird & Rubin ’77
- Maximize likelihood given incomplete data
  - Missing “indicator” (membership) variables, which if known, would have simplified solution
    - Mixture density example
    - See ICML2001 keynote talk at http://www.ics.uci.edu/~smyth/talks/
  - Technique:
    - estimate (expected value of) missing variables;
    - Solve for model parameters;
    - re-estimate and iterate
  - Applications:
    - Mixture density functions (➔ soft k-means)
    - Mixture-of-experts for function approximation
    - Baum-Welch algorithm for HMMs
    - Motion segmentation
    - ……
- Ref: Blimes J.A., **A Gentle Tutorial of the EM algorithm ….** 1998

Fitting Mixture Models example (adapted from ICML 2001 Talk by Padhraic Smyth)

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}, c_k)$$

$$= \sum_{k=1}^{K} p(\mathbf{x} \mid c_k) p(c_k)$$

$$= \sum_{k=1}^{K} p(\mathbf{x} \mid c_k, \theta_k) \alpha_k$$

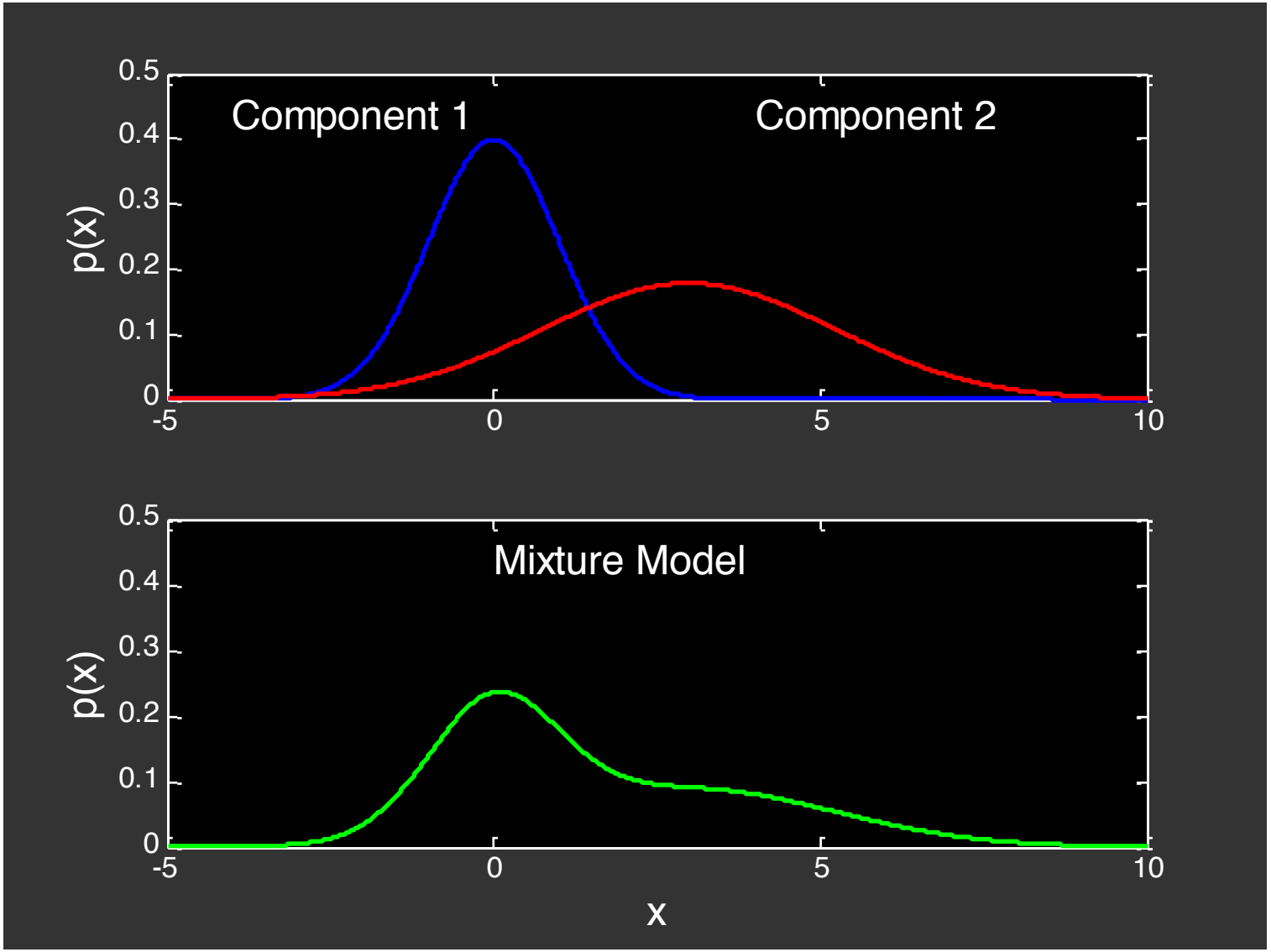
Weight$_k$

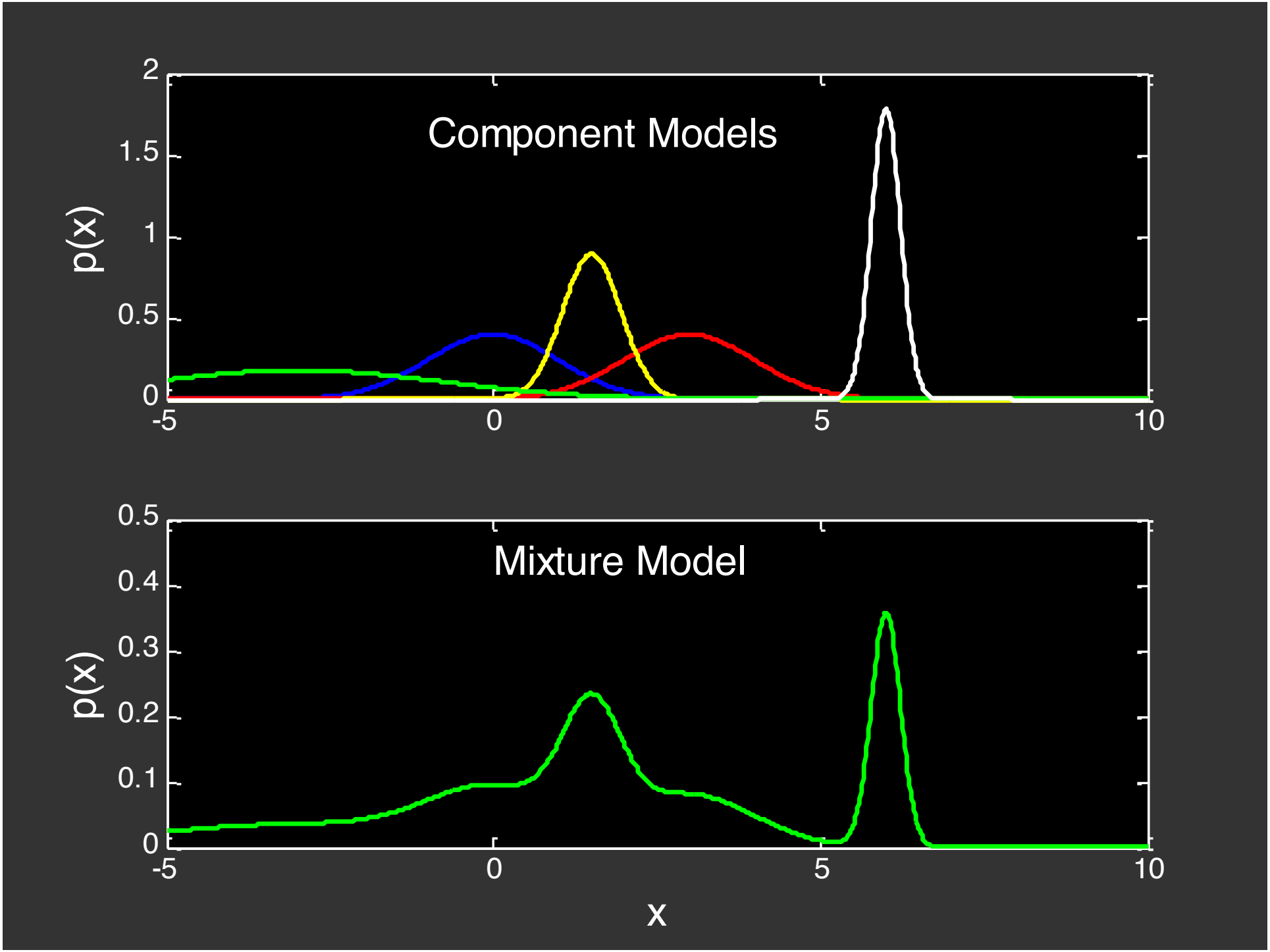
Component Model$_k$

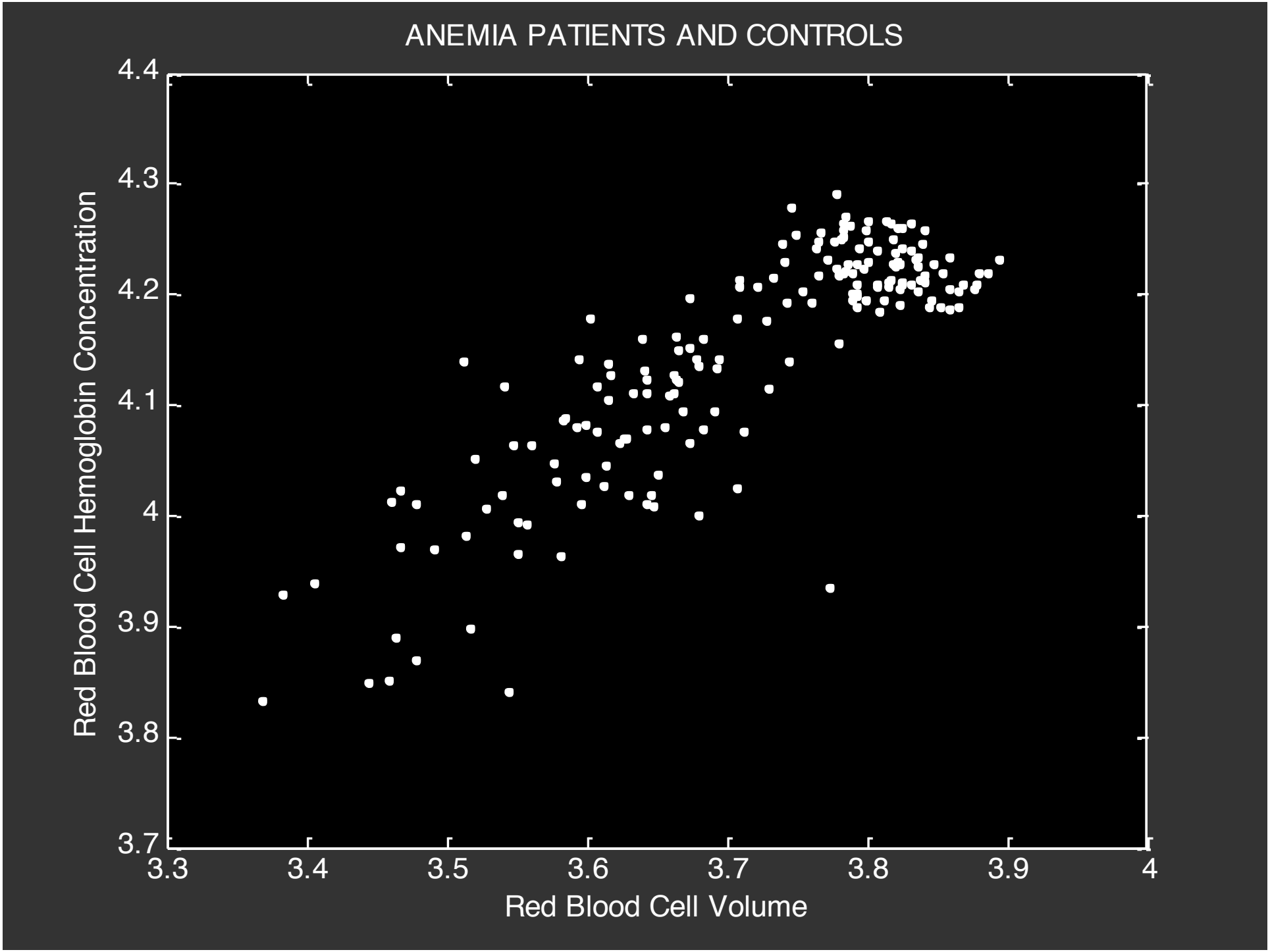
parameters

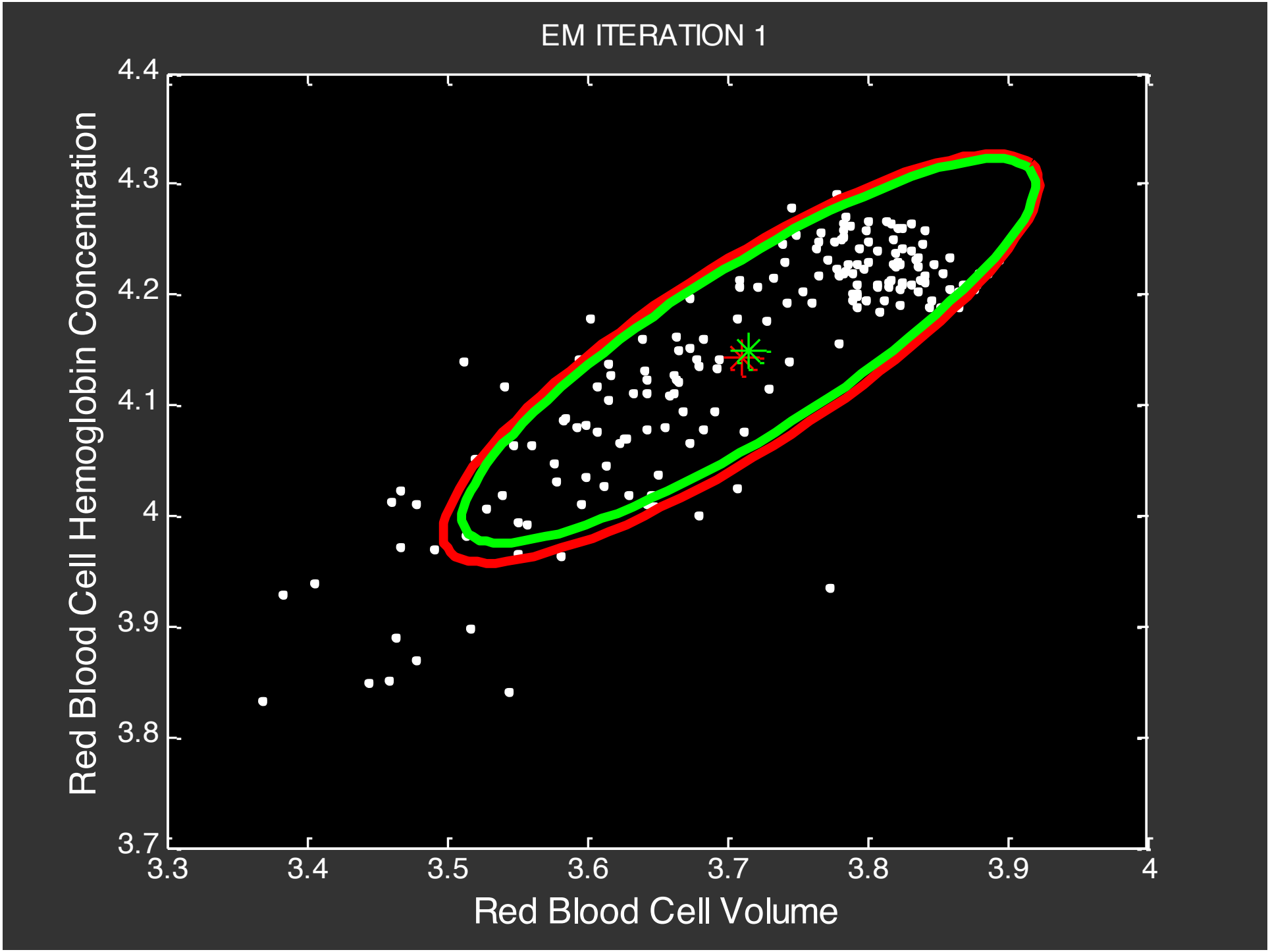
e.g. Mixture of two 1-Dimensional Gaussians:
Fit parameters $\{\theta, \alpha\} = \{\mu 1 , \sigma 1 , \mu 2 , \sigma 2 , \alpha 1\}$

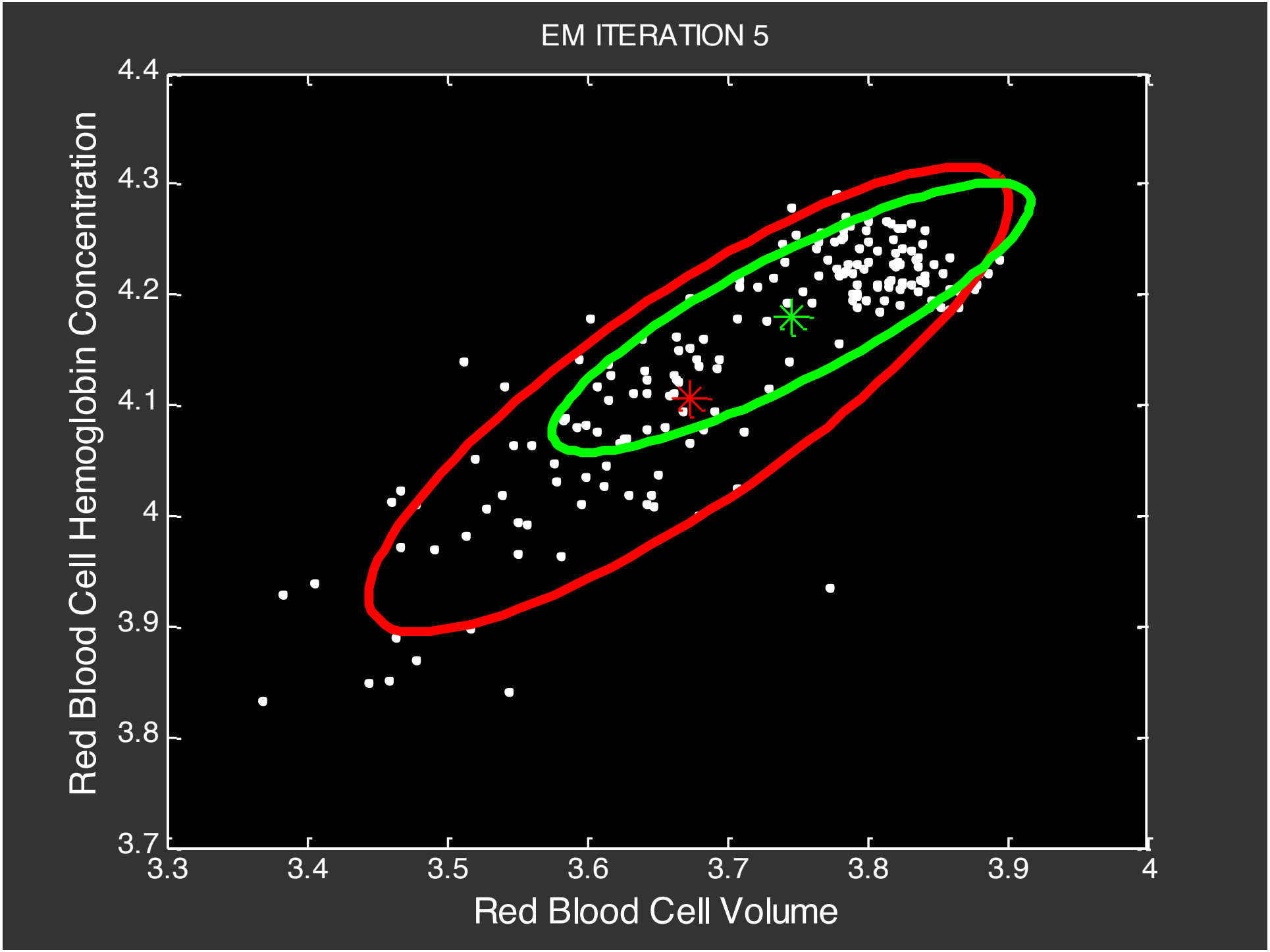
Mixture of Gaussians

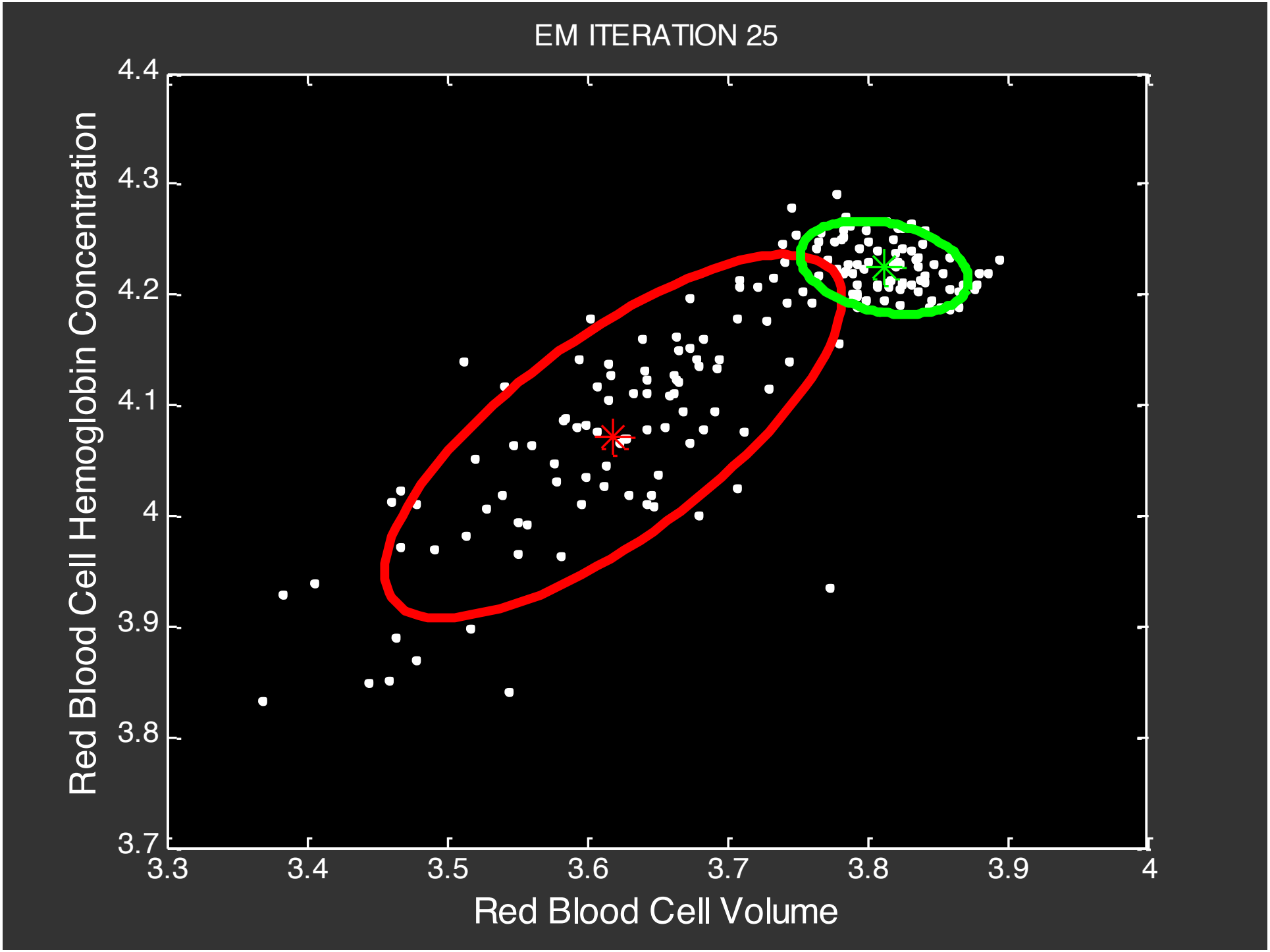


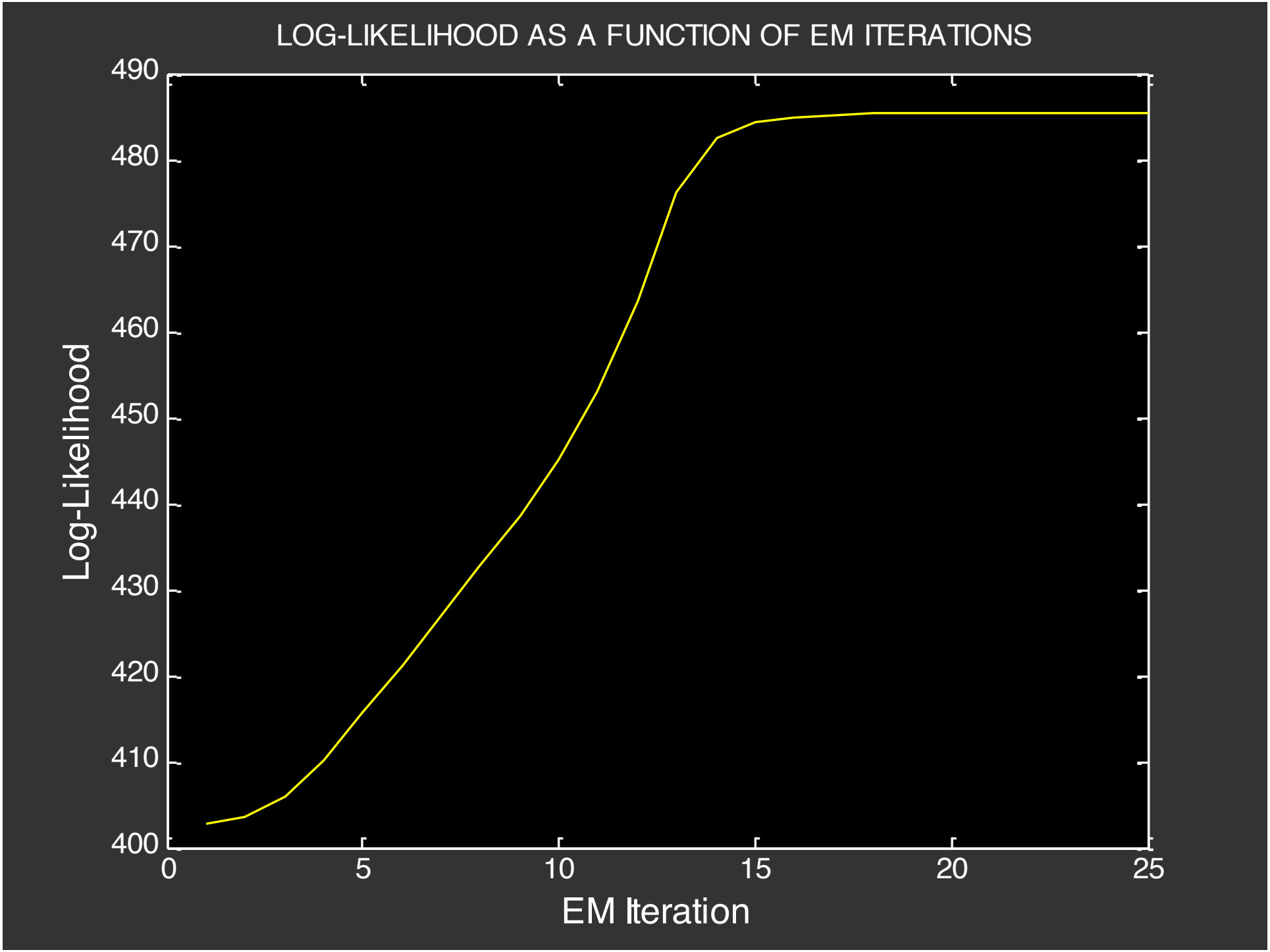

Component Models
p(x)
2
1.5
1
0.5
0
-5
0
5
10
Mixture Model
p(x)
0.5
0.4
0.3
0.2
0.1
0
-5
0
5
10
x

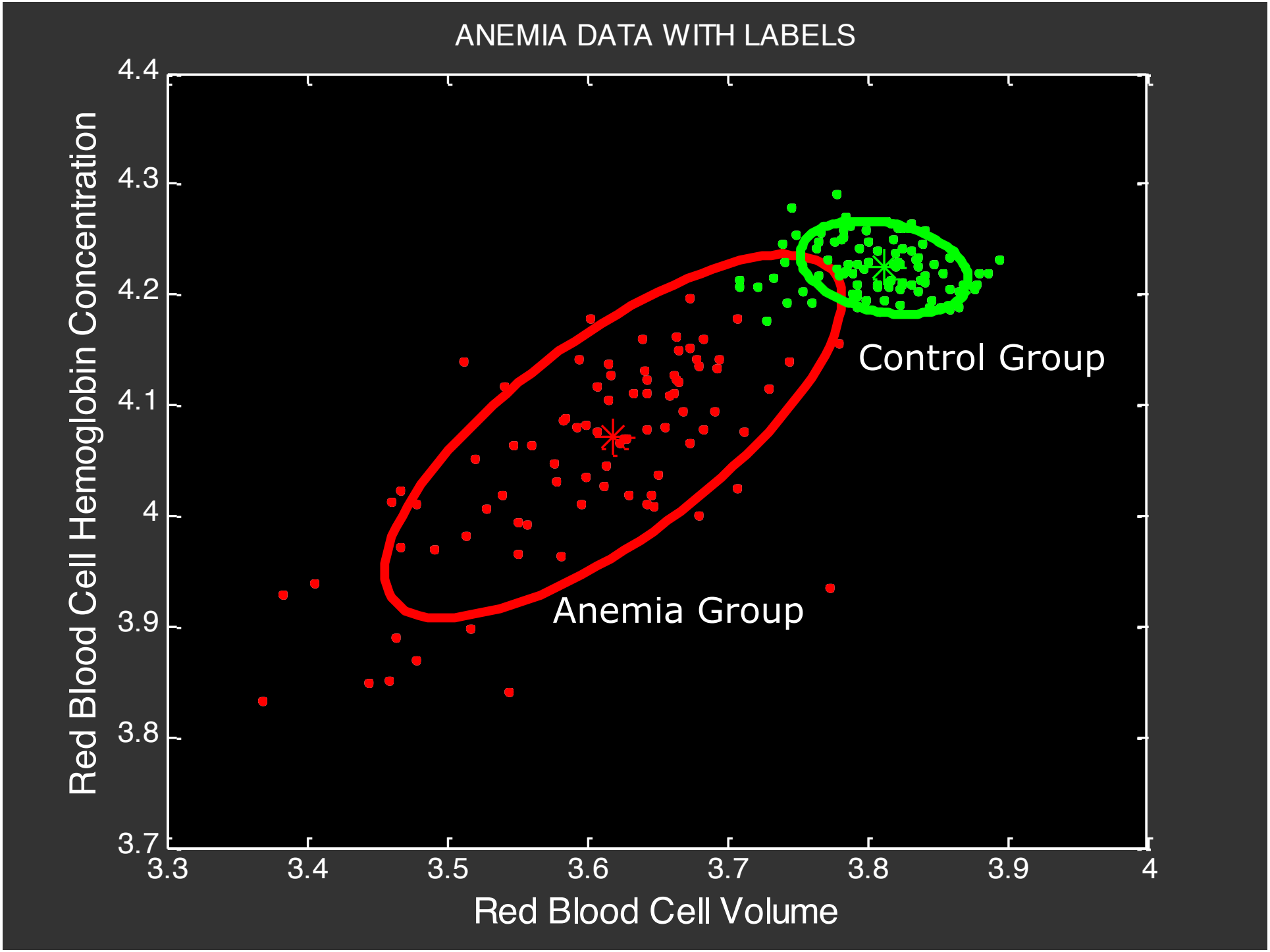

ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


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 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 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


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


ANEMIA DATA WITH LABELS
Red Blood Cell Hemoglobin Concentration
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
Red Blood Cell Volume
Control Group
Anemia Group

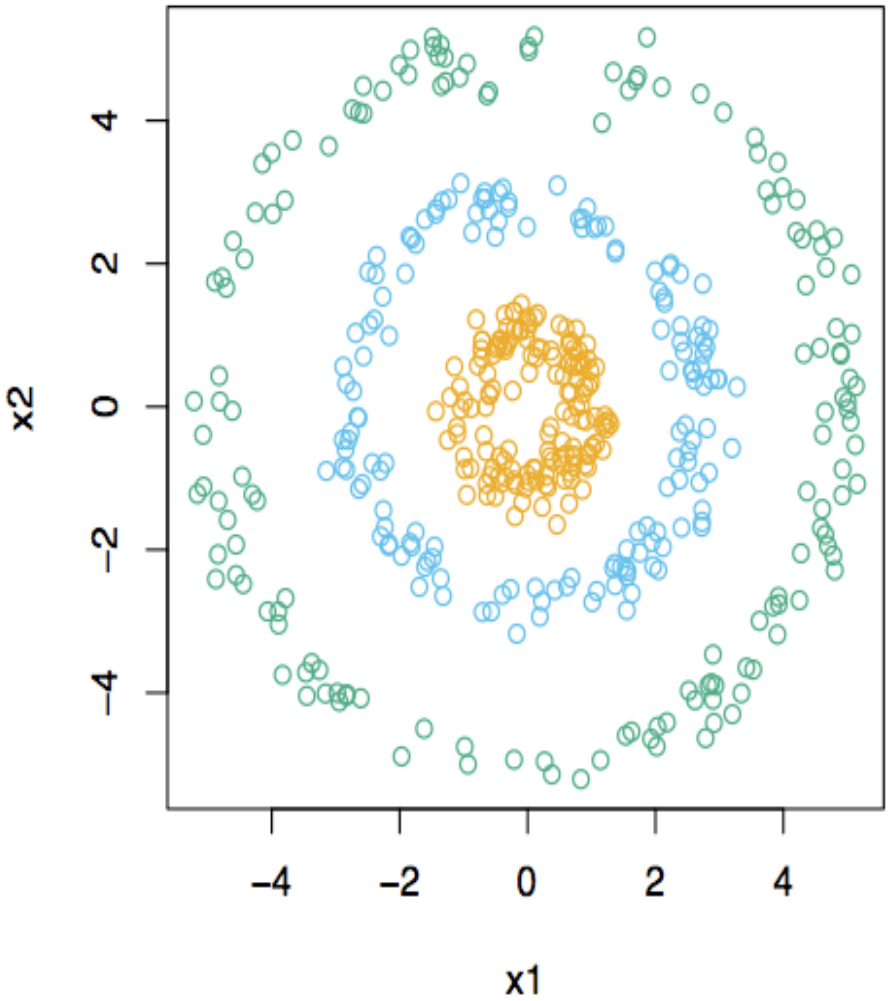
Spectral Clustering

Reference: Ulrike Von Luxburg. 2007. "A tutorial on spectral clustering." Statistics and Computing

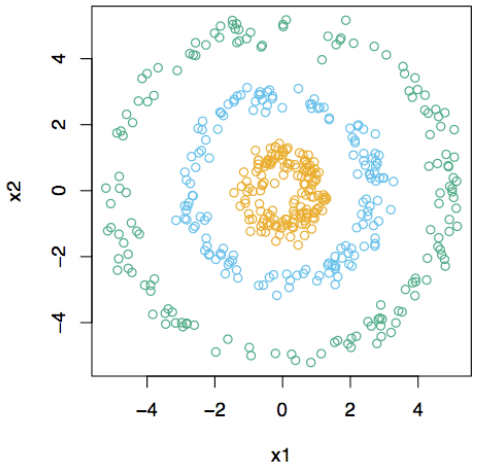
- Most useful when the data clusters are not spherical

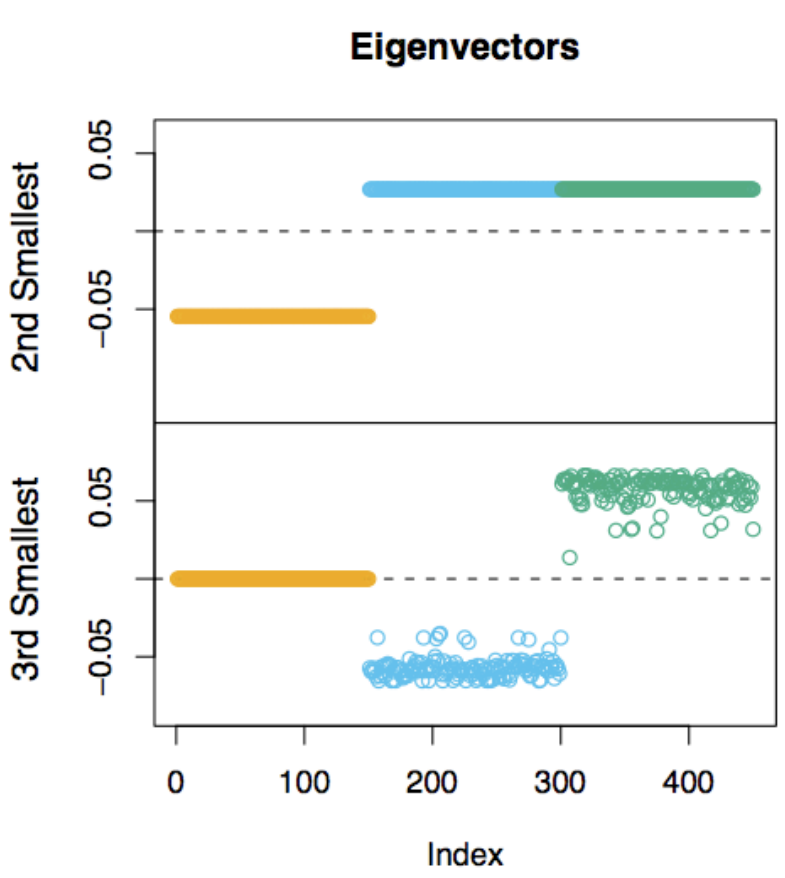
- Intuition: random walk

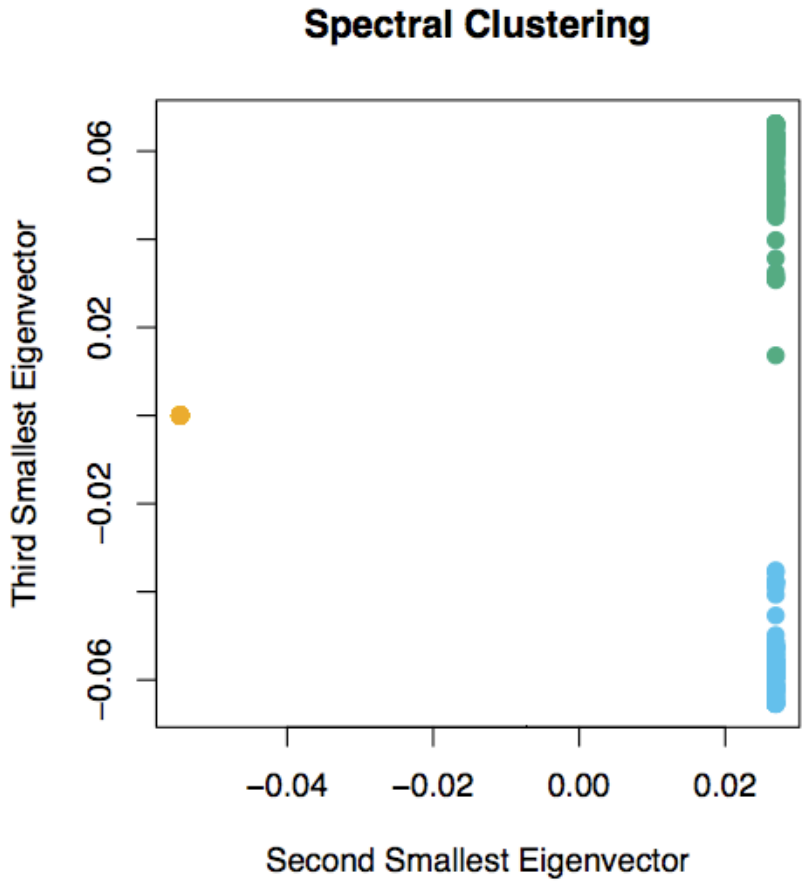
1. Compute the **similarity graph**.
2. Compute the eigenvectors of the resulting **Laplacian matrix**.
3. Compute the data clustering using K-means on projections on lowest eigenvectors.



How it works


x2
x1
4
2
0
−2
−4
−4
−2
0
2
4

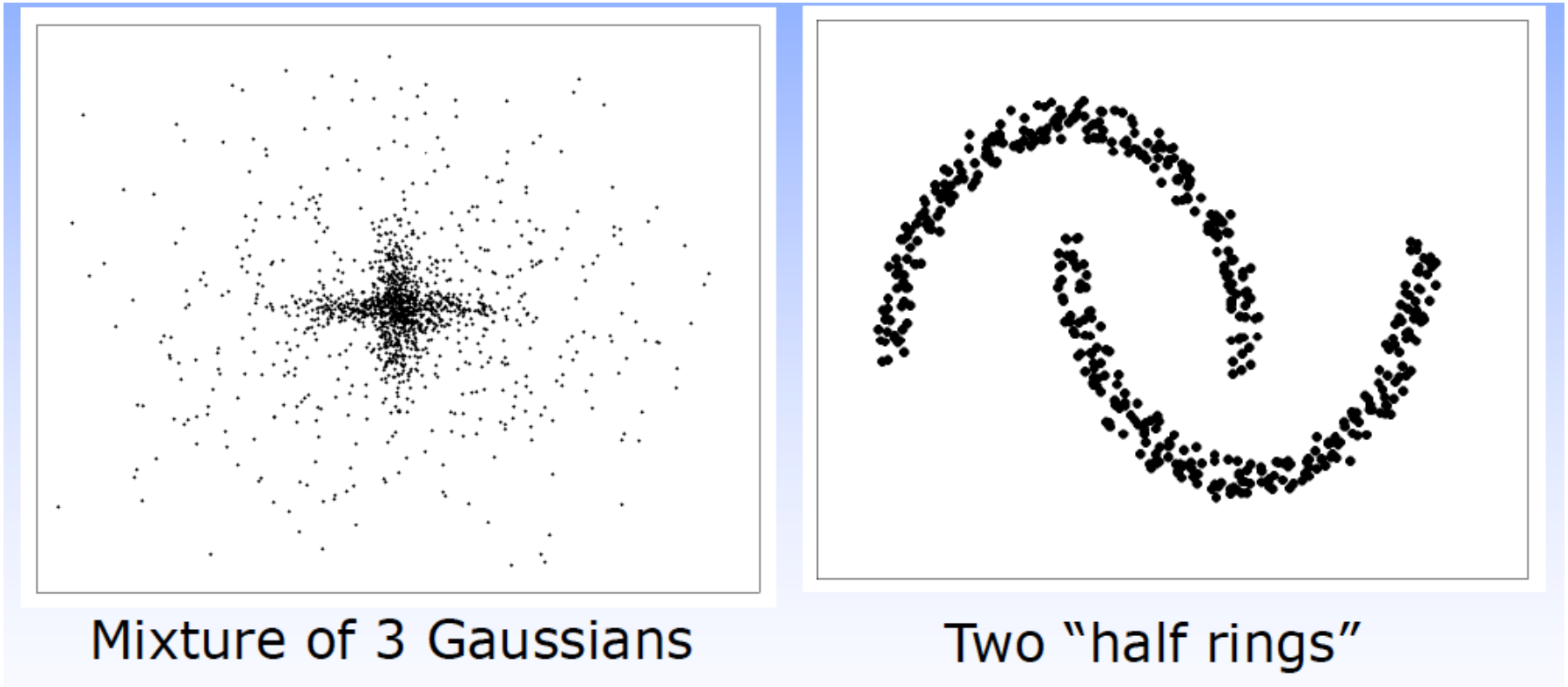

Eigenvectors
2nd Smallest
3rd Smallest
0.05
−0.05
0.05
−0.05
0
100
200
300
400
Index


Spectral Clustering
Third Smallest Eigenvector
0.06
0.02
−0.02
−0.06
−0.04
−0.02
0.00
0.02
Second Smallest Eigenvector

No Best Clustering Algorithm

Need good data representation + good fit with model

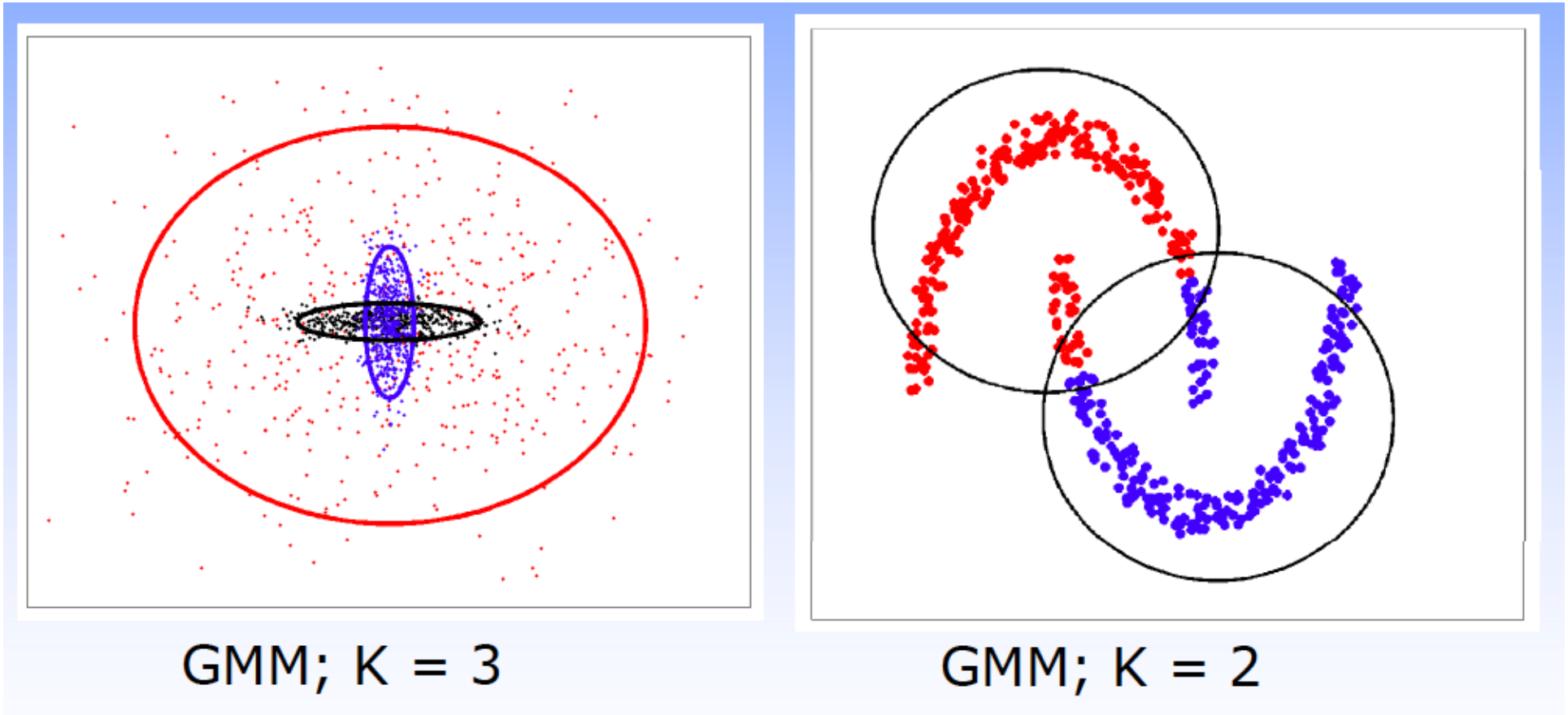
Each model imposes its own structure (assumptions) on the data



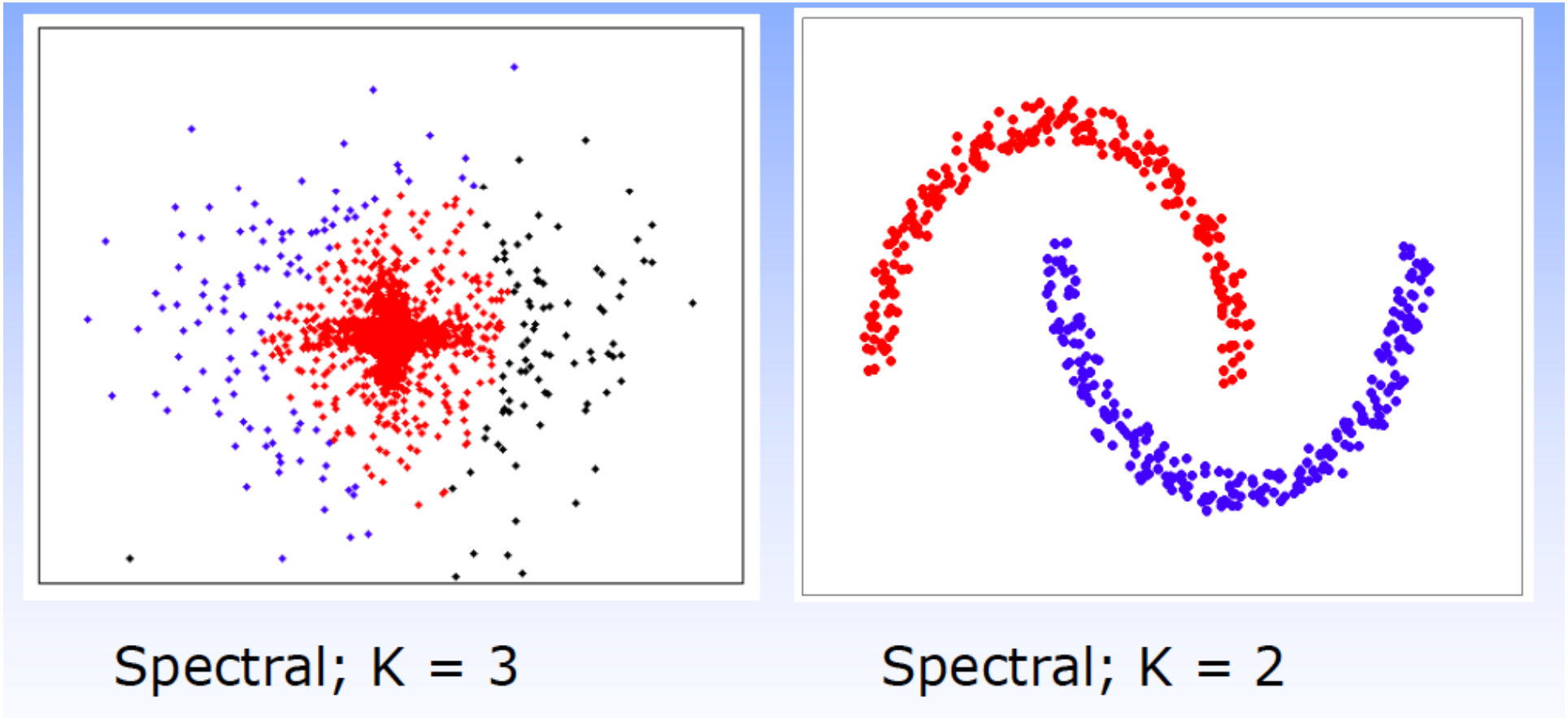
Mixture of 3 Gaussians

Two "half rings"

GMM solution



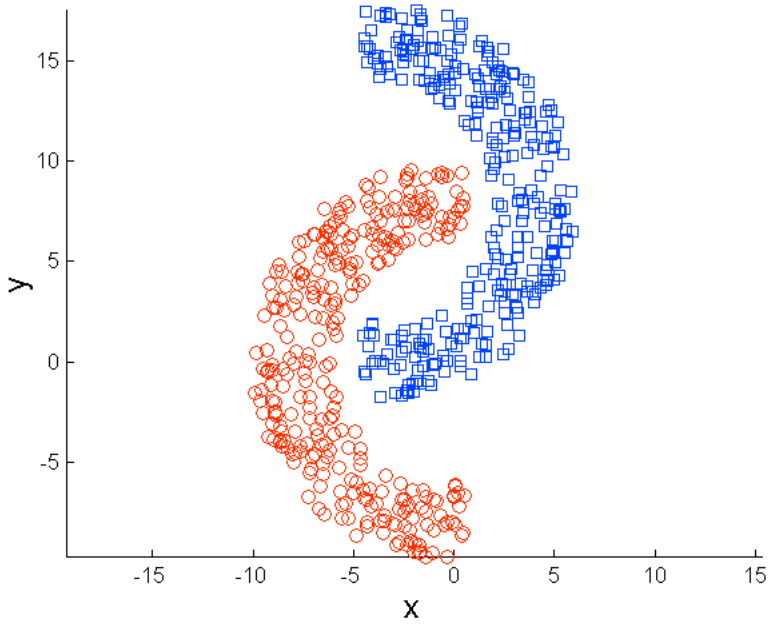
.. Vs. Spectral Clustering



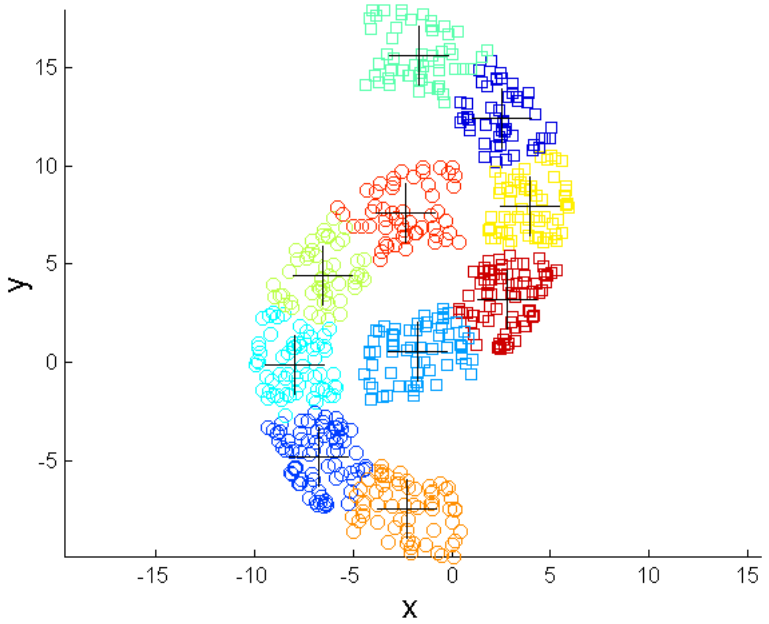
Spectral; K = 3

Spectral; K = 2

Overcoming K-means Limitations



Original Points



K-means Clusters

K-means + SL hierarchical : also helps with scaling

Also K-means is easily "kernelizable",
i.e. one can do k-means in a(possibly implicit) high-dimensional space

Online/Streaming Kmeans

- Mean can be maintained online with O(1) memory!

- Competitive Learning
- Frequency sensitive competitive learning
  - Using “conscience mechanism”

- Fixed vs. decaying learning rate

General Issues in Clustering

- Similarity [0,1] or distance [0, infinity]
  - what distance metric to use
  - dealing with mixed attributes
- Scaling of data
- "scale" of clustering (how many clusters?)
  - Part of "model selection" problem
  - Several approaches for determining k, e.g. BIC (X-means), gap statistics, "cubic clustering criterion (SAS)
- Cluster Validity
- Cluster Quality

Distances

- Binary/Categorical
  - e.g. Jaccard's coefficient
    similarity = a / (a + b + c)

	+	-
+	a	b
-	c	d

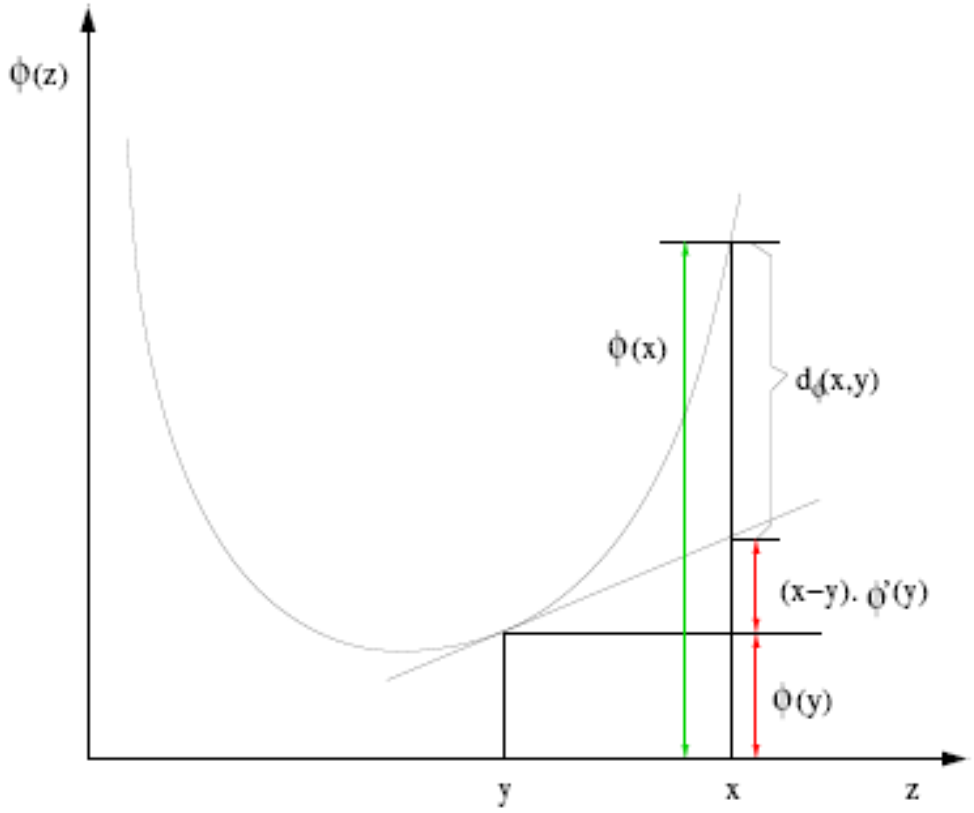
- Metrical:
  - L1 (Manhattan)
  - L2 (Euclidean)
  - Mahalonobis distance (scaled by variance)

- Mixed Data Types?

Clustering with Bregman Divergences

- Banerjee, Merugu, Dhillon, Ghosh, 2004
  - Extends both hard and soft clustering to all Bregman divergences
    - With same convergence guarantees!
  - Hard Clustering: KMeans-type algo possible for any Bregman Divergence
  - Bijection: convex function <--> Bregman divergence <--> exp. Family
  - Soft Clustering: efficient algo for learning mixtures of any exponential family
    - Rate distortion <--> max. likelihood

Bregman Divergences



ϕ is strictly convex, differentiable

$$d_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y}) \rangle$$

Bregman Loss Functions

- $\phi(x) = x^2$ is strictly convex and differentiable on $\mathbb{R}$
  - $D_\phi(x, y) = (x - y)^2$ (squared Euclidean distance)

- $\phi(\mathbf{p}) = \sum_{j=1}^{d} p_j \log p_j$ (negative entropy) is strictly convex and differentiable on the d-simplex
  - $D_\phi(\mathbf{p}, \mathbf{q}) = \sum_{j=1}^{d} p_j \log\left(\frac{p_j}{q_j}\right)$ (KL-divergence)

- $\phi(x) = -\log x$ is strictly convex and differentiable on $\mathbb{R}_{++}$
  - $D_\phi(x, y) = \frac{x}{y} - \log\left(\frac{x}{y}\right) - 1$ (Itakura-Saito distance)

Bregman Hard Clustering

- Initialize $\{\mu_h\}_{h=1}^{k}$
- Repeat until *convergence*
  - { Assignment Step }
    Assign x to $\mathcal{X}_h$ if $h = \underset{h'}{\operatorname{argmin}}\, d_\phi(\mathrm{x}, \mu_{h'})$
  - { Re-estimation step }
    For all h

$$\mu_h = \frac{\sum_{\mathrm{x} \in \mathcal{X}_h} p(\mathrm{x})\, \mathrm{x}}{\sum_{\mathrm{x} \in \mathcal{X}_h} p(\mathrm{x})}$$

Algorithm Properties

- Guarantee: Monotonically decreases a global objective function $E[d_\phi(X_h, \mu_h)]$ till convergence
- Scalability: Every iteration is linear in the size of the input
- Exhaustiveness: If such an algorithm exists for a loss function $L(x, \mu)$, then L has to be a Bregman divergence
- Linear Separators: Hyperplane separators for all Bregman Divergences
- Mixed Data types:
  - Allows appropriate Bregman divergence for subsets of features

Bregman Soft Clustering

- Initialize $\{\pi_h, \mu_h\}_{h=1}^{k}$
- Repeat until *convergence*
  - { Expectation Step }
    For all x, h,

$$p(h|x) = \pi_h \exp(-D_\phi(x, \mu_h))/Z(x),$$

    where $Z(x)$ is the log-partition function
  - {Maximization step}
    For all h,

$$\pi_h = \frac{1}{n}\sum_x p(h|x)$$

$$\mu_h = \frac{\sum_x p(h|x)\ x}{\sum_x p(h|x)}$$

Some Additional Clustering Issues in DM

- scalability to large #records and/or attributes
  - sampling, I/O, ….
- categorical/ mixed data
- sparsity (market baskets)
- integrating with concept hierarchies
- minimize domain knowledge based "knobs"
- sensitivity to input order
- constraint-based clustering
- interpretability and usability
- integration with RDMS, SQL
- Incrementals (time, data, trends)

Scaling K-means to Large Databases:

- http://citeseer.ist.psu.edu/bradley98scaling.html

  requirements: single scan with possible early termination; work within limited RAM buffer, ...

- Idea 1: selectively store "important" portions of data; summarize other portions

- Idea 2: compress sub-clusters by locally fitting a Gaussian: sufficient stats is SUM, SUMSQ, N

Compression Steps:

- Primary Compression
  - compress all points within a Mahalanobis radius of estimated cluster center, or
  - study "worst case" scenario on perturbing cluster means within computed confidence intervals
- Secondary Compression:
  identify "tight" sub-clusters of points amongst data not discarded in the primary phase.
  locate dense portions of data
  apply tightness criteria
  use agglomerative clustering later to reduce # clusters to K.

One Scan, Many solutions:

- Refining Initial Points for K-Means Clustering
  http://citeseer.ist.psu.edu/bradley98refining.html
- Problem: K-means is sensitive to initial points
  - get J solutions from multiple (severely sub)-samples
    - yields CM, a set of JxK Cluster Means
  - (re-assign centers of singleton clusters if needed)
  - Cluster CM J times, one for each set of initial points
  - choose solution with minimum distortion over CM.

  Computationally efficient, but:
    - random sampling from database is still expensive
    - still need to know K.

Specialized "Scalable" Algorithms

- BIRCH
  - (Balanced Iterative Reducing and Clustering using Hierarchies)
  - Partition based
- CLIQUE
  - (Clustering In QUEst) - grid/density based
- Chameleon
  - Graph/density/agglomerative

BIRCH [Zhang/Ramakrishnan/Livny, 96]

- **Goal:** Cluster with single scan; limited main memory
- **Approach:**
  - (i) incrementally build an R-tree like indexing structure, pointing to summarized clusters
  - (ii) modify tree by further compaction if need be.

- Cluster Feature (CF)= {N, Sum, SumSQ} (summary info)
- Indexing structure: **Clustering Feature (CF)-Tree with 2** parameters:
  - B (max # entries/node, want node to fit in page)
  - T (max radius/variance of a cluster)

- R-tree like, with internal node entries [CF_i, $child_i$]
- Leaf nodes indicate the clusters, also described by CFs

BIRCH Algorithm

- **Step 1:** Insert entry into current CF tree:
  - traverse tree to find nearest leaf, enter and update
    - Not within T, create new entry. Update leaf stats, parent stats
    - Within T, and enough space (B)? Update leaf stats, parent stats
    - Not enough space? Split node into two, choosing farthest entrees as seeds
    - Splitting may propagate upwards
- **Step 2:** Condense CF tree and refine
  - use main memory agglomerative clustering (merge) on CF tree
  - Prune for outliers, etc.

Critique:

+ve : on-line, single scan

-ve: Tree insertion order dependent; need iterations to get appropriate K; negatives of K-means carry over, including high-D problems.

Clique [AGGR98] (Density based)

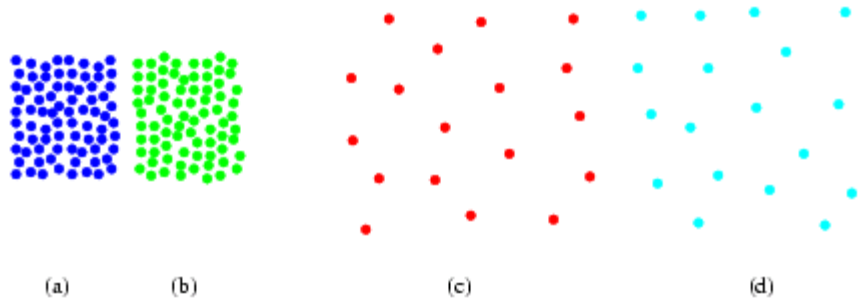
- idea: if k-dim data is dense, then so are its projections on k-1 dim space
  - find dense intervals in n-dim space
  - intersect with 1-dim dense intervals to get candidate dense n+1 dim intervals ("apriori" style!)

- Critique:
  - generates minimal, rectangular descriptions
  - simple; not very accurate

Chameleon [KHK 99]

An agglomerative, hierarchical, DYNAMIC clustering algorithm.

- Automatically adapts to the internal characteristics of merged clusters

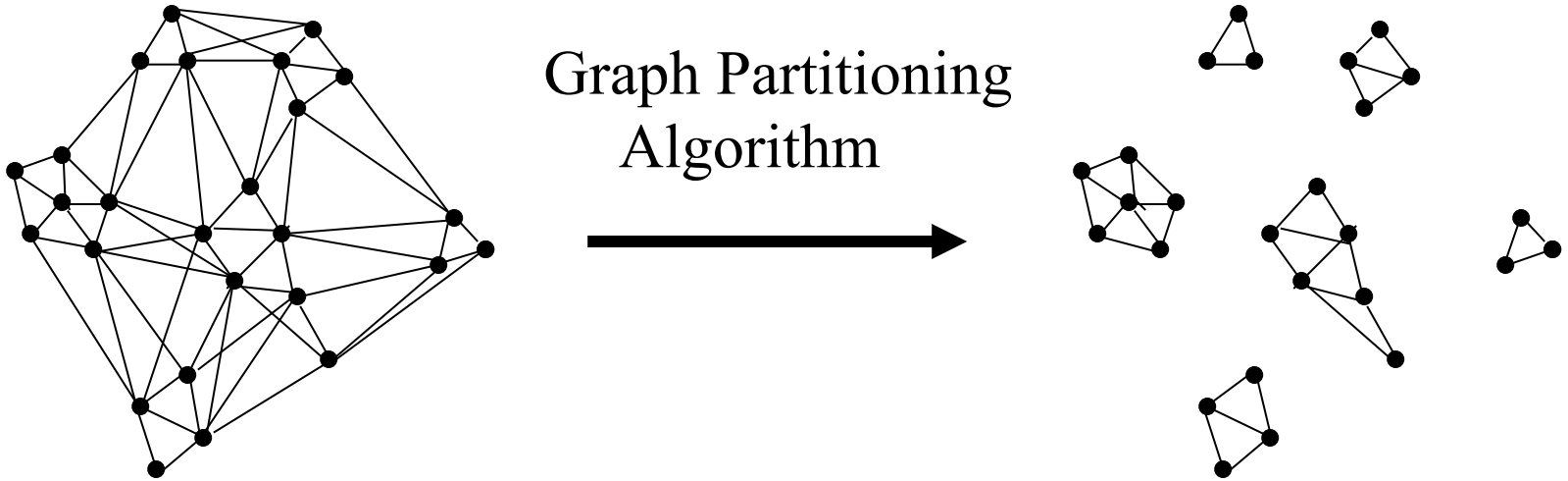


(which pair will you merge first?)

- Applicable to all types of data
- Can discover natural clusters of different shapes and sizes

CHAMELEON Phase I: Finding Initial Sub-clusters

- **Input:** similarity matrix
- **Construct** k-nearest neighbor graph (sparse, edges weighted by closeness)
- **Partition** using fast graph partitioning (METIS)



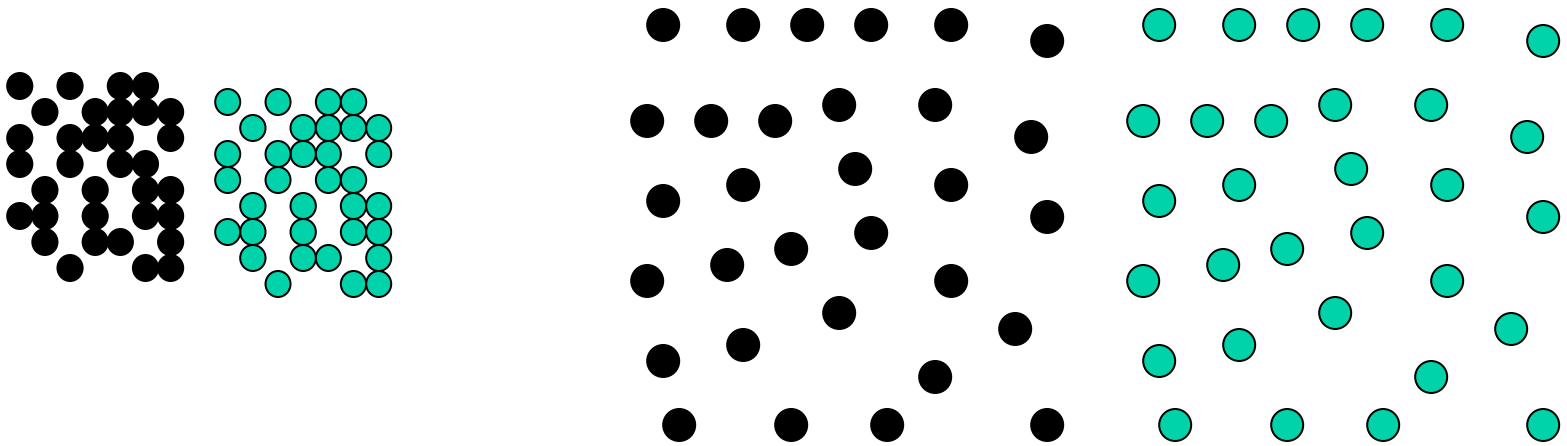
Phase II: Merging Sub-Clusters

- **CHAMELEON selects the most similar cluster-pairs to merge hierarchically on the basis of 2 criteria :**
  - **RELATIVE CLOSENESS**

  AND
  - **RELATIVE INTER-CONNECTIVITY**

Relative => dynamic

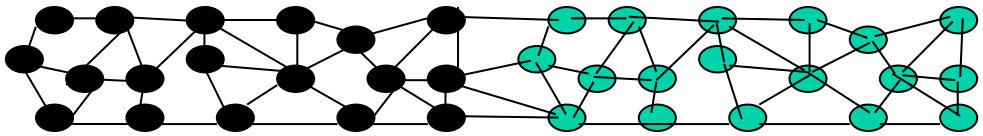
Absolute vs. Relative Closeness



Higher Absolute Closeness
Lower Relative Closeness

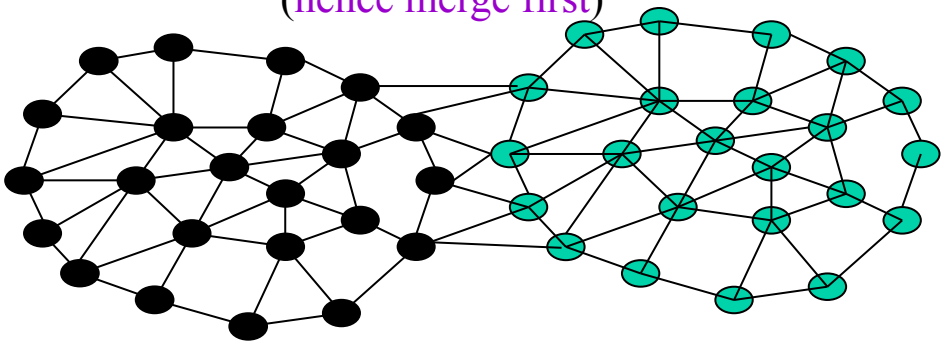
Lower Absolute Closeness
Higher Relative Closeness
(hence merge first)

Absolute vs. Relative Interconnectivity



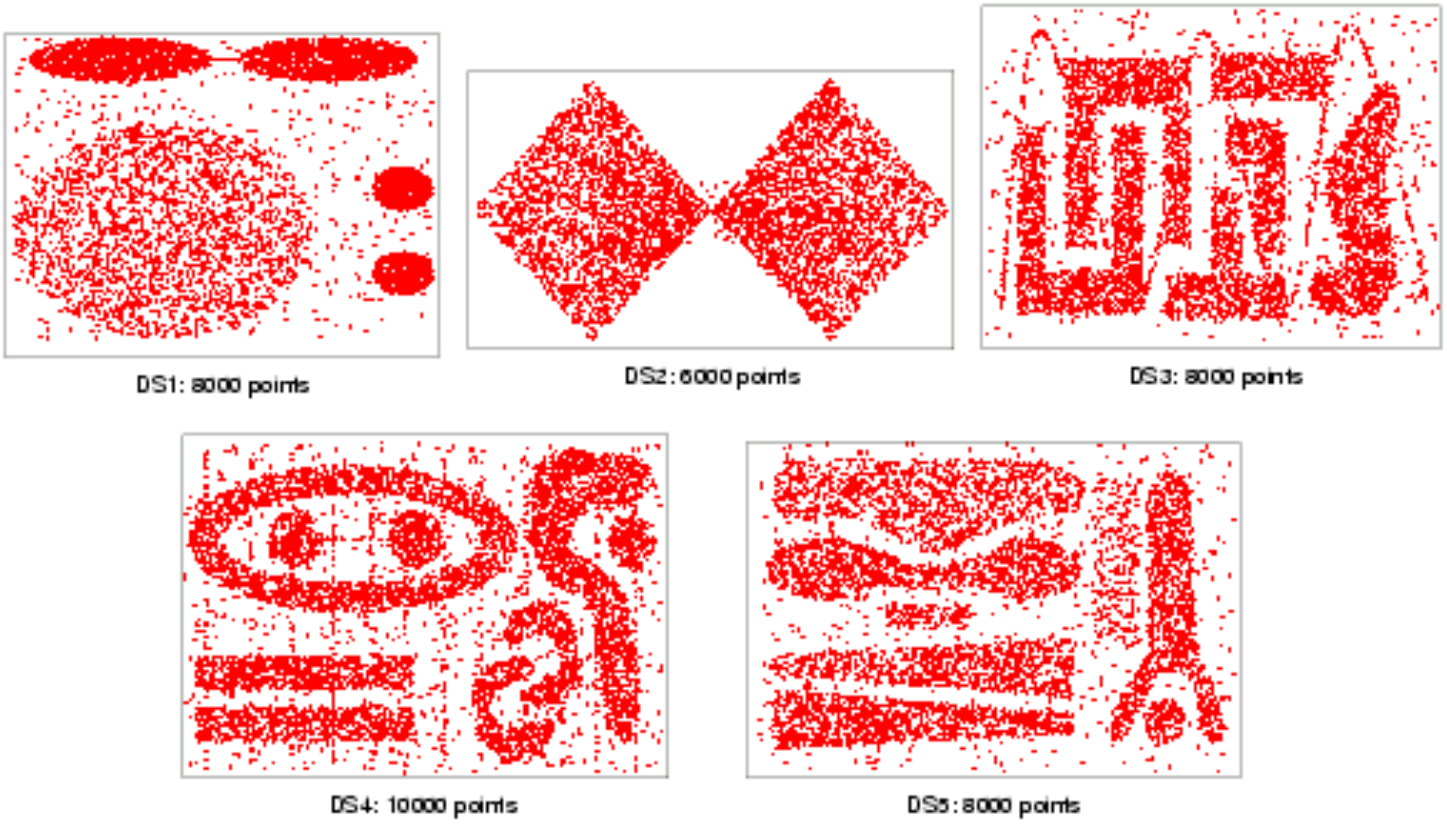
Lower Absolute Interconnectivity
Higher Relative Interconnectivity

(hence merge first)



Higher Absolute Interconnectivity
Lower Relative Interconnectivity

Results


DS1: 8000 points
DS2: 6000 points
DS3: 8000 points
DS4: 10000 points
DS5: 8000 points

Approaches to Clustering

- 1. Partitional (optimization; hill climbing)
  - K-means (popular! Problems?)
  - K-Medoids
  - FORGY, ….
- 2. Hierarchical (Dendograms)
  - Agglomerative (bottom up)
    - single linkage (min over closest points: generates min spanning tree(MST))
    - complete linkage (min over furthest points)
  - Divisive (top down)

Approaches to Clustering II

- 3. Density based
  - try to address linking problems
  - mode analysis of Wishart, ‘69
- 4. Graph Partitioning using Similarity Metric
  - suitable for large, sparse matrices
- 5 Soft clustering
  - Each object can belong to multiple clusters

Generative (model based or probabilistic)

vs. discriminative approaches

Evaluation of Clustering Algorithms

- type of data handled
- Assumptions about data distribution
- Quality
  - Extrinsic: (use "hidden" class labels), e.g. purity, entropy, mutual information,….
  - Intrinsic: cost function; compactness + separation,…

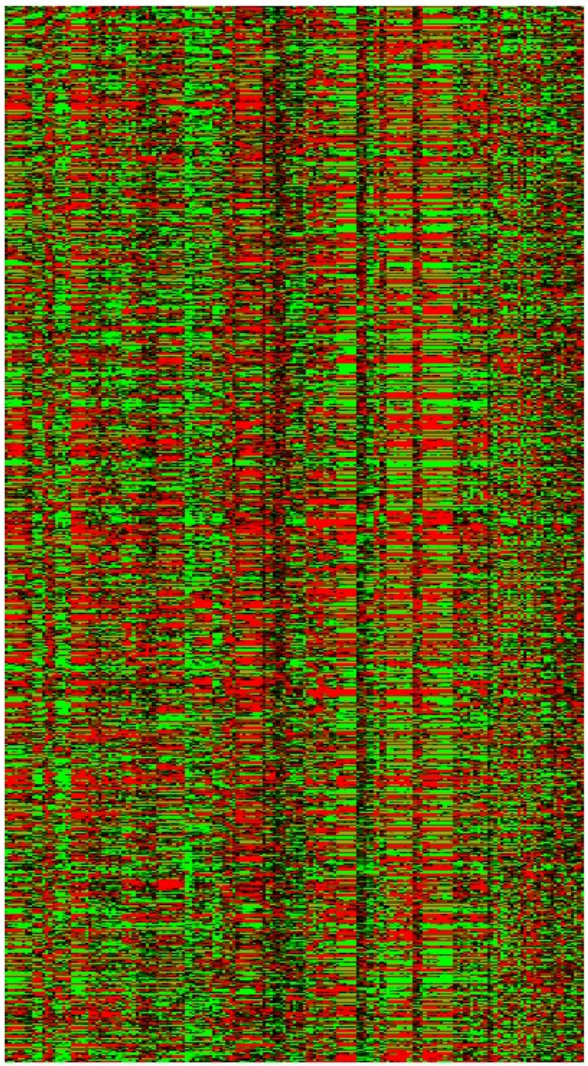
  ***WARNING: comparing across different k; across cost functions
- scalability

Co-clustering (Bi-clustering)

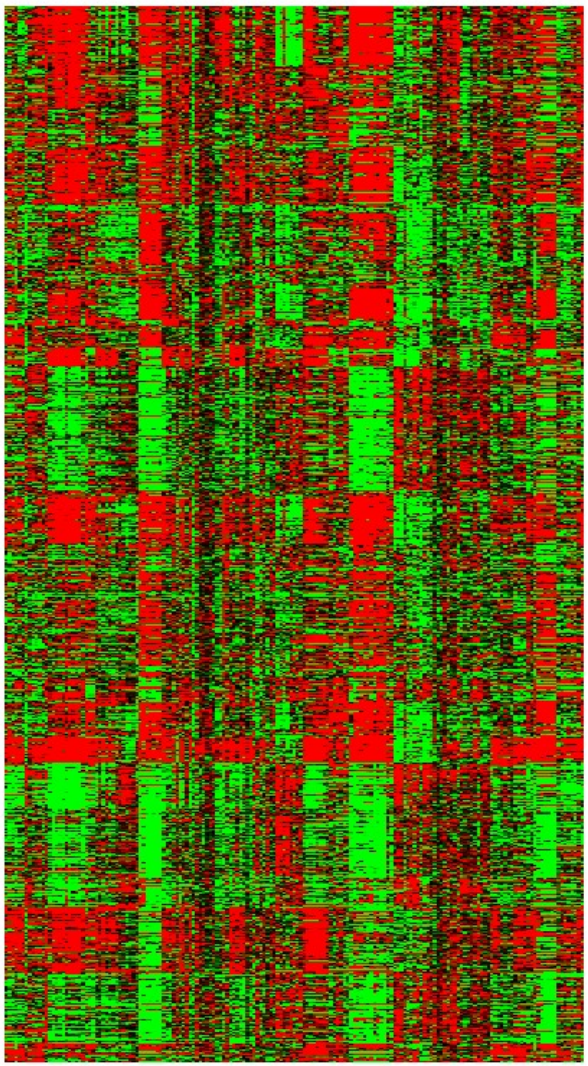
- Given a multi-dimensional data matrix, co-clustering refers to **simultaneous** clustering along multiple dimensions
- In a two-dimensional case it is simultaneous clustering of columns and rows
  - Simultaneous feature reduction and clustering.
- Applications:
  - Clustering of Text: Documents vs. words
  - Microarray Analysis: Genes vs conditions
  - Recommender Systems: Users vs. ratings

Co-clustering ("Biclustering") of Microarrays

Original Microarray Dataset



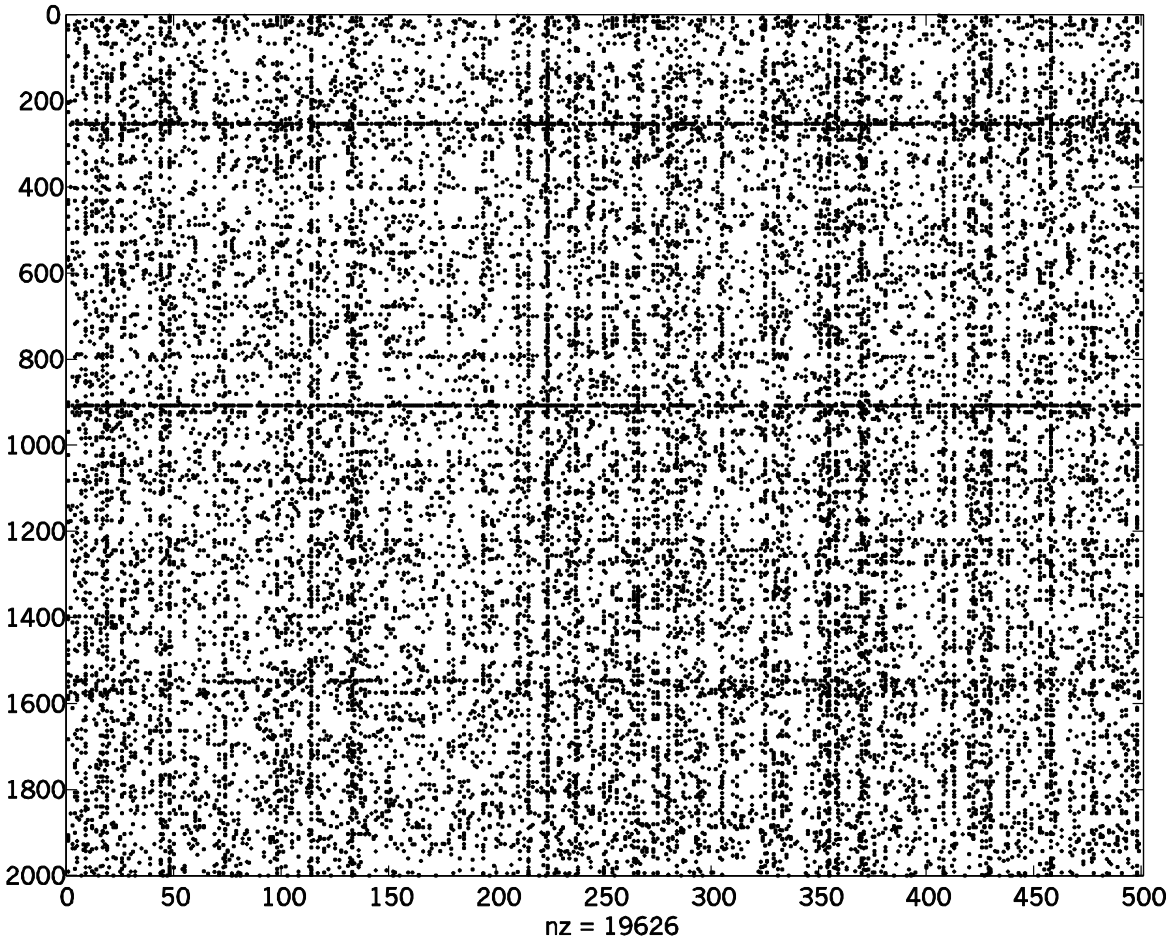
The Same Microarray Dataset after Co-Clustering



Rows in the original data (left) represent genes, columns are experimental conditions, and matrix entries depict expression levels (red: over-expressed, green: under-expressed).

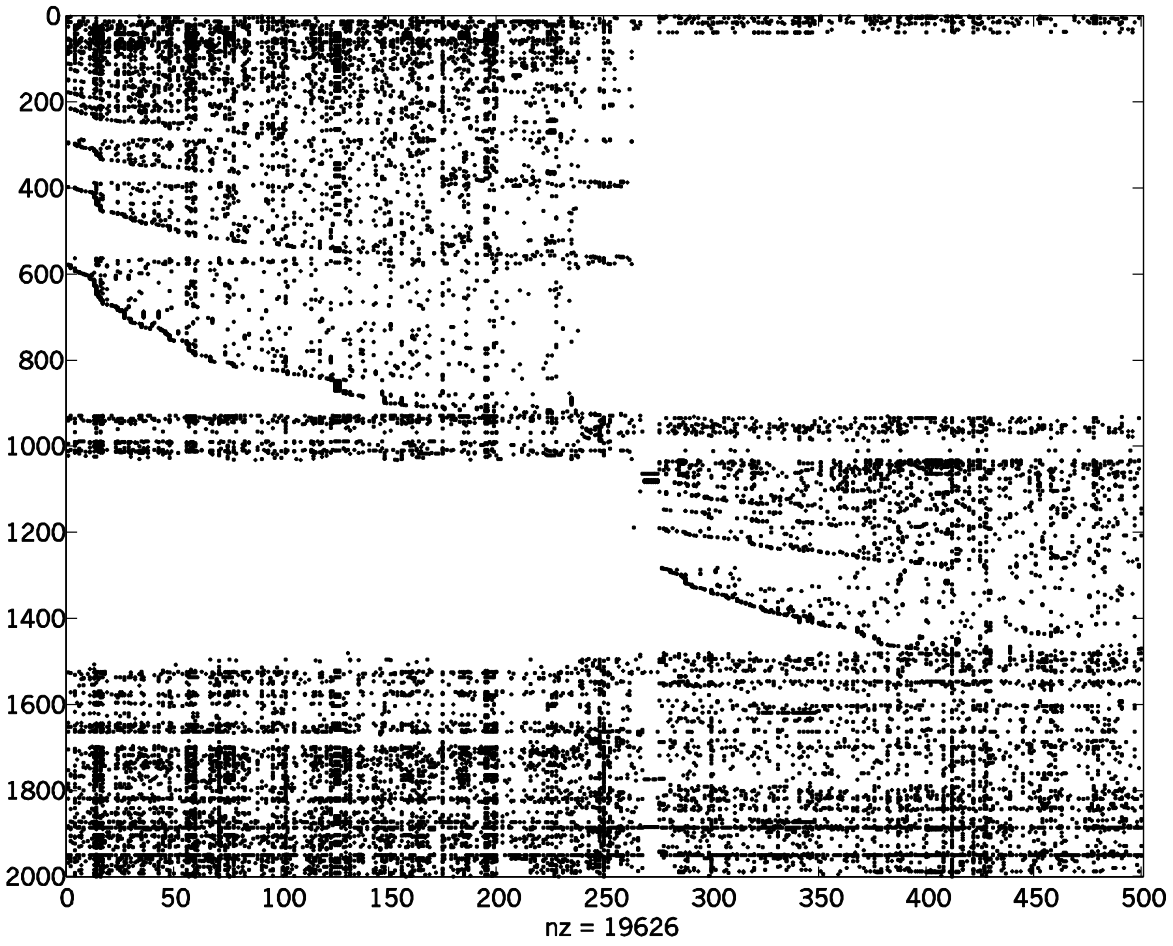
By simultaneously reordering both rows and columns, co-clustering clearly brings out groups of genes (visualized via block of red or green) that are over or under expressed for certain groups of experimental conditions.

Illustration using Classic3 dataset: Raw Data (docs vs. words)



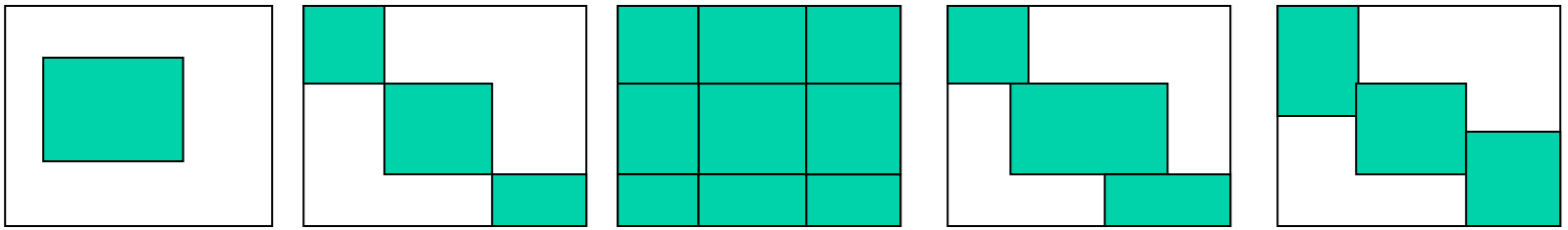
Result using Information Theoretic Co-clustering (Dhillon, Mallela, Kumar 03)

- Both rows and cols have been re-ordered.



Flavors of co-clustering structures and their relevance to Gene Array data

Various algorithms, irrespective of the similarity type, discover one or more of the following co-cluster structures, **in increasing order of generality** –



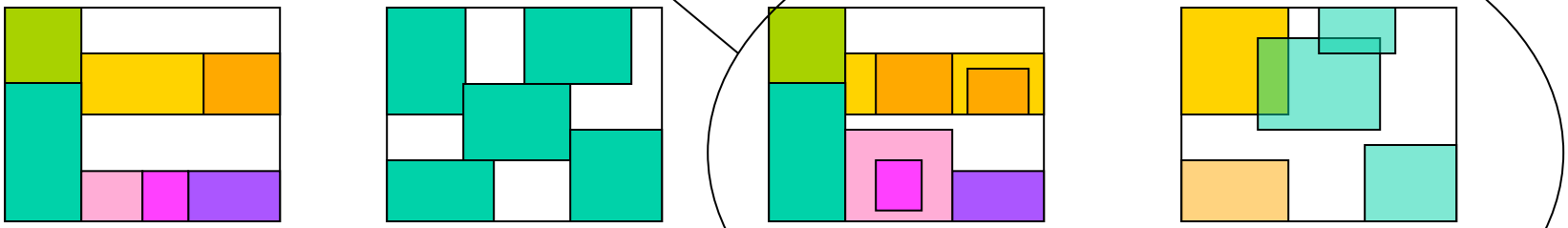
Single

Exclusive row & column

Checkerboard

Exclusive row, or Exclusive column

Most relevant to gene array data



Non-overlapping, tree structure

Non-overlapping, non-exclusive

Overlapping with hierarchical structure

Arbitrarily positioned overlapping

Semi-Supervised Clustering

Partition ***unlabeled*** data into groups + use limited supervision to aid and bias clustering

Goal: Examples in same cluster similar, separate clusters different + supervision is maximally respected

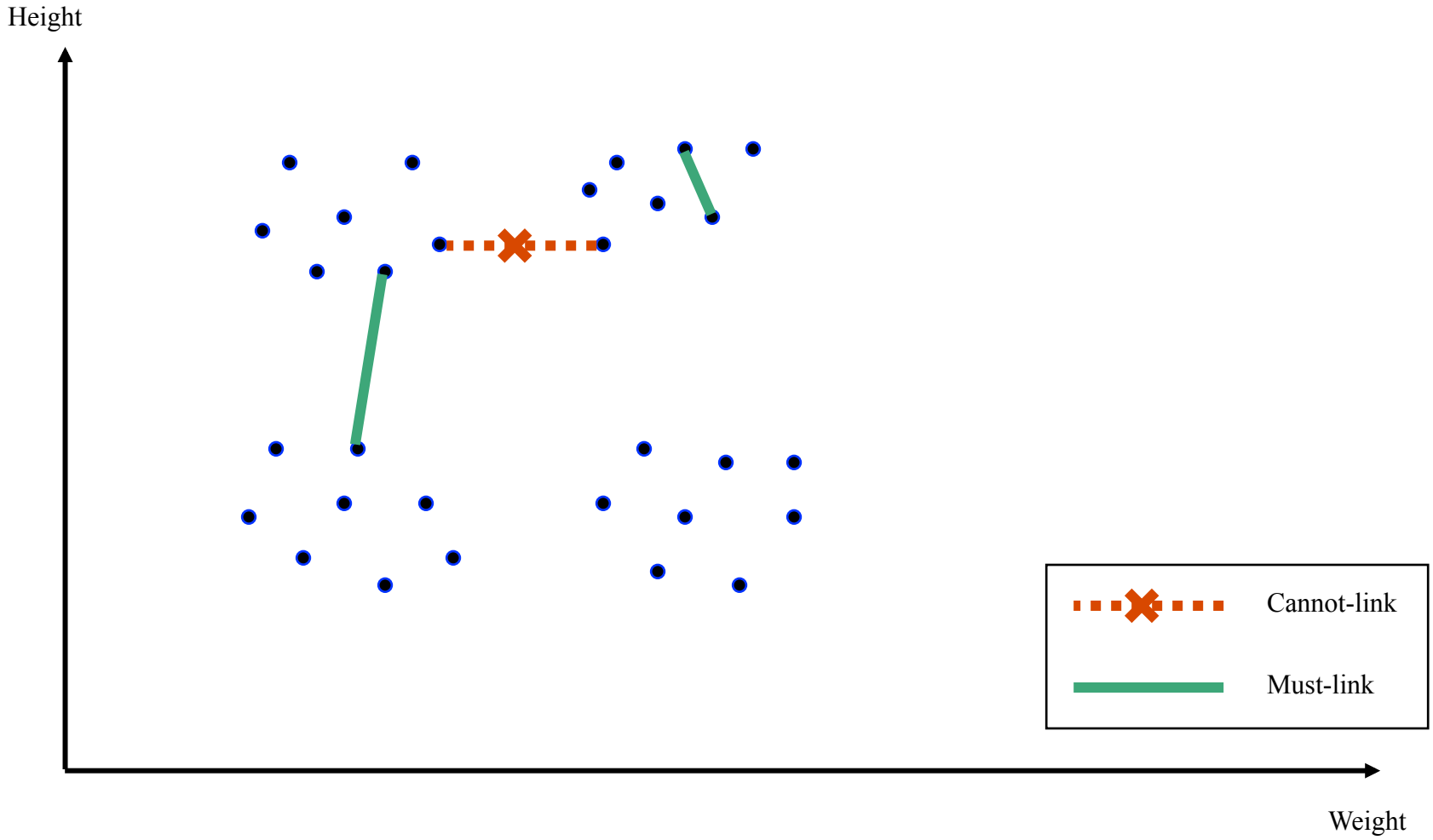
References:

Sugato Basu, Arindam Banerjee, and Raymond J. Mooney, **"Active Semi-Supervision for Pairwise Constrained Clustering",** *Proc.* SDM-2004.

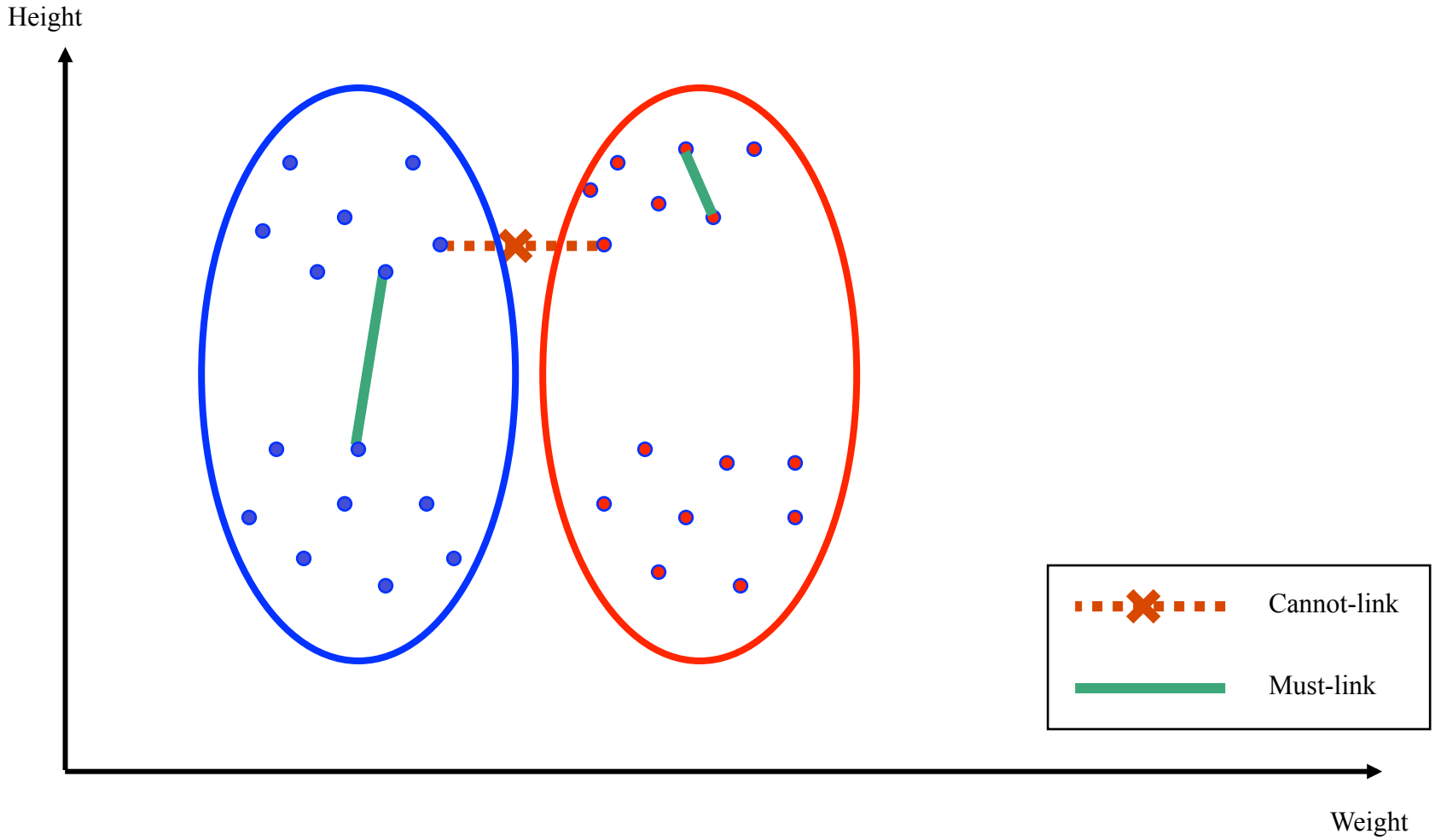
Ian Davidson and S. S. Ravi, **"Clustering** with Constraints and the k-means algorithm", Proc. 5th SDM, 2005.

Some of these slides are adapted with permission from a talk by Sugato Basu

Semi-supervised Clustering: **Constraint-based**



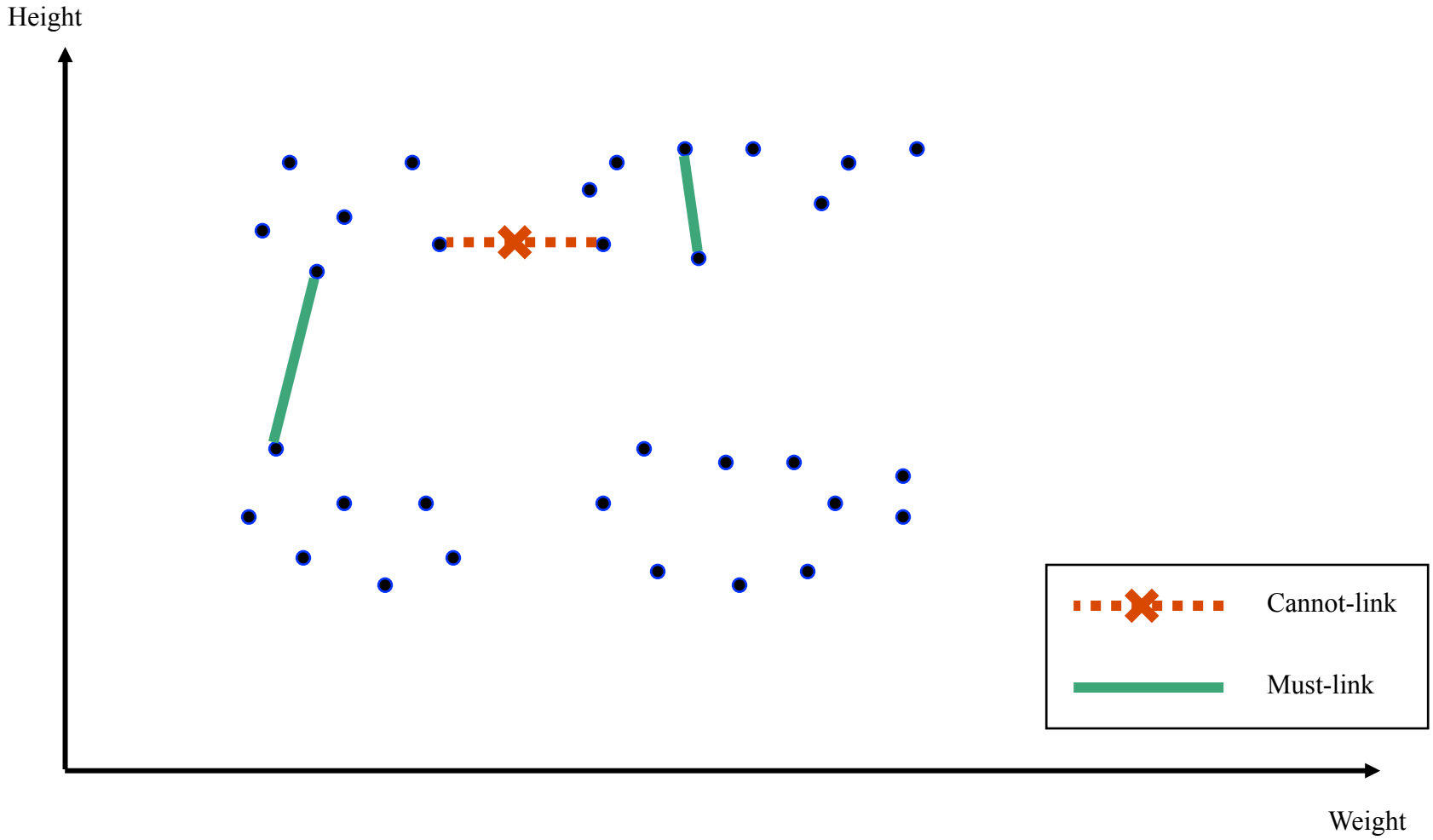
Semi-supervised Clustering: **Constraint-based**
Clustering respecting constraints



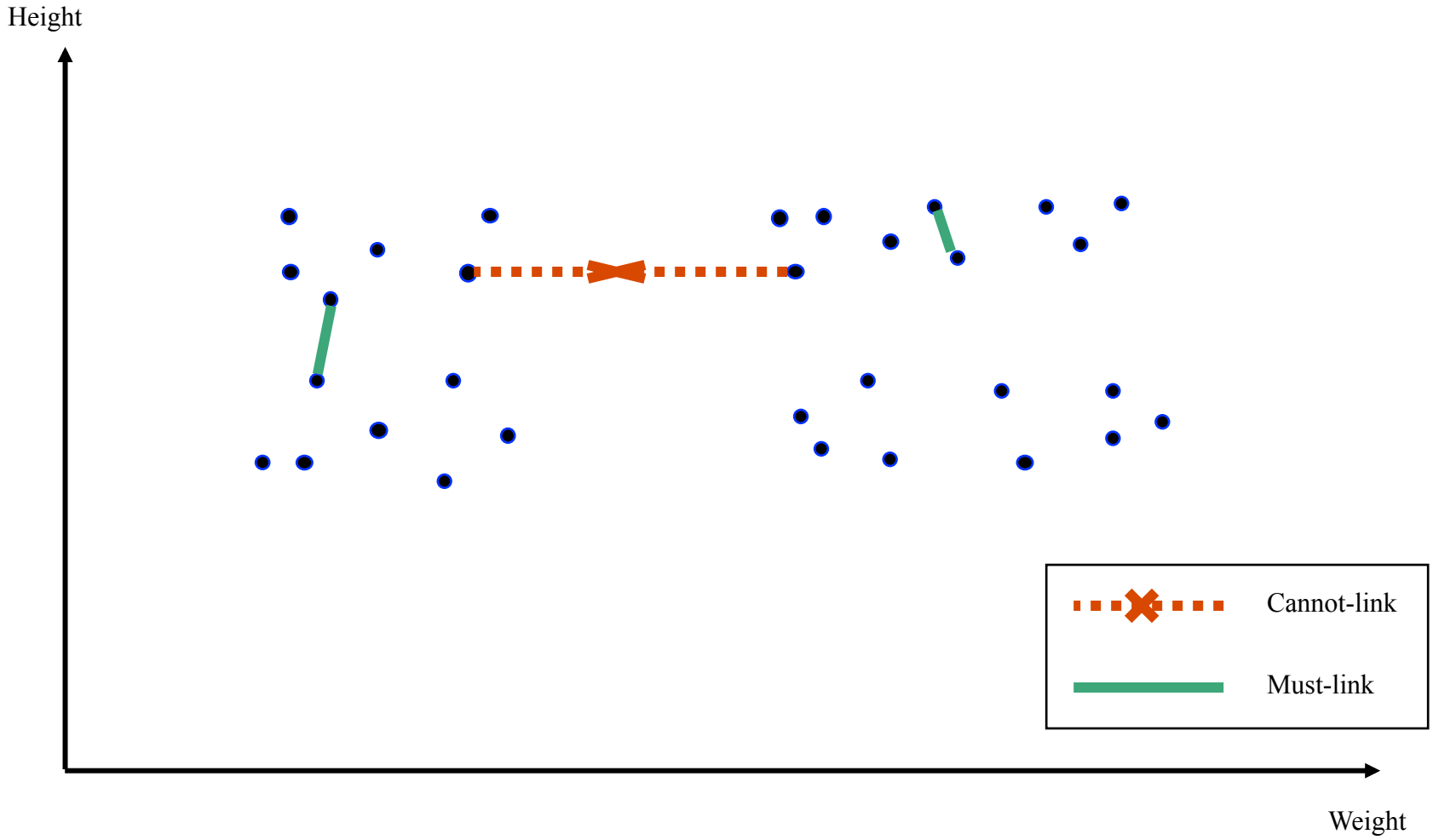
Semi-Supervised Clustering: Metric-based

- Metric-based approaches:
  - Supervision used to learn clustering distance metric
  - Standard clustering algorithm applied with learned metric

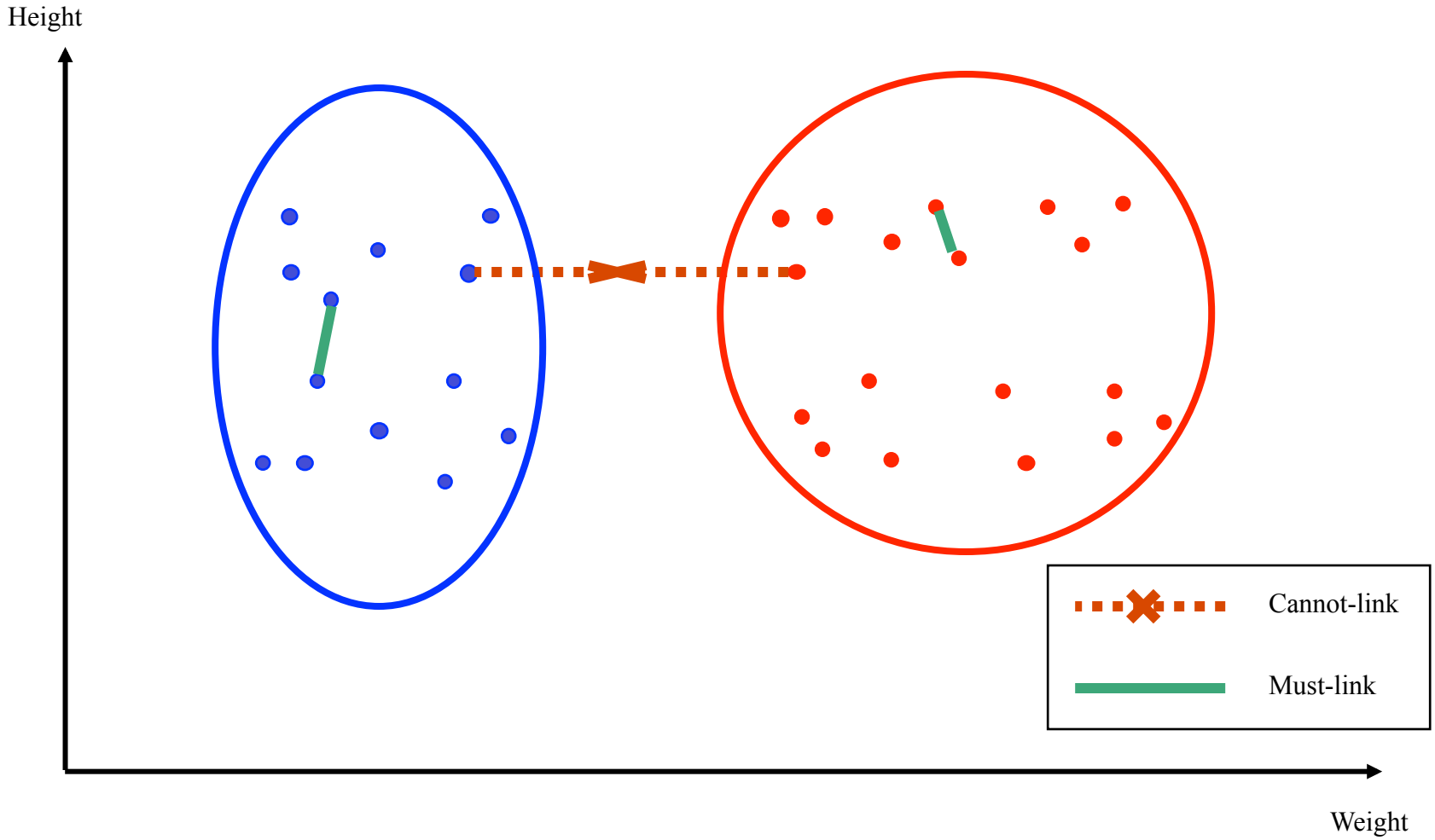
Semi-supervised Clustering: **Metric-based**



Semi-supervised Clustering: **Metric-based**
Distances Transformed by Learned Metric



Semi-supervised Clustering: **Metric-based**
Clustering with Trained Metric

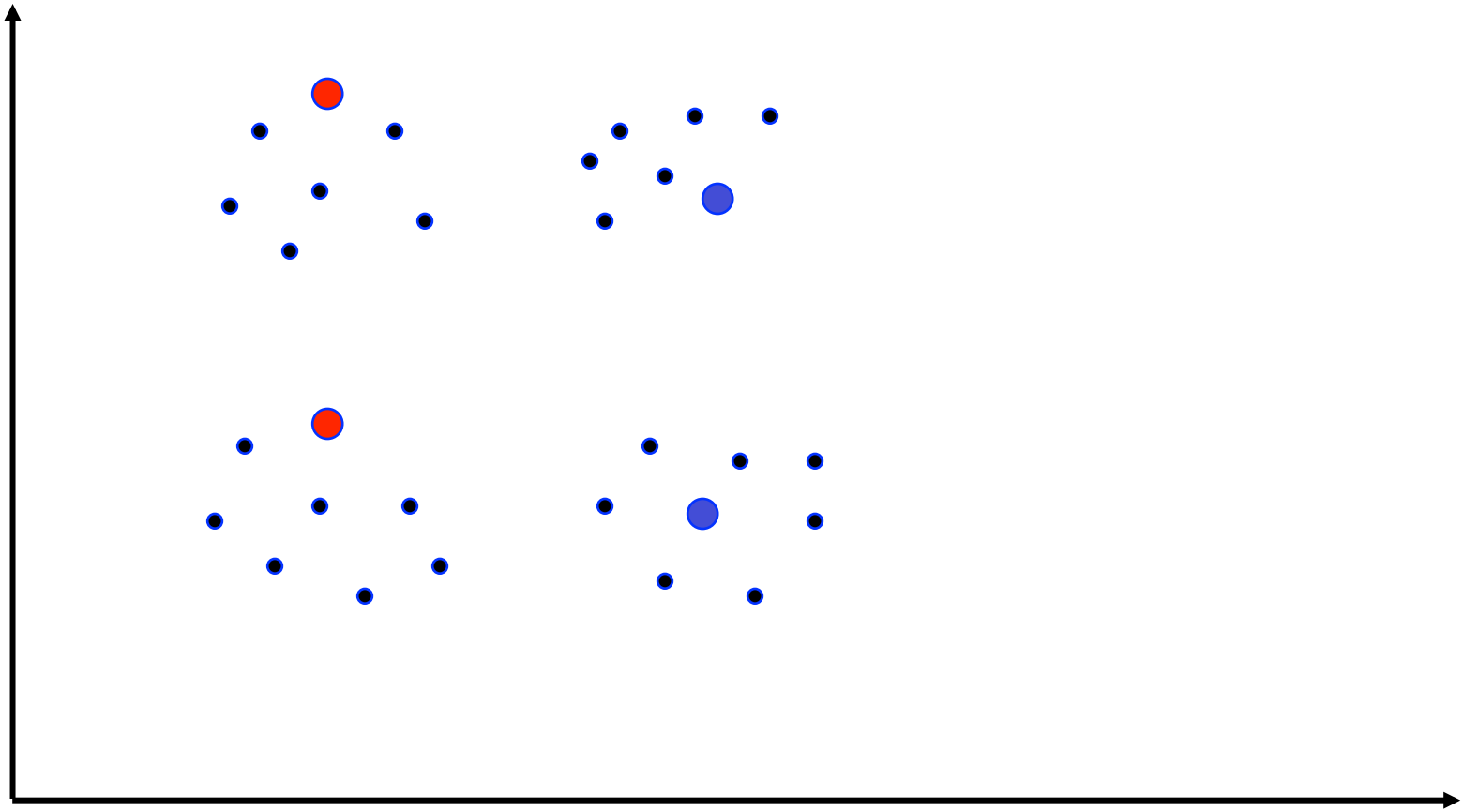


Semi-Supervised KMeans with (class) labels

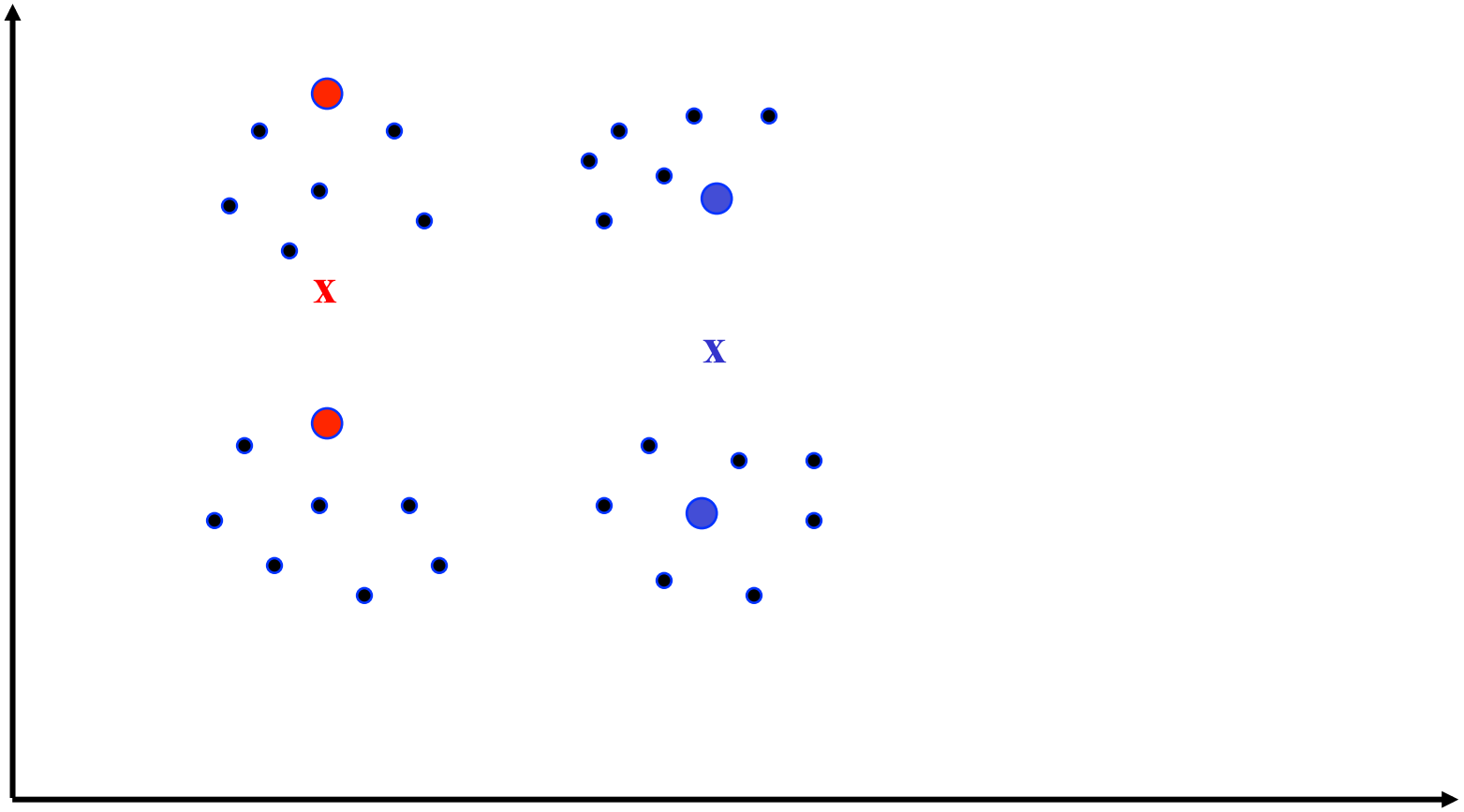
- **Seeded KMeans:**
  - Initial center for cluster *i* = mean of seed points having label *i*
    - If #seed labels < #clusters, initialize remaining centers randomly
  - Use seed points only for initialization, run normal KMeans

- **Constrained KMeans:**
  - Initialize cluster centers from seed points
  - Cluster labels of seed data are kept unchanged in the cluster assignment steps, only labels of the non-seed data are re-estimated
  - Mean re-estimation step unchanged

Semi-Supervised KMeans Example (K=2)
Pointwise Labels



Semi-Supervised KMeans Example
Initialize Means Using Labeled Data



Semi-Supervised KMeans Example
Assign Points to Clusters