# Unsupervised Data Mining: From Batch to Stream Mining Algorithms

Prof. Dr. Stefan Kramer
Johannes Gutenberg-Universität Mainz

# Acknowledgements

- E. Frank
- I. Witten
- … and others

# Outline

- k-Means
- Probability-based/model-based clustering: the EM algorithm

# K-Means

# Partitioning Algorithms: Basic Concept

- *Partitioning method*: construct a partition of a database D of n objects into a set of k clusters such that the sum of the squared distances is minimized

$$\Sigma_{m=1}^{k}\Sigma_{t_{mi}\in Km}(C_m - t_{mi})^2$$

- Given a k, find a partition of k *clusters* that optimizes the chosen partitioning criterion
  - globally optimal: exhaustively enumerate all partitions
  - heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen 67): each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (partition around medoids) (Kaufman & Rousseeuw 87): each cluster is represented by one of the objects in the cluster

# Goal of Clustering 1

- Total sum of pairwise distances:

$$T(C) = W(C) + B(C)$$

- Within cluster distances:

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d(x_i, x_{i'})$$

- Between cluster distances:

$$B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d(x_i, x_{i'})$$

# Goal of Clustering 2

- *Assuming* K *is fixed* in this simple setting, *maximizing* B(C) *and minimizing* W(C) *is the same!*
  - we assume real-valued data and use the ***squared** Euclidean distance*

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = \left\| x_i - x_{i'} \right\|^2$$

$$T = \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d(x_i, x_{i'}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left( \sum_{C(i')=k} d(x_i, x_{i'}) + \sum_{C(i')\neq k} d(x_i, x_{i'}) \right)$$

# Goal of Clustering 3

- Thus, the goal is to minimize:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} \left\| x_i - x_{i'} \right\|^2 = \sum_{k=1}^{K} \sum_{C(i)=k} \left\| x_i - \bar{x}_k \right\|^2$$

$$\min_{C} \sum_{k=1}^{K} \sum_{C(i)=k} \left\| x_i - \bar{x}_k \right\|^2$$

$$\min_{C,\{m_k\}_1^K} \sum_{k=1}^{K} \sum_{C(i)=k} \left\| x_i - m_k \right\|^2$$

- *Alternating optimization procedure…*

# K-Means Algorithm

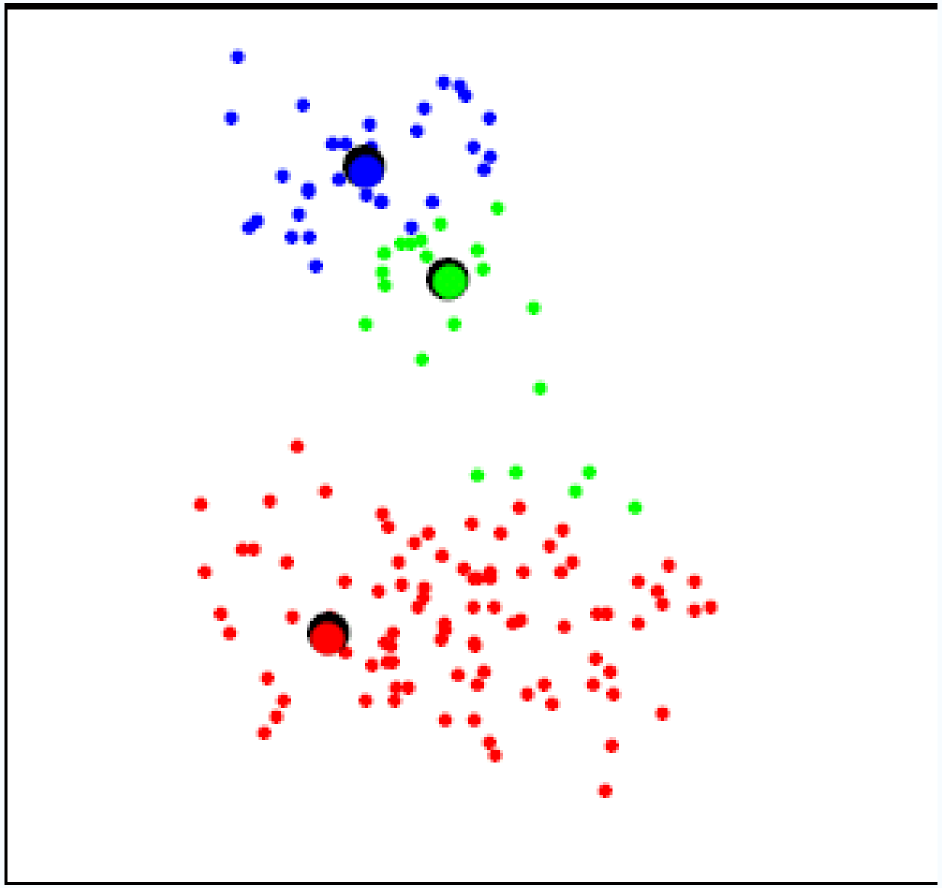
Given data points D ={$\mathbf{x}_1,\ldots,\mathbf{x}_n$} , find K clusters {$C_1,\ldots,C_K$}

**for** $k = 1, \ldots, K$ let $\mathbf{r}(k)$ be a randomly chosen point from $D$;
**while** changes in clusters $C_k$ happen **do**
    form clusters:
    **for** $k = 1, \ldots, K$ **do**
        $C_k = \{\mathbf{x} \in D \mid d(\mathbf{r}_k, \mathbf{x}) \leq d(\mathbf{r}_j, \mathbf{x})$ for all $j = 1, \ldots, K, j \neq k\}$;
    **end**;
    compute new cluster centers:
    **for** $k = 1, \ldots, K$ **do**
        $\mathbf{r}_k$ = the vector mean of the points in $C_k$
    **end**;
**end**;

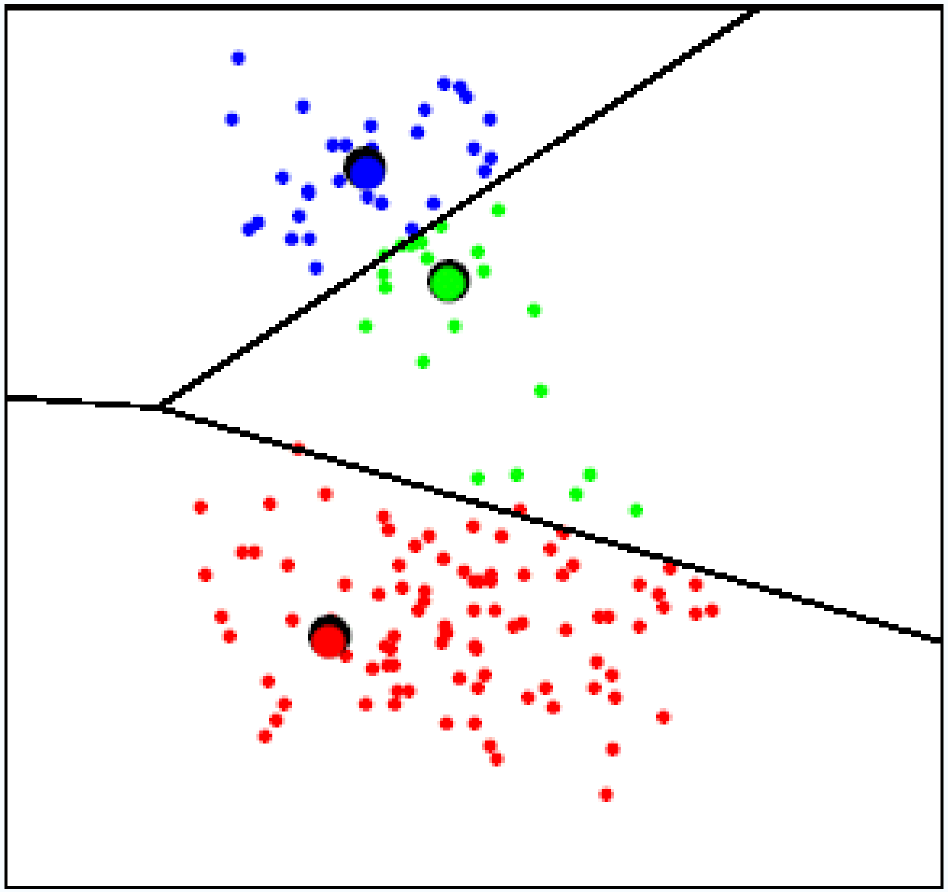
# Complexity

- K number of cluster centers
- n sample size
- I number of iterations
- *Suggestions?*

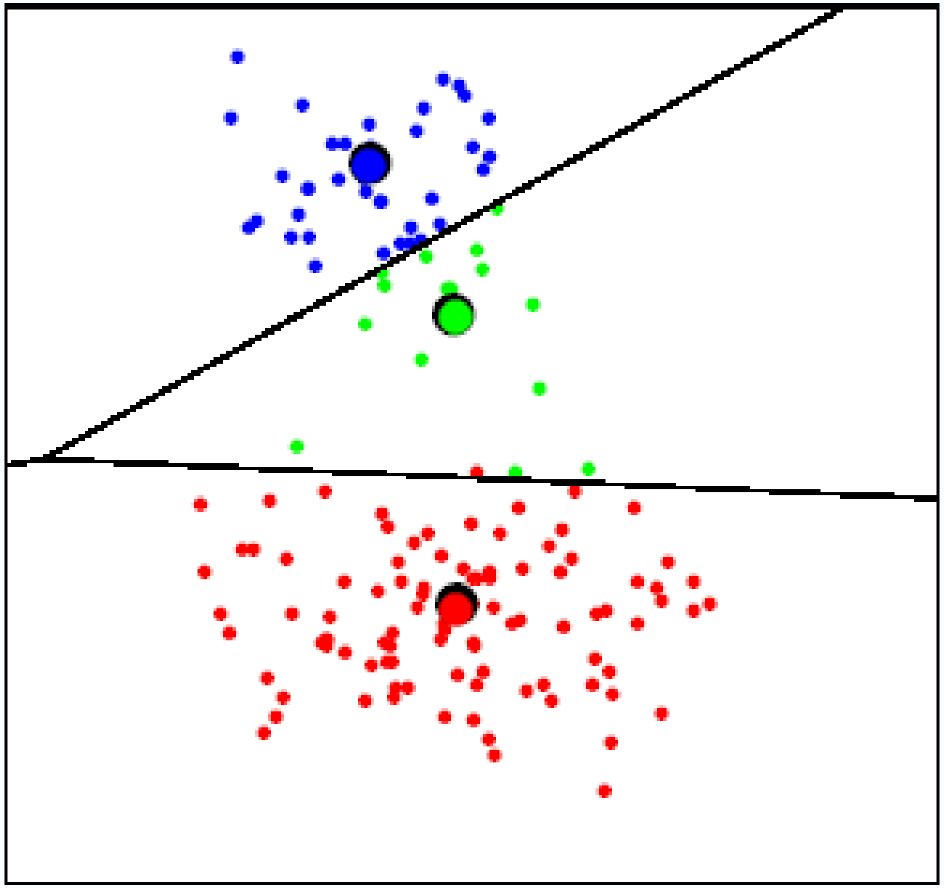
# K-Means: Initial Centroids



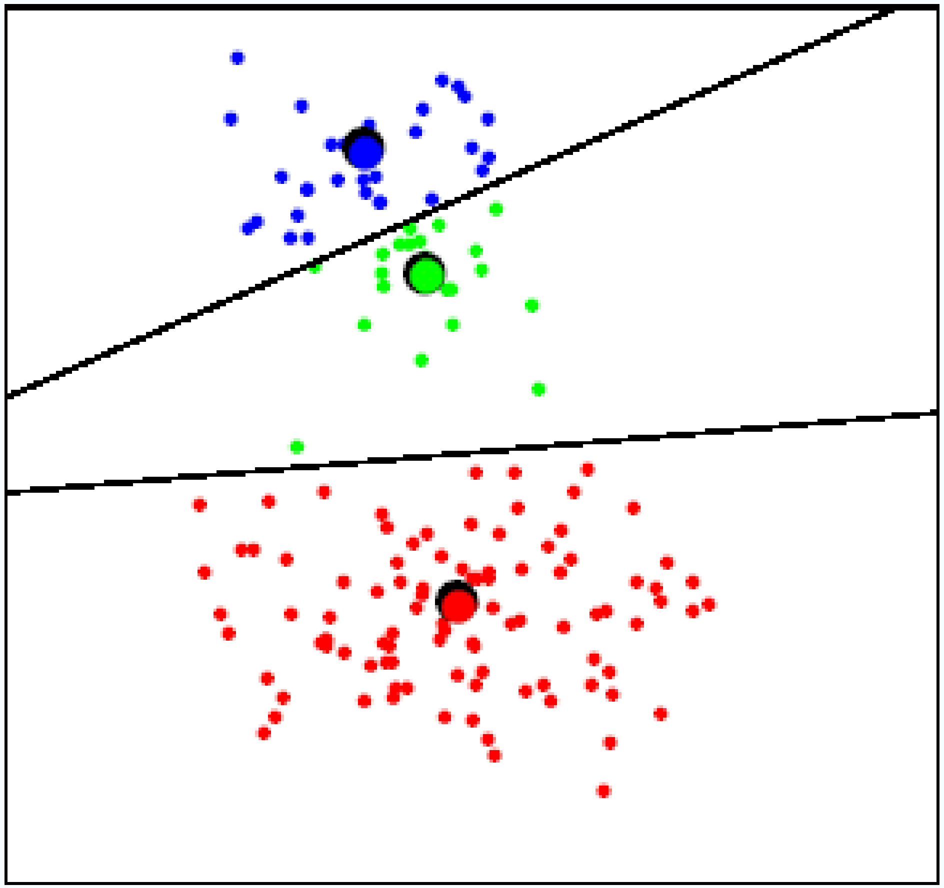
# K-Means: Initial Partition



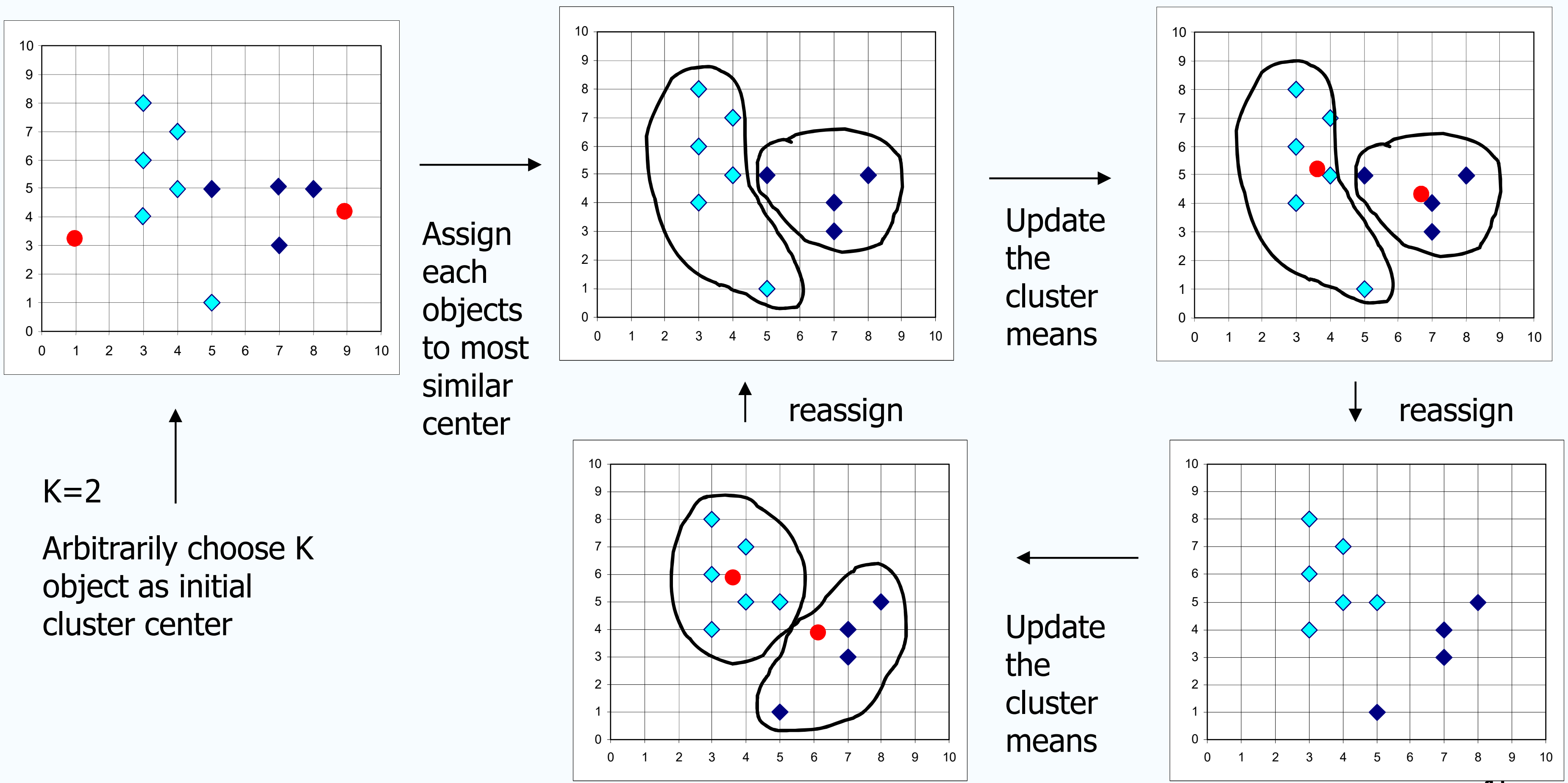
# K-Means: Iteration 2



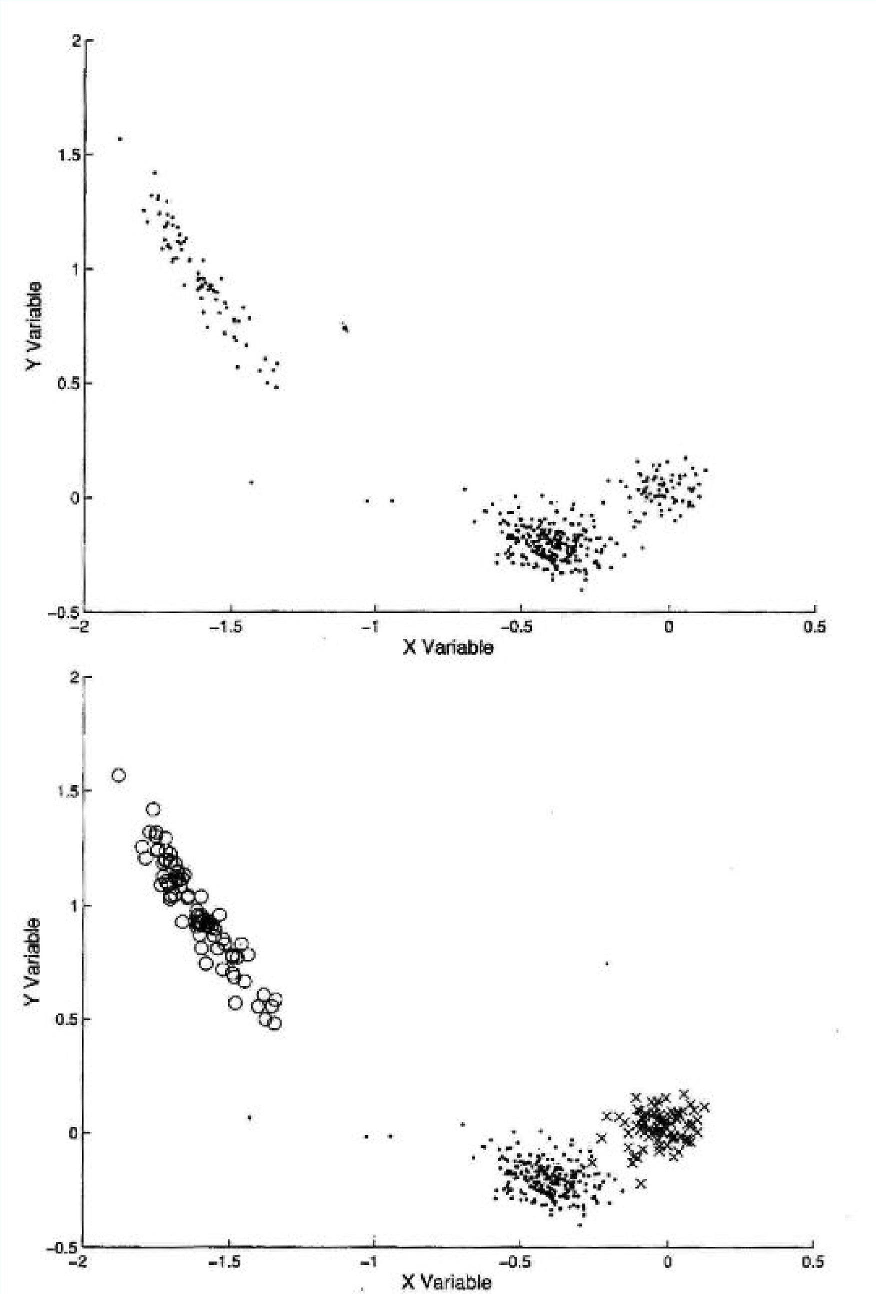
# K-Means: Iteration 20



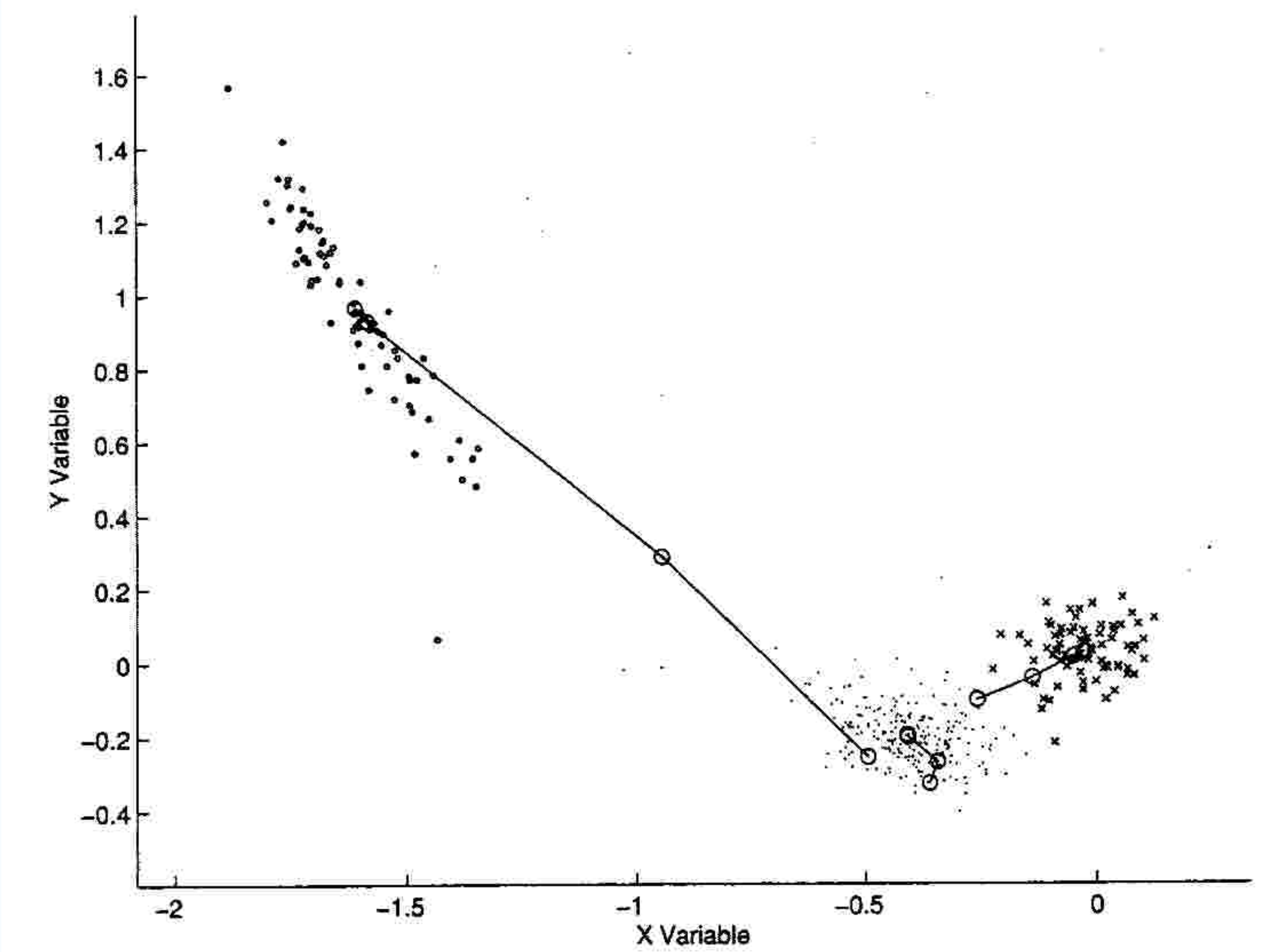
# Another Example: K-Means At Work



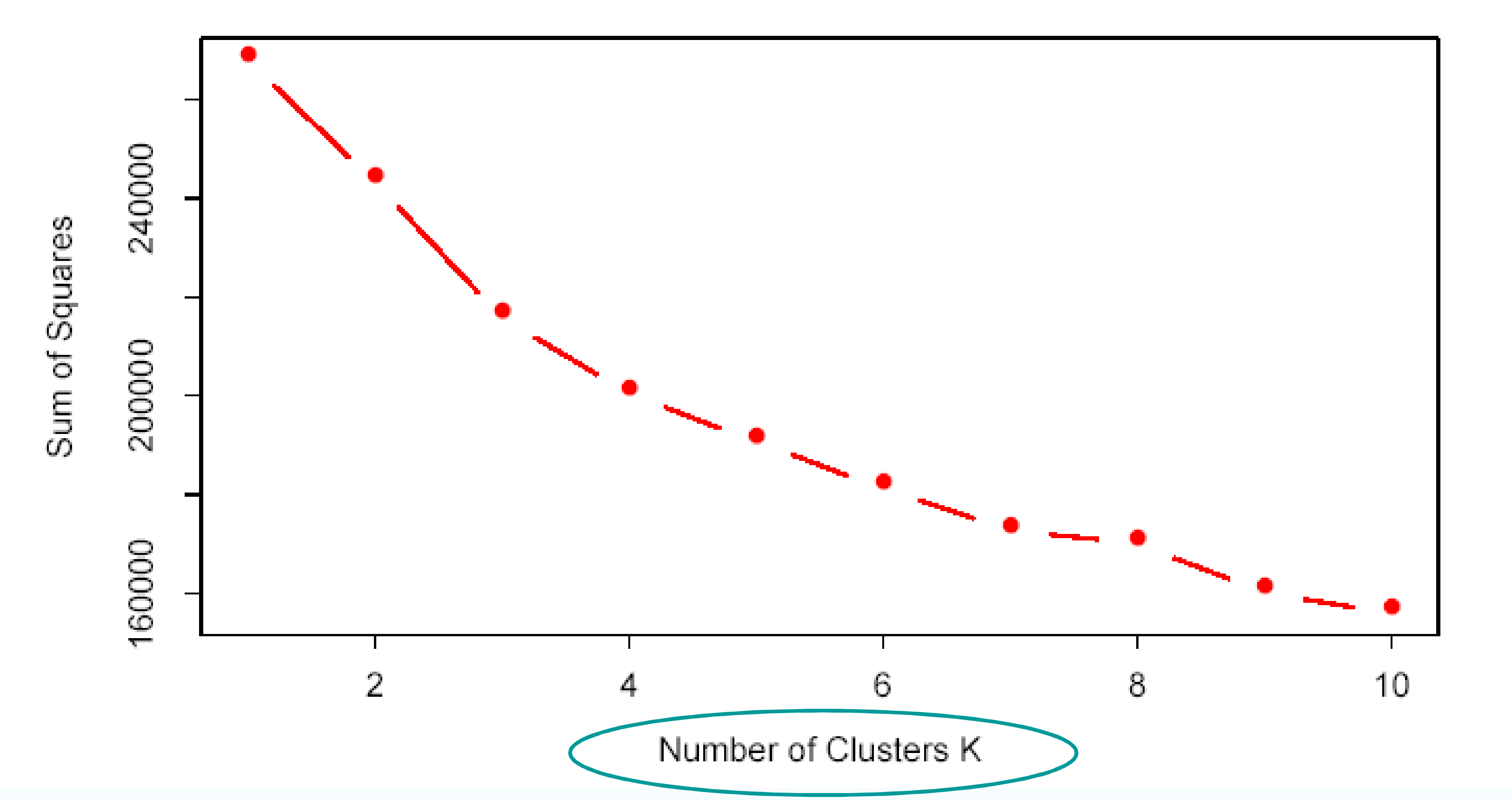
# Antenna Data with/without Class Labels



# Trajectory of Cluster Means on Antenna Data



# K-Means Applied to Microarray Data




Y axis: total within-cluster sum of squares

# K-Means Discussion 1

- Most suitable for real-valued data
- Prefers small, spherical clusters, unable to discover clusters with *non-convex shapes*
- Converges to a local optimum („trapped in local minimum“)
- *Not too many theoretical results on convergence*
- Example for local minimum:
  - four instances at the vertices of a two-dimensional rectangle
  - two cluster centers at the midpoints of the rectangle’s long sides

# K-Means Discussion 2

- Result can vary significantly based on initial choice of seeds
- Simple way to increase chance of finding a global optimum: restarts with different random seeds (“random restarts”)
- Linear in the examples – suitable for *large datasets*
- K has to be specified by the user – extensions as X-Means possible

# Probability-Based/Model-Based Clustering: The EM Algorithm

# Probability-Based Clustering

- Problem with previous approaches:
  - *ad hoc*: what are actually the clusters that are obtained? (“semantics”)
  - instances are “deterministically” assigned to one cluster
- From a probabilistic perspective, we want to find the *most likely clusters given the data*
- Also: instance only has certain *probability of belonging to a particular cluster*

# Finite Mixtures

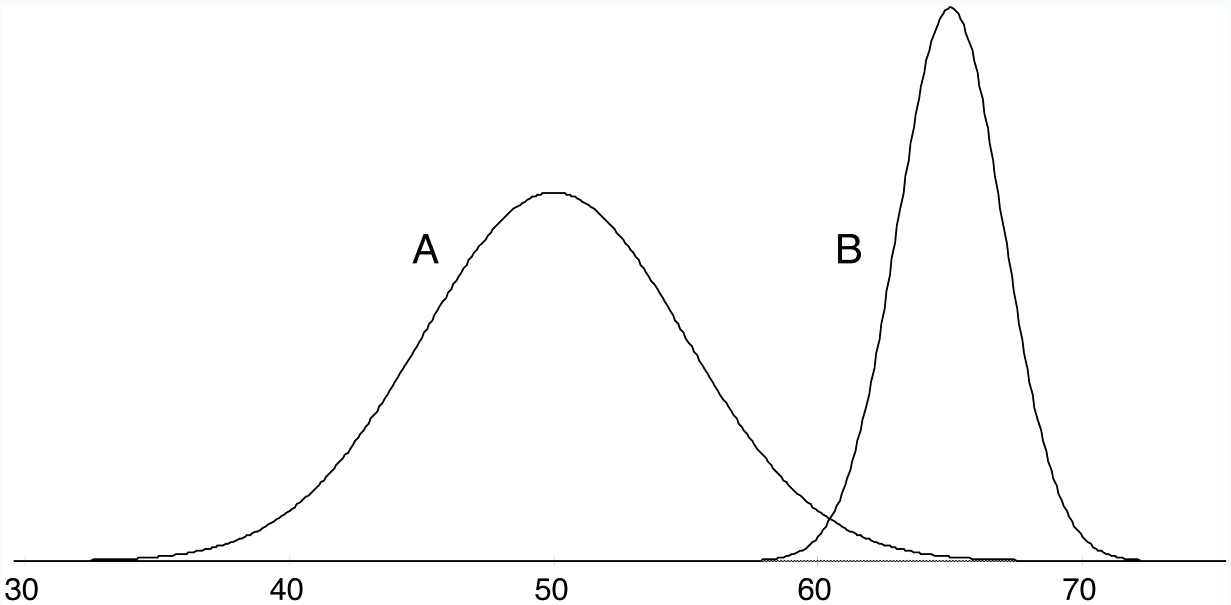
- Probabilistic clustering algorithms model the data as a mixture of distributions
- They are called *finite mixtures* because there is only a finite number of clusters being represented
- *Each cluster* is represented by *one distribution*
  - distribution governing the probabilities of attribute values in the clusters
- Usually individual distributions are normal distributions
- Distributions are combined using cluster weights

# A Two-Class Mixture Model

data

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 51 | B | 62 | B | 64 | A | 48 | A | 39 | A | 51 |
| A | 43 | A | 47 | A | 51 | B | 64 | B | 62 | A | 48 |
| B | 62 | A | 52 | A | 52 | A | 51 | B | 64 | B | 64 |
| B | 64 | B | 64 | B | 62 | B | 63 | A | 52 | A | 42 |
| A | 45 | A | 51 | A | 49 | A | 43 | B | 63 | A | 48 |
| A | 42 | B | 65 | A | 48 | B | 65 | B | 64 | A | 41 |
| A | 46 | A | 48 | B | 62 | B | 66 | A | 48 | | |
| A | 45 | A | 49 | A | 43 | B | 65 | B | 64 | | |
| A | 45 | A | 46 | A | 40 | A | 46 | A | 48 | | |

model




$\mu_A$=50, $\sigma_A$ =5, $p_A$=0.6 $\mu_B$=65, $\sigma_B$ =2, $p_B$=0.4

# *Using* the Mixture Model

- The probability of an instance x belonging to cluster A is:

$$\Pr[A \mid x] = \frac{\Pr[x \mid A]\Pr[A]}{\Pr[x]} = \frac{f(x;\mu_A,\sigma_A)p_A}{\Pr[x]}$$

with $$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- The *likelihood* of a *single instance* x given *all clusters* is:

$$\Pr[x \mid \text{the distributions}] = \sum_i \Pr[x \mid \text{cluster}_i]\Pr[\text{cluster}_i]$$

# Learning the Clusters

- Assume we know that there are k clusters
- To learn the clusters we need to determine their *parameters*, i.e., their means and standard deviations
- We actually have a *performance criterion*: *likelihood of training data* given the clusters
- Fortunately, there exists an algorithm that finds a local maximum of the likelihood

# EM Algorithm for Clustering

- EM algorithm: expectation-maximization algorithm
- In a sense, generalization of k-means to *probabilistic setting*
- Similar iterative procedure:
  1. Calculate cluster probability for each instance (**E**xpectation step)
  2. Estimate distribution parameters based on the cluster probabilities (**M**aximization step)
- Cluster probabilities are stored as instance weights

# More on EM

- Estimating parameters from weighted instances:

$$\mu_A = \frac{w_1 x_1 + w_2 x_2 + \ldots + w_n x_n}{w_1 + w_2 + \ldots + w_n}$$

$$\sigma_A{}^2 = \frac{w_1 (x_1 - \mu)^2 + w_2 (x_2 - \mu)^2 + \ldots + w_n (x_n - \mu)^2}{w_1 + w_2 + \ldots + w_n}$$

# Log-Likelihood

- Procedure stops when *log-likelihood* saturates

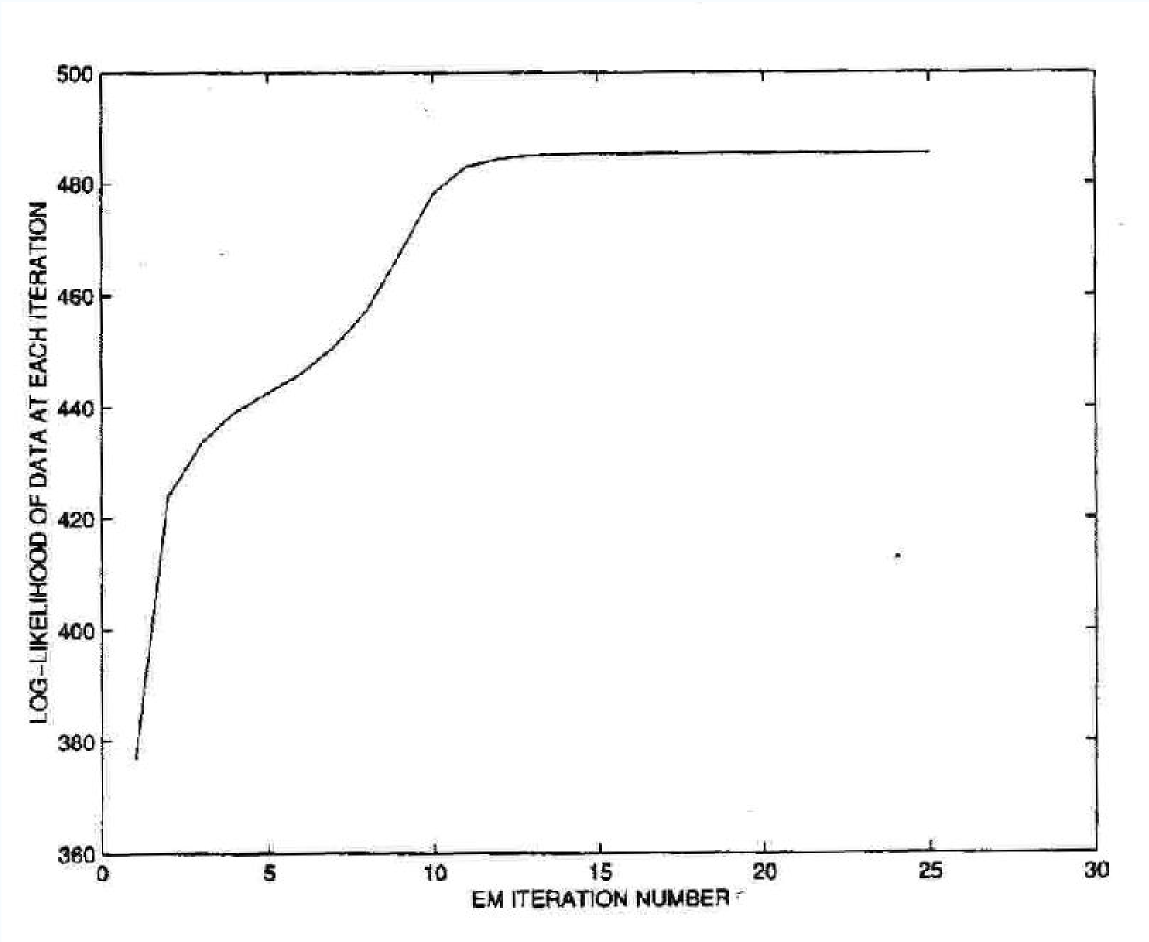
$$\log \prod_{x_i \in D} likelihood(x_i) = \log \prod_{x_i \in D} \sum_{k \in Cluster} p_k P(x_i \mid C_k) =$$

$$\sum_{x_i \in D} \log \sum_{k \in Cluster} p_k P(x_i \mid C_k)$$

- For two clusters A and B:

$$\sum_{x_i \in D} \log(p_A P(x_i \mid A) + p_B P(x_i \mid B))$$

# EM: Log-Likelihood over a Few Iterations



# Special Case of EM: Putting it All Together

**while** log likelihood changes **do**

% calculate cluster probability

% for each instance (E-step)

**for each** instance $x_i$ **do**

$w_i = f(x_i; \mu_A, \sigma_A)p_A / P[x_i]$

% estimate distribution parameters based

% on the cluster probabilities (M-step)

$\mu_A = \Sigma w_i x_i / \Sigma w_i$

$\sigma_A^2 = \Sigma w_i(x_i - \mu_A)^2 / \Sigma w_i$

% analogously for $\mu_B$ and $\sigma_B^2$ with $(1-w_i)$

# Special Case of EM: Putting it All Together

**while** log likelihood change

% calculate cluster proba

% for each instance (E-ste

**for each** instance $x_i$ **do**

$w_i = f(x_i; \mu_A, \sigma_A) p_A / P[x_i]$

% estimate distribution parameters based

% on the cluster probabilities (M-step)

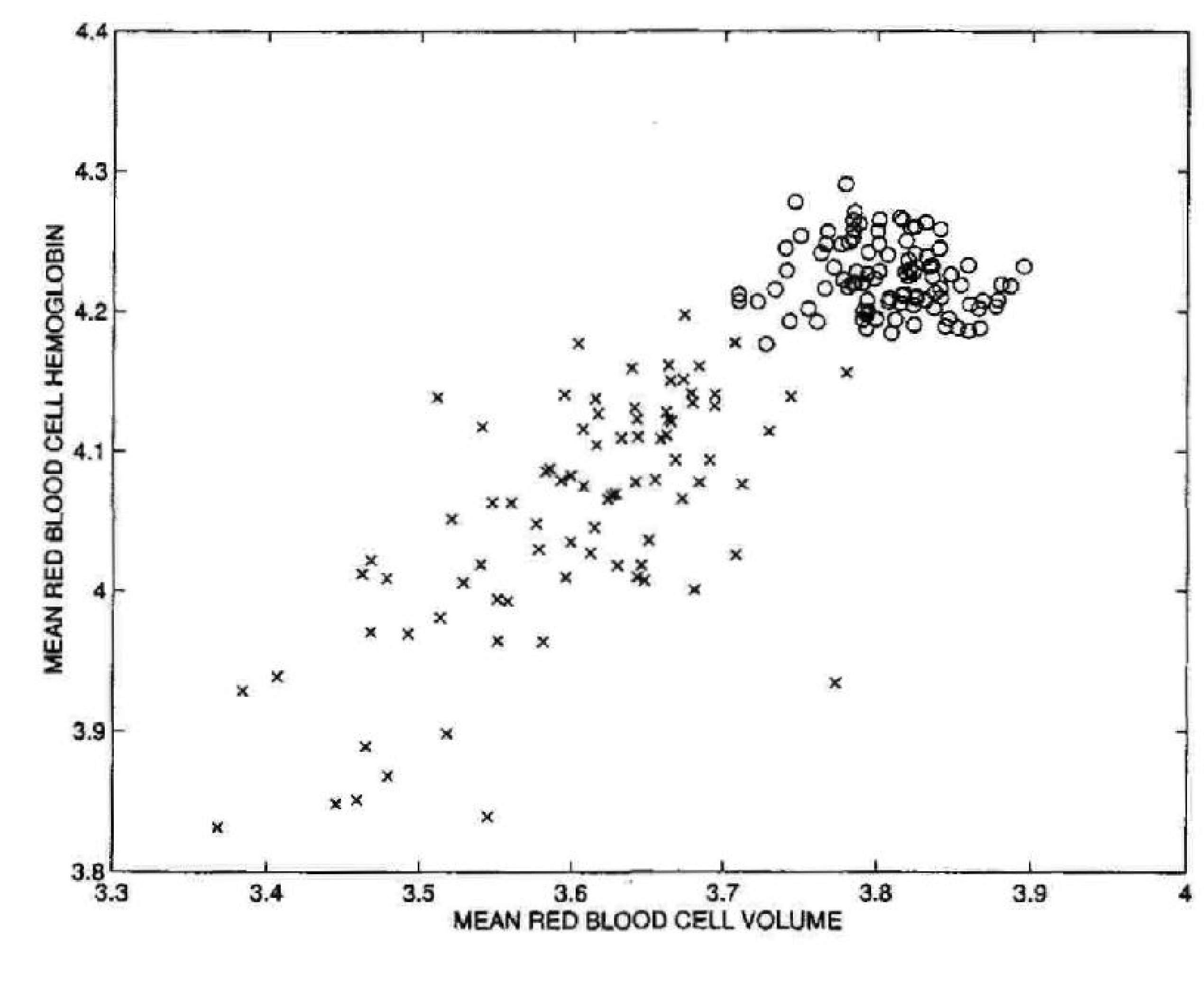
$\mu_A = \Sigma w_i x_i / \Sigma w_i$

$\sigma_A^2 = \Sigma w_i (x_i - \mu_A)^2 / \Sigma w_i$

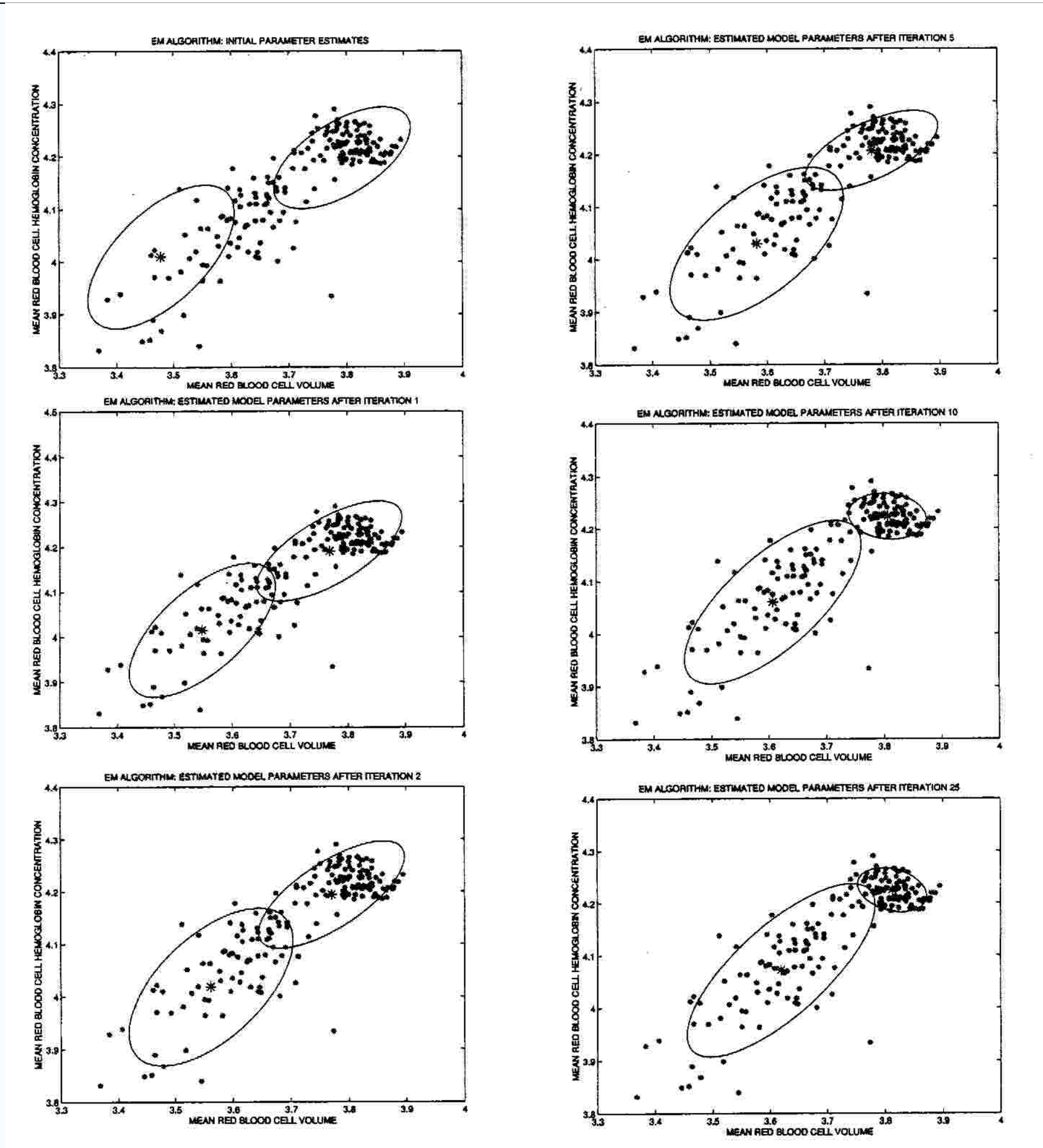
% analogously for $\mu_B$ and $\sigma_B^2$ with $(1-w_i)$

**for** $k = 1, \ldots, K$ let $\mathbf{r}(k)$ be a randomly chosen point from $D$;
**while** changes in clusters $C_k$ happen **do**
form clusters:
**for** $k = 1, \ldots, K$ **do**
$C_k = \{\mathbf{x} \in D \mid d(\mathbf{r}_k, \mathbf{x}) \leq d(\mathbf{r}_j, \mathbf{x})$ for all $j = 1, \ldots, K, j \neq k\}$;
**end**;
compute new cluster centers:
**for** $k = 1, \ldots, K$ **do**
$\mathbf{r}_k$ = the vector mean of the points in $C_k$
**end**;
**end**;

# Red Blood Cell Data




Circles: healthy, crosses: iron deficient anemia




# Example Run EM on Red Blood Cell Data

$3\sigma$ covariance ellipses and means for iterations 1, 2, 5, 10, 25

# Extensions 1

- Using more than two distributions: easy
- Several attributes: easy if independence is assumed
- Correlated attributes: difficult
  - Modeled jointly using a bivariate normal distribution with a covariance matrix
  - With $n$ attributes this requires estimating $n + n(n+1)/2$ parameters
- Nominal attributes: easy if independent

# Extensions 2

- Correlated nominal attributes: difficult
  - two correlated attributes result in v1 x v2 parameters
- Missing values: easy
- Distributions other than the normal distribution can be used:
  - log-normal if predetermined minimum is given
  - log-odds if bounded from above and below
  - Poisson for attributes that are integer counts
- Cross-validation can be used to estimate k

# Discussion

- Can be used to fill in missing values
- Big advantage of probabilistic clustering schemes: likelihood of data can be estimated and used to compare different clusterings