The 3rd IIA-PennState Astrostatistics School

19-27 July 2010

# Cluster Analysis

**Jia Li**

Department of Statistics
Penn State University

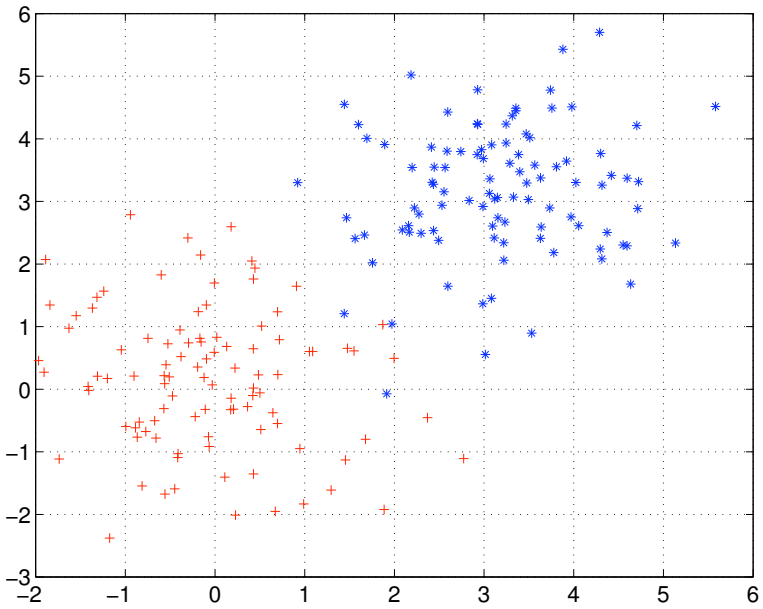
Notes revised and lectures delivered by

**T.Krishnan**

Strand Life Sciences, Bangalore

# Clustering

- A basic tool in data mining/pattern recognition:
  - Divide a set of data into groups.
  - Samples in one cluster are close and clusters are far apart.




- Motivations:
  - Discover classes of data in an unsupervised way (unsupervised learning).
  - Efficient representation of data: fast retrieval, data complexity reduction.
  - Various engineering purposes: tightly linked with pattern recognition.

# Approaches to Clustering

- Represent samples by feature vectors.
- Define a distance measure to assess the closeness between data.
- "Closeness" can be measured in many ways.
  - Define distance based on various norms.
  - For stars with measured parallax, the multivariate "distance" between stars is the spatial Euclidean distance. For a galaxy redshift survey, however, the multivariate "distance" depends on the Hubble constant which scales velocity to spatial distance. For many astronomical datasets, the variables have incompatible units and no prior known relationship. The result of clustering will depends on the arbitrary choice of variable scaling.

# Approaches to Clustering

- Clustering: grouping of similar objects (unsupervised learning)
- Approaches
  - Prototype methods:
    - K-means (for vectors)
    - K-center (for vectors)
    - D2-clustering (for bags of weighted vectors)
  - Statistical modeling
    - Mixture modeling by the EM algorithm
    - Modal clustering
  - Pairwise distance based partition:
    - Spectral graph partitioning
    - Dendrogram clustering (agglomerative): single linkage (friends of friends algorithm), complete linkage, etc.

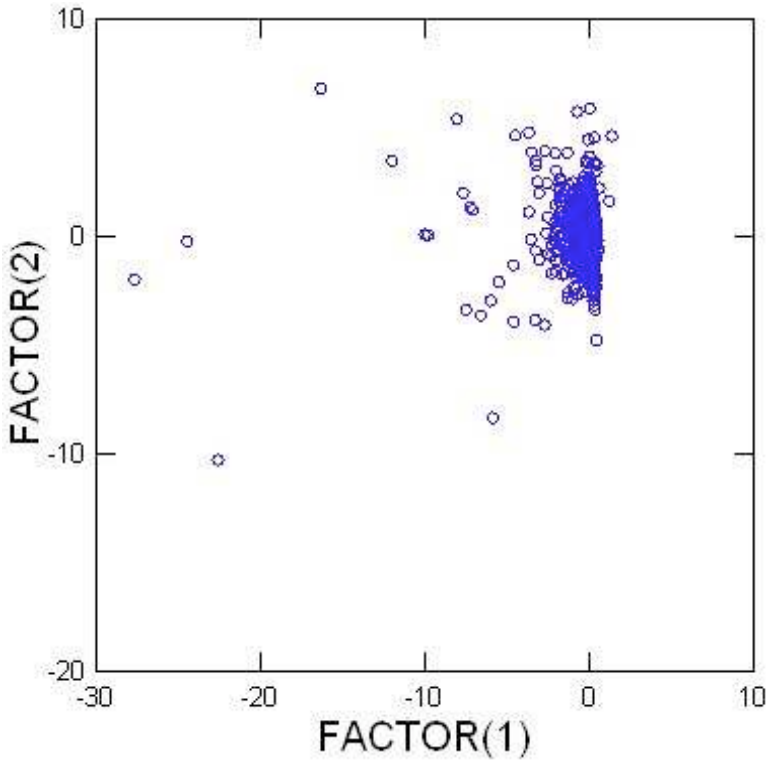
# Agglomerative Clustering

- Generate clusters in a hierarchical way.
- Let the data set be $A = \{x_1, ..., x_n\}$.
- Start with $n$ clusters, each containing one data point.
- Merge the two clusters with minimum pairwise distance.
- Update between-cluster distance.
- Iterate the merging procedure.
- The clustering procedure can be visualized by a tree structure called *dendrogram*.
- Definition for between-cluster distance?
  - For clusters containing only one data point, the between-cluster distance is the between-object distance.
  - For clusters containing multiple data points, the between-cluster distance is an agglomerative version of the between-object distances.
    - Examples: minimum or maximum between-objects distances for objects in the two clusters.
  - The agglomerative between-cluster distance can often be computed recursively.

# Principal Components Clustering

If the several dimensions can be satisfactorily reduced to two, by say Principal Components, then plotting the two component scores for each object will result in a picture which will help find clusters.

Combo17 Example:




This figure suggests one strong dense cluster and the remaining perhaps forming another dissipated one.

# K-means

- Assume there are $M$ prototypes denoted by

$$\mathcal{Z} = \{z_1, z_2, ..., z_M\} .$$

- Each training sample is assigned to one of the prototype. Denote the assignment function by $A(\cdot)$. Then $A(x_i) = j$ means the $i$th training sample is assigned to the $j$th prototype.

- Goal: minimize the total mean squared error between the training samples and their representative prototypes, that is, the trace of the pooled within cluster covariance matrix.

$$\arg\min_{\mathcal{Z},A} \sum_{i=1}^{N} \| x_i - z_{A(x_i)} \|^2$$

- Denote the objective function by

$$L(\mathcal{Z}, A) = \sum_{i=1}^{N} \| x_i - z_{A(x_i)} \|^2 \ .$$

- Intuition: training samples are tightly clustered around the prototypes. Hence, the prototypes serve as a compact representation for the training data.

# Necessary Conditions

- If $\mathcal{Z}$ is fixed, the optimal assignment function $A(\cdot)$ should follow the nearest neighbor rule, that is,
$$A(x_i) = \arg\min_{j \in \{1,2,\ldots,M\}} \| x_i - z_j \| \ .$$
- If $A(\cdot)$ is fixed, the prototype $z_j$ should be the average (centroid) of all the samples assigned to the $j$th prototype:
$$z_j = \frac{\sum_{i:A(x_i)=j} x_i}{N_j} \ ,$$
where $N_j$ is the number of samples assigned to prototype $j$.

# The Algorithm

- Based on the necessary conditions, the k-means algorithm alternates the two steps:
  - For a fixed set of centroids (prototypes), optimize $A(\cdot)$ by assigning each sample to its closest centroid using Euclidean distance.
  - Update the centroids by computing the average of all the samples assigned to it.
- The algorithm converges since after each iteration, the objective function decreases (non-increasing).
- Usually converges fast.
- Stopping criterion: the ratio between the decrease and the objective function is below a threshold.

# Example

- Training set: $\{1.2, 5.6, 3.7, 0.6, 0.1, 2.6\}$.
- Apply k-means algorithm with 2 centroids, $\{z_1, z_2\}$.
- Initialization: randomly pick $z_1 = 2$, $z_2 = 5$.

| fixed | update |
|---|---|
| 2 | {1.2, 0.6, 0.1, 2.6} |
| 5 | {5.6, 3.7} |
| {1.2, 0.6, 0.1, 2.6} | 1.125 |
| {5.6, 3.7} | 4.65 |
| 1.125 | {1.2, 0.6, 0.1, 2.6} |
| 4.65 | {5.6, 3.7} |

The two prototypes are: $z_1 = 1.125$, $z_2 = 4.65$. The objective function is $L(\mathcal{Z}, A) = 5.3125$.

- Initialization: randomly pick $z_1 = 0.8$, $z_2 = 3.8$.

| fixed | update |
|---|---|
| 0.8 | {1.2, 0.6, 0.1} |
| 3.8 | {5.6, 3.7, 2.6} |
| {1.2, 0.6, 0.1 } | 0.633 |
| {5.6, 3.7, 2.6 } | 3.967 |
| 0.633 | {1.2, 0.6, 0.1} |
| 3.967 | {5.6, 3.7, 2.6} |

  The two prototypes are: $z_1 = 0.633$, $z_2 = 3.967$. The objective function is $L(\mathcal{Z}, A) = 5.2133$.

- Starting from different initial values, the k-means algorithm converges to different local optimum.

- It can be shown that $\{z_1 = 0.633, z_2 = 3.967\}$ is the global optimal solution.

# Initialization

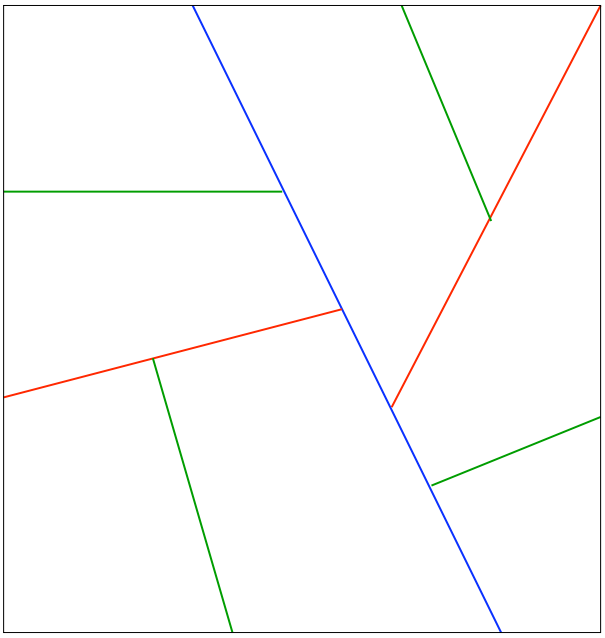
- Randomly pick up the prototypes to start the k-means iteration.
- Different initial prototypes may lead to different local optimal solutions given by k-means.
- Try different sets of initial prototypes, compare the objective function at the end to choose the best solution.
- When randomly select initial prototypes, better make sure no prototype is out of the range of the entire data set.
- Initialization in the above simulation:
  - Generated $M$ random vectors with independent dimensions. For each dimension, the feature is uniformly distributed in $[-1, 1]$.
  - Linearly transform the $j$th feature, $Z_j$, $j = 1, 2, ..., p$ in each prototype (a vector) by: $Z_j s_j + m_j$, where $s_j$ is the sample standard deviation of dimension $j$ and $m_j$ is the sample mean of dimension $j$, both computed using the training data.

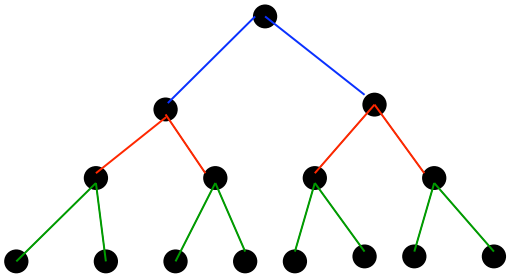
# Linde-Buzo-Gray (LBG) Algorithm

- An algorithm developed in vector quantization for the purpose of data compression.
- Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Communication*, Vol. COM-28, pp. 84-95, Jan. 1980.
- The algorithm
  1. Find the centroid $z_1^{(1)}$ of the entire data set.
  2. Set $k = 1$, $l = 1$.
  3. If $k < M$, split the current centroids by adding small offsets.
     - If $M - k \geq k$, split all the centroids; otherwise, split only $M - k$ of them.
     - Denote the number of centroids split by $\tilde{k} = \min(k, M - k)$.
     - For example, to split $z_1^{(1)}$ into two centroids, let $z_1^{(2)} = z_1^{(1)}$, $z_2^{(2)} = z_1^{(1)} + \epsilon$, where $\epsilon$ has a small norm and a random direction.
  4. $k \leftarrow k + \tilde{k}$; $\quad l \leftarrow l + 1$.
  5. Use $\{z_1^{(l)}, z_2^{(l)}, ..., z_k^{(l)}\}$ as initial prototypes. Apply k-means iteration to update these prototypes.
  6. If $k < M$, go back to step 3; otherwise, stop.

# Tree-structured Clustering

- Studied extensively in vector quantization from the perspective of data compression.
- Referred to as tree-structured vector quantization (TSVQ).
- The algorithm
  1. Apply 2 centroids k-means to the entire data set.
  2. The data are assigned to the 2 centroids.
  3. For the data assigned to each centroid, apply 2 centroids k-means to them separately.
  4. Repeat the above step.





- Compare with LBG:
  - For LBG, after the initial prototypes are formed by splitting, k-means is applied to the overall data set. The final result is $M$ prototypes.
  - For TSVQ, data partitioned into different centroids at the same level will never affect each other in the future growth of the tree. The final result is a tree structure.
- Fast searching
  - For k-means, to decide which cell a query $x$ goes to, blue$M$ (the number of prototypes) distances need to be computed.
  - For the tree-structured clustering, to decide which cell a query $x$ goes to, only blue$2\log_2(M)$ distances need to be computed.
- Comments on tree-structured clustering:
  - It is structurally more constrained. But on the other hand, it provides more insight into the patterns in the data.
  - It is greedy in the sense of optimizing at each step sequentially. An early bad decision will propagate its effect.
  - It provides more algorithmic flexibility.

# Example Distances

- Suppose cluster $r$ and $s$ are two clusters merged into a new cluster $t$. Let $k$ be any other cluster.
- Denote between-cluster distance by $D(\cdot, \cdot)$.
- How to get $D(t, k)$ from $D(r, k)$ and $D(s, k)$?
  - *Single-link clustering*:

    $$D(t, k) = \min(D(r, k), D(s, k))$$

    $D(t, k)$ is the *minimum* distance between two objects in cluster $t$ and $k$ respectively.
  - *Complete-link clustering*:

    $$D(t, k) = \max(D(r, k), D(s, k))$$

    $D(t, k)$ is the *maximum* distance between two objects in cluster $t$ and $k$ respectively.
  - *Average linkage clustering*:
    Unweighted case:

    $$D(t, k) = \frac{n_r}{n_r + n_s} D(r, k) + \frac{n_s}{n_r + n_s} D(s, k)$$

    Weighted case:

    $$D(t, k) = \frac{1}{2} D(r, k) + \frac{1}{2} D(s, k)$$

    $D(t, k)$ is the average distance between two objects in cluster $t$ and $k$ respectively. For the unweighted case, the number of elements in each cluster is taken into consideration, while in the weighted case each cluster is weighted equally. So objects in smaller cluster are weighted more heavily than those in larger clusters.
  - *Centroid clustering:*
    Unweighted case:

    $$\begin{aligned} D(t, k) &= \frac{n_r}{n_r + n_s} D(r, k) + \frac{n_s}{n_r + n_s} D(s, k) \\ &\quad - \frac{n_r n_s}{n_r + n_s} D(r, s) \end{aligned}$$

    Weighted case:

    $$D(t, k) = \frac{1}{2} D(r, k) + \frac{1}{2} D(s, k) - \frac{1}{4} D(r, s)$$

    A centroid is computed for each cluster and the distance between clusters is given by the distance between their respective centroids.

– *Ward's clustering:*

$$\begin{aligned} D(t,k) &= \frac{n_r + n_k}{n_r + n_s + n_k} D(r,k) \\ &+ \frac{n_s + n_k}{n_r + n_s + n_k} D(s,k) \\ &- \frac{n_k}{n_r + n_s + n_k} D(r,s) \end{aligned}$$

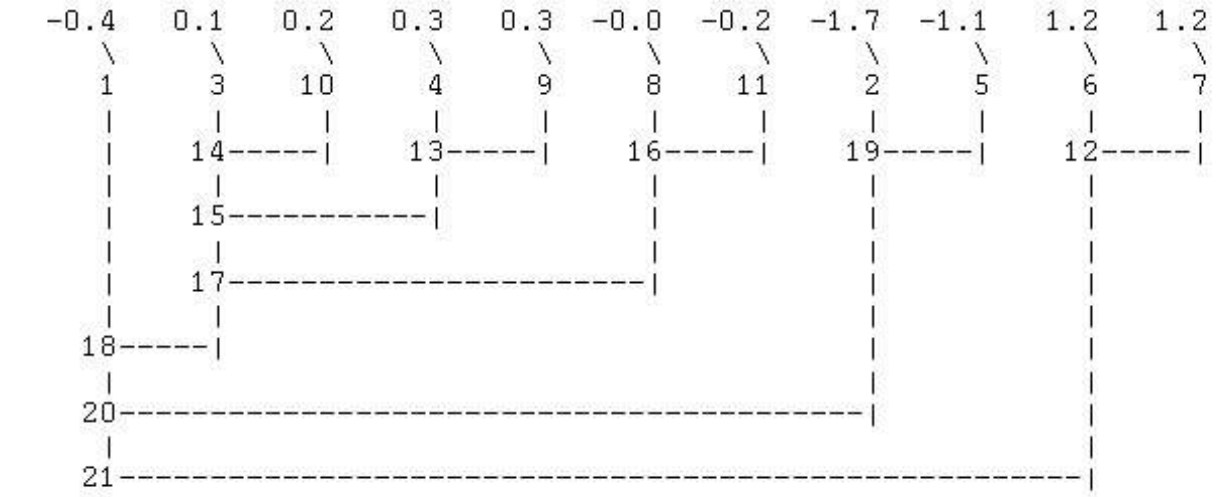
Merge the two clusters for which the change in the variance of the clustering is minimized. The variance of a cluster is defined as the sum of squared-error between each object in the cluster and the centroid of the cluster.

- The dendrogram generated by single-link clustering tends to look like a chain. Clusters generated by complete-link may not be well separated. Other methods are intermediates between the two.

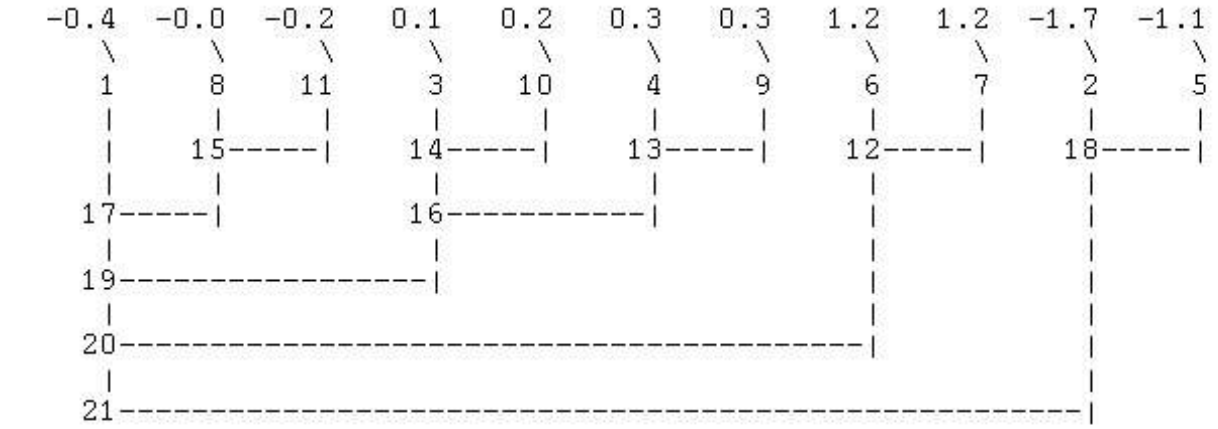
# Pseudo Code

1. Begin with $n$ clusters, each containing one object. Number the clusters 1 through $n$.
2. Compute the between-cluster distance $D(r, s)$ as the between-object distance of the two objects in $r$ and $s$ respectively, $r, s = 1, 2, ..., n$. Let square matrix $D = (D(r, s))$.
3. Find the most similar pair of clusters $r$, $s$, that is, $D(r, s)$ is minimum among all the pairwise distances.
4. Merge $r$ and $s$ to a new cluster $t$. Compute the between-cluster distance $D(t, k)$ for all $k \neq r, s$. Delete the rows and columns corresponding to $r$ and $s$ in $D$. Add a new row and column in D corresponding to cluster $t$.
5. Repeat Step 3 a total of $n - 1$ times until there is only one cluster left.

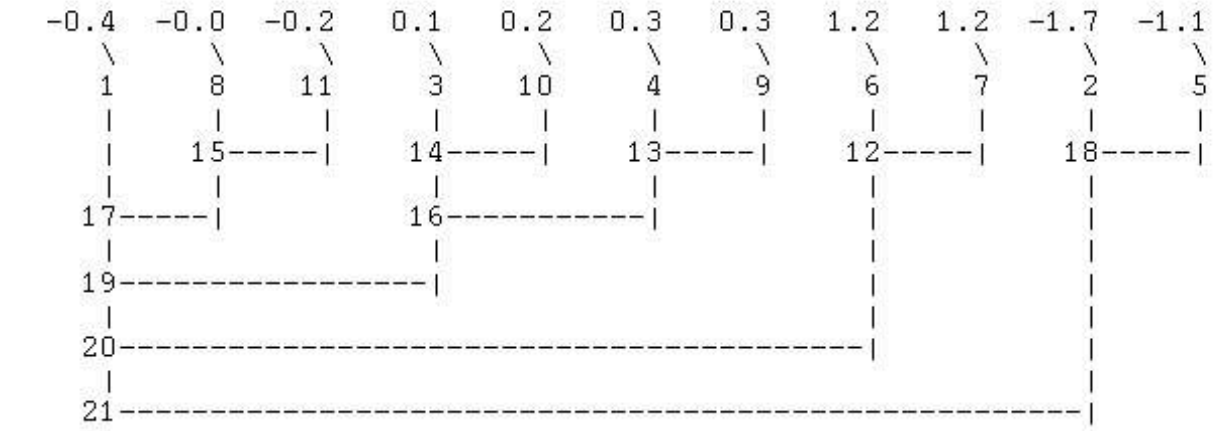
```
Data set (11 points):
-0.432565,  -0.037633,  -0.186709,  0.125332,  0.174639
0.287676,   0.327292,   1.190915,   1.189164,  -1.665584,  -1.146471

Single-link clustering:
```




```
Complete-link clustering:
```




```
Ward's clustering:
```



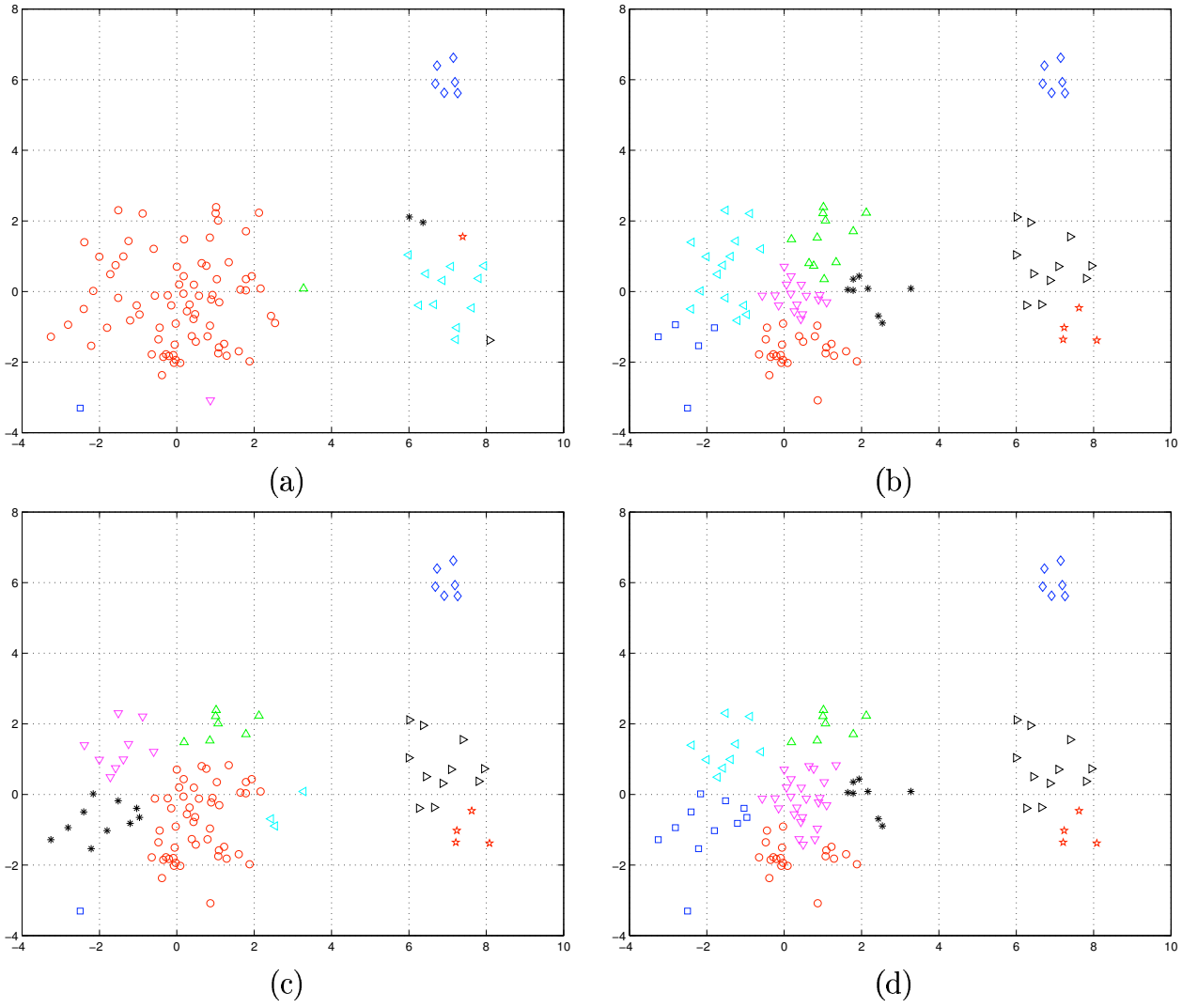
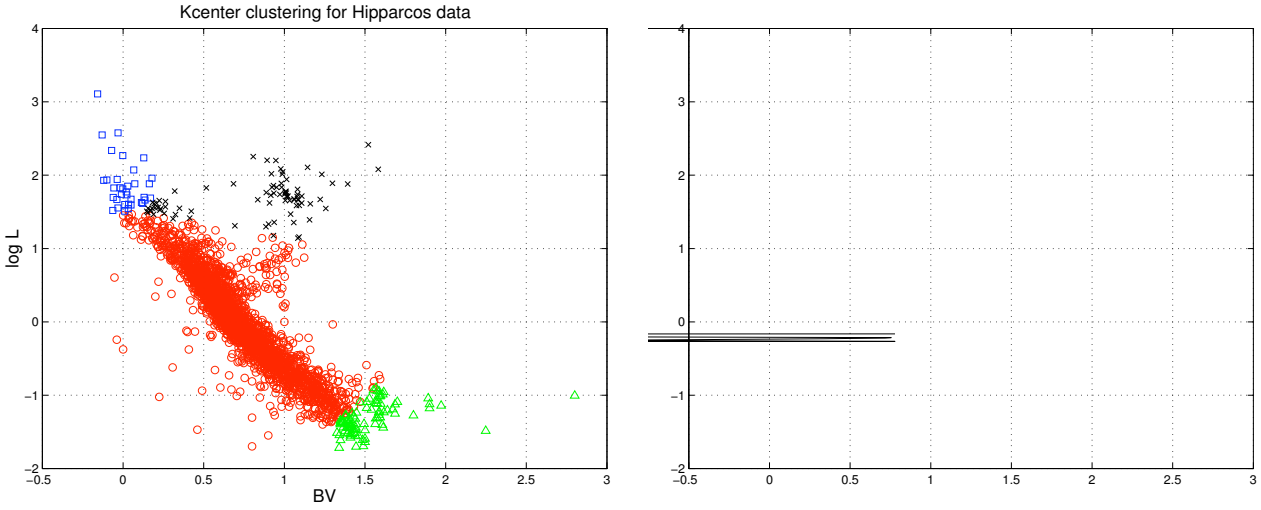
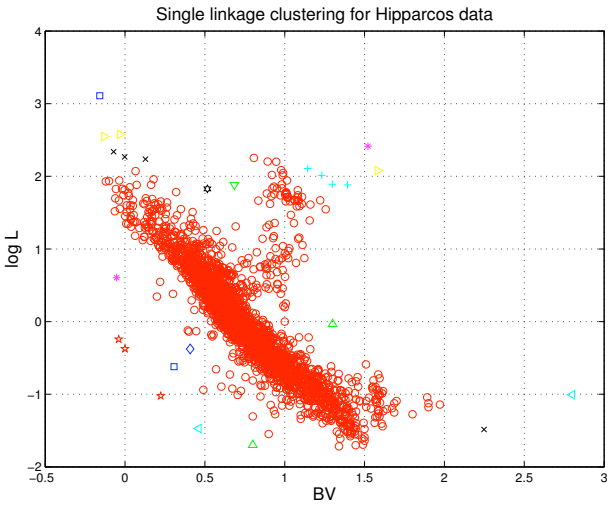



Figure 1: Agglomerate clustering of a data set (100 points) into 9 clusters. (a): Single-link, (b): Complete-link, (c): Average linkage, (d) Wards clustering

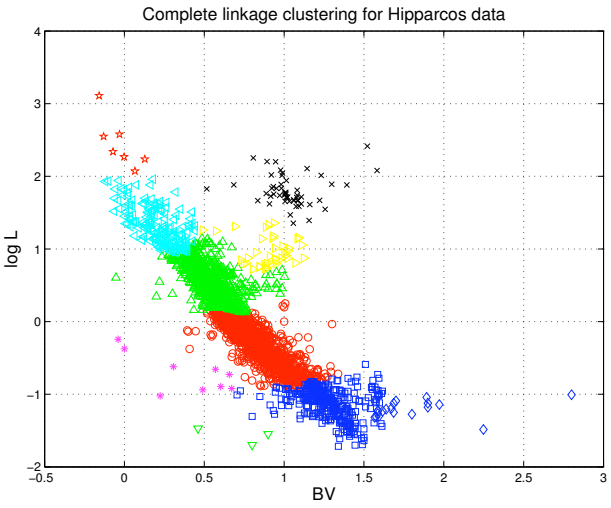
# Hipparcos Data

- Clustering based on $\log L$ and $BV$.

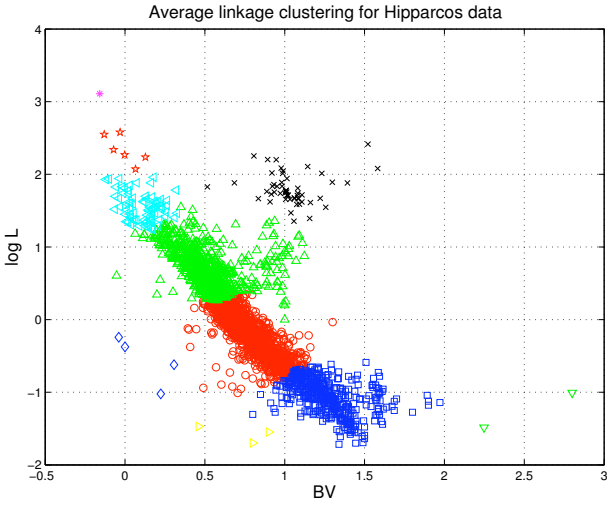




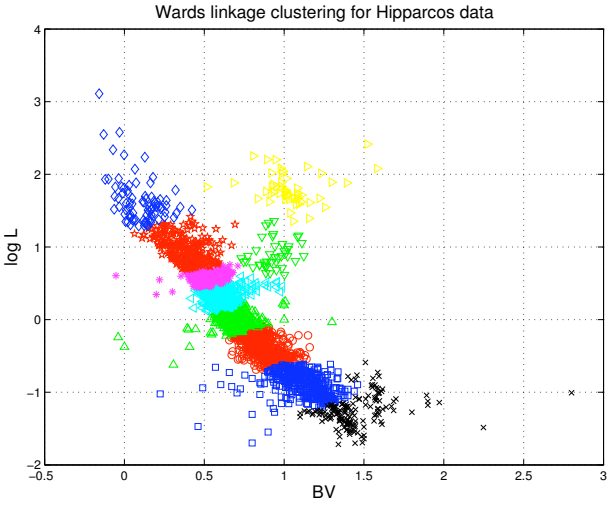

(a) Single linkage #clusters=20




(b) Complete linkage #clusters=10

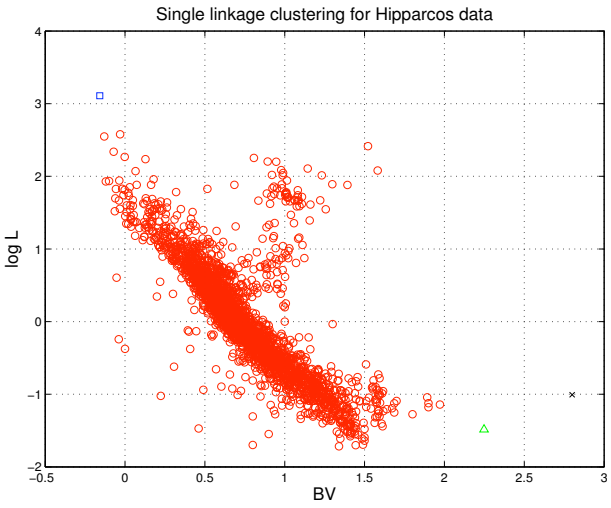



(c) Average linkage #clusters=10

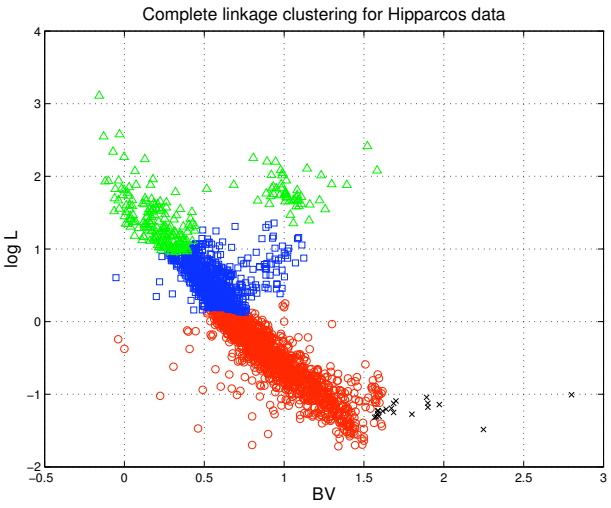



(d) Ward's linkage #clusters=10

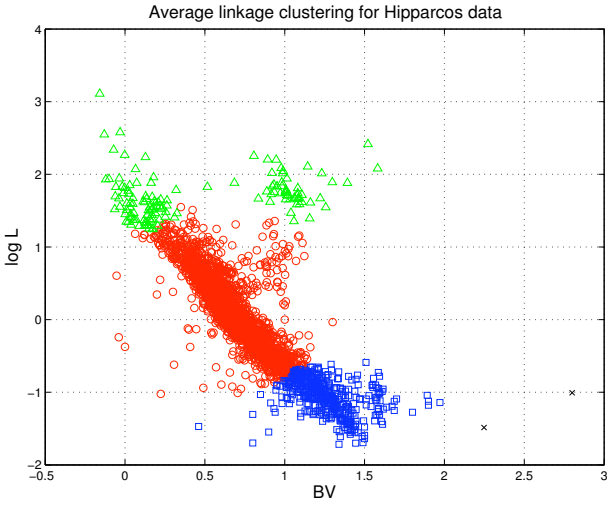
Figure 3: Clustering of the Hipparcos data



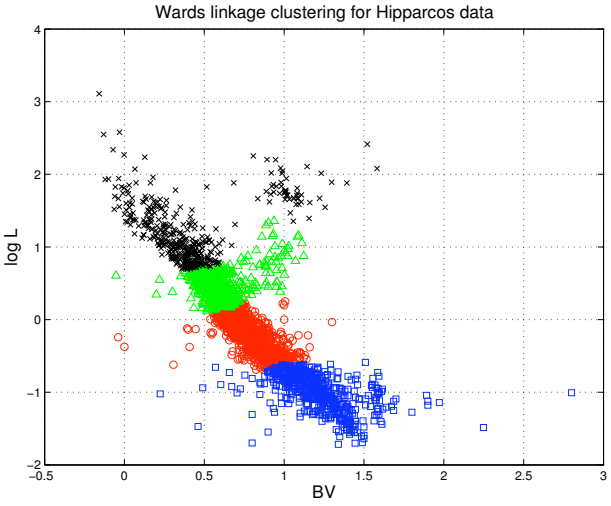

(a) Single linkage #clusters=4




(b) Complete linkage #clusters=4




(c) Average linkage #clusters=4




(d) Ward's linkage #clusters=4

Figure 4: Clustering of the Hipparcos data

# Mixture Model-based Clustering

- Each cluster is mathematically represented by a parametric distribution. Examples: Gaussian (continuous), Poisson (discrete).
- The entire data set is modeled by a mixture of these distributions.
- An individual distribution used to model a specific cluster is often referred to as a component distribution.
- Suppose there are $K$ components (clusters). Each component is a Gaussian distribution parameterized by $\mu_k$, $\Sigma_k$. Denote the data by $X$, $X \in \mathcal{R}^d$. The density of component $k$ is

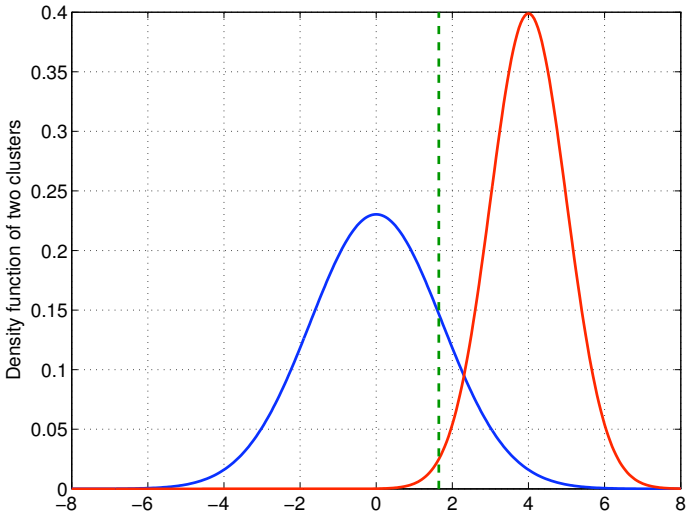
$$\begin{aligned} f_k(x) &= \phi(x \mid \mu_k, \Sigma_k) \\ &= \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} \exp\left(\frac{-(x-\mu_k)^t \Sigma_k^{-1} (x-\mu_k)}{2}\right) . \end{aligned}$$

- The prior probability (weight) of component $k$ is $a_k$. The mixture density is:

$$f(x) = \sum_{k=1}^{K} a_k f_k(x) = \sum_{k=1}^{K} a_k \phi(x \mid \mu_k, \Sigma_k) .$$

# Advantages

- A mixture model with high likelihood tends to have the following traits:
  - Component distributions have high "peaks" (bluedata in one cluster are tight)
  - The mixture model "covers" the data well (bluedominant patterns in the data are captured by component distributions).
- blueAdvantages
  - Well-studied statistical inference techniques available.
  - Flexibility in choosing the component distributions.
  - Obtain a density estimation for each cluster.
  - A "soft" classification is available.



# EM Algorithm

- The parameters are estimated by the maximum likelihood (ML) criterion using the EM algorithm.
- The EM algorithm provides an iterative computation of maximum likelihood estimation when the observed data are incomplete.
- Incompleteness can be conceptual.
  - We need to estimate the distribution of $X$, in sample space $\mathcal{X}$, but we can only observe $X$ indirectly through $Y$, in sample space $\mathcal{Y}$.
  - In many cases, there is a mapping $x \to y(x)$ from $\mathcal{X}$ to $\mathcal{Y}$, and $x$ is only known to lie in a subset of $\mathcal{X}$, denoted by $\mathcal{X}(y)$, which is determined by the equation $y = y(x)$.
  - The distribution of $X$ is parameterized by a family of distributions $f(x \mid \theta)$, with parameters $\theta \in \Omega$, on $x$. The distribution of $y$, $g(y \mid \theta)$ is

$$g(y \mid \theta) = \int_{\mathcal{X}(y)} f(\mathbf{x} \mid \theta) dx \ .$$

- The EM algorithm aims at finding a $\theta$ that maximizes $g(y \mid \theta)$ given an observed $y$.
- Introduce the function

$$Q(\theta' \mid \theta) = E(\log f(x \mid \theta') \mid y, \theta) \ ,$$

  that is, the expected value of $\log f(x \mid \theta')$ according to the conditional distribution of $x$ given $y$ and parameter $\theta$. The expectation is assumed to exist for all pairs $(\theta', \theta)$. In particular, it is assumed that $f(x \mid \theta) > 0$ for $\theta \in \Omega$.
- blue**EM Iteration**:
  - E-step: Compute $Q(\theta \mid \theta^{(p)})$.
  - M-step: Choose $\theta^{(p+1)}$ to be a value of $\theta \in \Omega$ that maximizes $Q(\theta \mid \theta^{(p)})$.

# EM for the Mixture of Normals

- Observed data (incomplete): $\{x_1, x_2, ..., x_n\}$, where $n$ is the sample size. Denote all the samples collectively by $\mathbf{x}$.
- Complete data: $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where $y_i$ is the cluster (component) identity of sample $x_i$.
- The collection of parameters, $\theta$, includes: $a_k$, $\mu_k$, $\Sigma_k$, $k = 1, 2, ..., K$.
- The likelihood function is:

$$L(\mathbf{x}|\theta) = \sum_{i=1}^{n} \log \left( \sum_{k=1}^{K} a_k \phi(x_i|\mu_k, \Sigma_k) \right) .$$

- $L(\mathbf{x}|\theta)$ is the objective function of the EM algorithm (maximize). Numerical difficulty comes from the sum inside the log.

- The $Q$ function is:

$$\begin{aligned} Q(\theta'|\theta) &= E\left[\log \prod_{i=1}^{n} a'_{y_i}\phi(x_i \mid \mu'_{y_i}, \Sigma'_{y_i}) \mid \mathbf{x}, \theta\right] \\ &= E\left[\sum_{i=1}^{n} \left(\log(a'_{y_i}) + \log \phi(x_i \mid \mu'_{y_i}, \Sigma'_{y_i}\right) \mid \mathbf{x}, \theta\right] \\ &= \sum_{i=1}^{n} E\left[\log(a'_{y_i}) + \log \phi(x_i \mid \mu'_{y_i}, \Sigma'_{y_i}) \mid x_i, \theta\right] \ . \end{aligned}$$

  The last equality comes from the fact the samples are independent.

- Note that when $x_i$ is given, only $y_i$ is random in the complete data $(x_i, y_i)$. Also $y_i$ only takes a finite number of values, i.e, cluster identities 1 to $K$. The distribution of $Y$ given $X = x_i$ is the posterior probability of $Y$ given $X$.

- Denote the posterior probabilities of $Y = k$, $k = 1, ..., K$ given $x_i$ by $p_{i,k}$. By the Bayes formula, the posterior probabilities are:

$$p_{i,k} \propto a_k \phi(x_i \mid \mu_k, \Sigma_k), \quad \sum_{k=1}^{K} p_{i,k} = 1 \ .$$

- Then each summand in $Q(\theta'|\theta)$ is

$$\begin{aligned} & E\left[\log(a'_{y_i}) + \log\phi(x_i \mid \mu'_{y_i}, \Sigma'_{y_i}) \mid x_i, \theta\right] \\ = & \sum_{k=1}^{K} p_{i,k} \log a'_k + \sum_{k=1}^{K} p_{i,k} \log\phi(x_i \mid \mu'_k, \Sigma'_k) \ . \end{aligned}$$

- Note that we cannot see the direct effect of $\theta$ in the above equation, but $p_{i,k}$ are computed using $\theta$, i.e, the current parameters. $\theta'$ includes the updated parameters.

- We then have:

$$\begin{aligned} Q(\theta'|\theta) \ = \ & \sum_{i=1}^{n}\sum_{k=1}^{K} p_{i,k} \log a'_k + \\ & \sum_{i=1}^{n}\sum_{k=1}^{K} p_{i,k} \log\phi(x_i \mid \mu'_k, \Sigma'_k) \end{aligned}$$

- Note that the prior probabilities $a'_k$ and the parameters of the Gaussian components $\mu'_k$, $\Sigma'_k$ can be optimized separately.

- The $a'_k$'s subject to $\sum_{k=1}^{K} a'_k = 1$. Basic optimization theories show that $a'_k$ are optimized by

$$a'_k = \frac{\sum_{i=1}^{n} p_{i,k}}{n} .$$

- The optimization of $\mu_k$ and $\Sigma_k$ is simply a maximum likelihood estimation of the parameters using samples $x_i$ with weights $p_{i,k}$. Basic optimization techniques also lead to

$$\mu'_k = \frac{\sum_{i=1}^{n} p_{i,k} x_i}{\sum_{i=1}^{n} p_{i,k}}$$

$$\Sigma'_k = \frac{\sum_{i=1}^{n} p_{i,k} (x_i - \mu'_k)(x_i - \mu'_k)^t}{\sum_{i=1}^{n} p_{i,k}}$$

- After every iteration, the likelihood function $L$ is guaranteed to increase (may not strictly).

- We have derived the EM algorithm for a mixture of Gaussians.

# EM Algorithm for the Mixture of Gaussians

Parameters estimated at the $p$th iteration are marked by a superscript $(p)$.

1. Initialize parameters

2. E-step: Compute the posterior probabilities for all $i = 1, ..., n,\ k = 1, ..., K$.

$$p_{i,k} = \frac{a_k^{(p)} \phi(x_i \mid \mu_k^{(p)}, \Sigma_k^{(p)})}{\sum_{k=1}^{K} a_k^{(p)} \phi(x_i \mid \mu_k^{(p)}, \Sigma_k^{(p)})} .$$

3. M-step:

$$a_k^{(p+1)} = \frac{\sum_{i=1}^{n} p_{i,k}}{n}$$

$$\mu_k^{(p+1)} = \frac{\sum_{i=1}^{n} p_{i,k} x_i}{\sum_{i=1}^{n} p_{i,k}}$$

$$\Sigma_k^{(p+1)} = \frac{\sum_{i=1}^{n} p_{i,k} (x_i - \mu_k^{(p+1)})(x_i - \mu_k^{(p+1)})^t}{\sum_{i=1}^{n} p_{i,k}}$$

4. Repeat step 2 and 3 until converge.

Comment: for mixtures of other distributions, the EM algorithm is very similar. The E-step involves computing the posterior probabilities. Only the particular distribution $\phi$ needs to be changed. The M-step always involves parameter optimization. Formulas differ according to distributions.

# Computation Issues

- If a different $\Sigma_k$ is allowed for each component, the likelihood function is not bounded. Global optimum is meaningless. (Don't overdo it!)
- How to initialize? Example:
  - Apply k-means first.
  - Initialize $\mu_k$ and $\Sigma_k$ using all the samples classified to cluster $k$.
  - Initialize $a_k$ by the proportion of data assigned to cluster $k$ by k-means.
- In practice, we may want to reduce model complexity by putting constraints on the parameters. For instance, assume equal priors, identical covariance matrices for all the components.