ECEN 689 Materials Informatics

Unsupervised Learning

Ulisses Braga-Neto

ECE Department

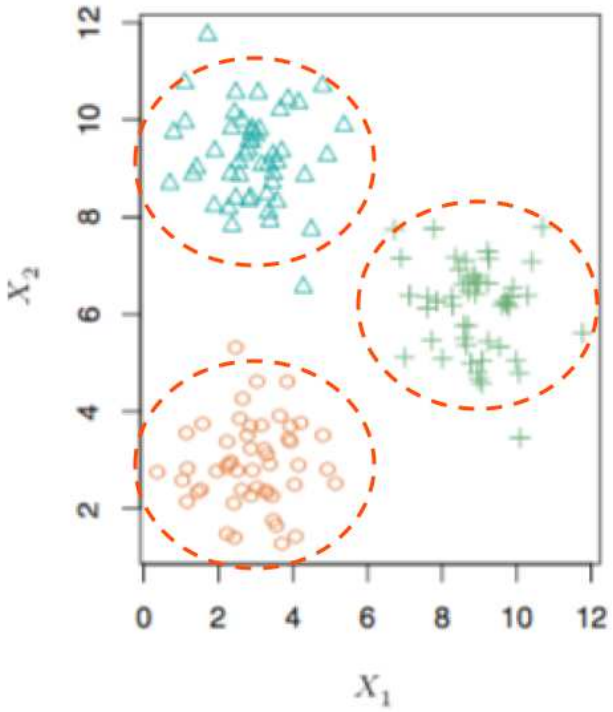
Texas A&M University

Unsupervised Learning

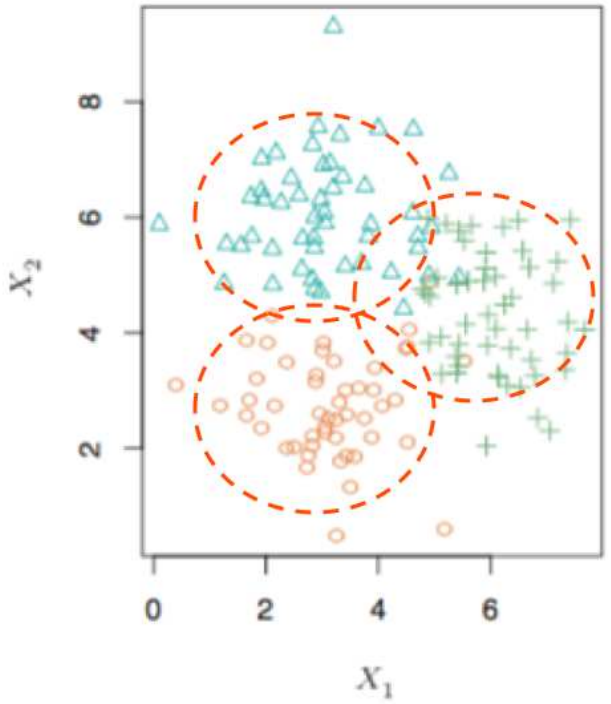
- In some situations, training data are available without labeling (or it is only partially labeled, a case we will not consider here).
- This could be because of the expense of labeling the data, or the unavailability of reliable labels, or because the data are perceived to come from a single group.
- The data could still be valuable for identifying the structure of the underlying data distribution. Namely, we are interested in:
  - Finding subgroups (“clusters”)
  - Building a hierarchical data representation.
  - Discovering new classes.
  - Visualization in low-dimensional spaces.

Example: Finding Groups

- Unspupervised learning is used to organize the data into groups.



Easy

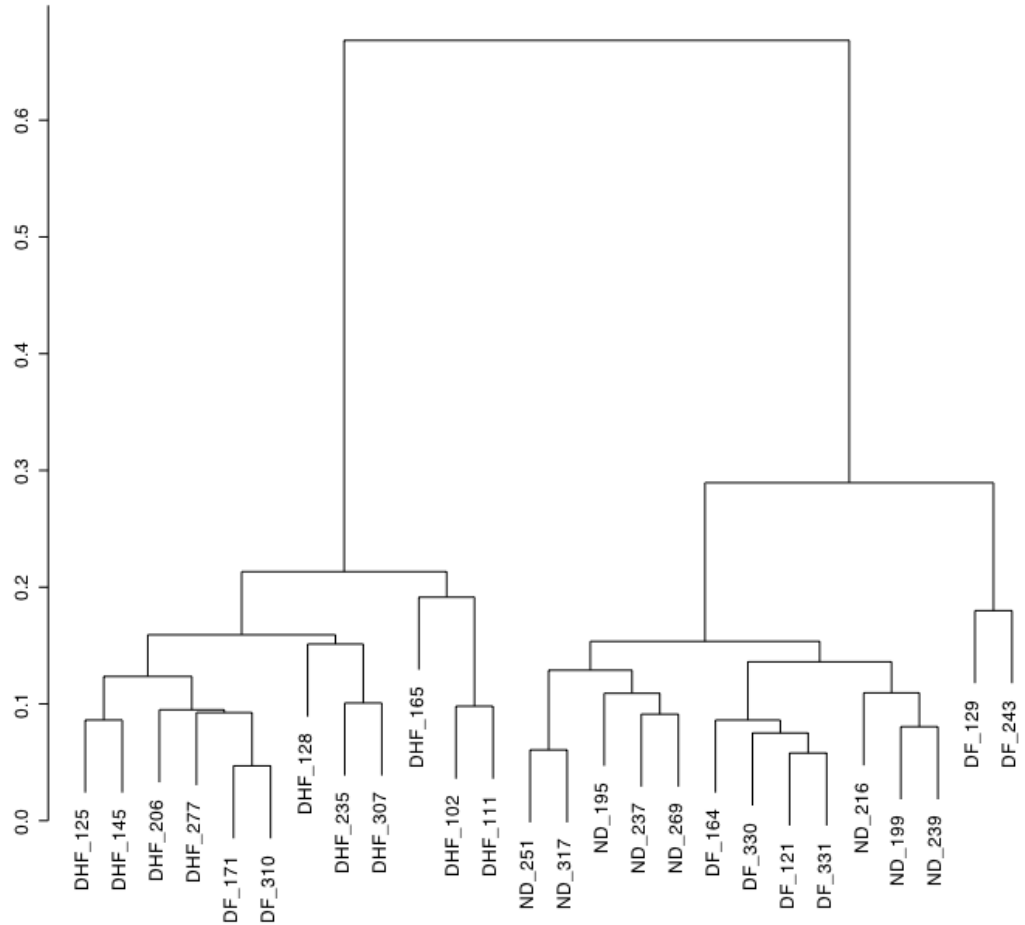


Hard

(Adapted from James et al. "Introduction to Statistical Learning")

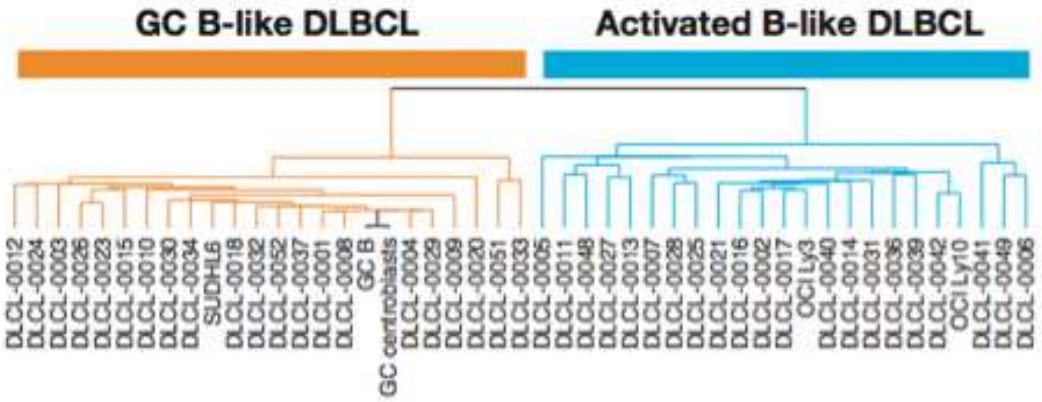
Example: Hierarchical Representation

- Unspupervised learning is used to build a hierarchical representation of the information.



Example: Class Discovery

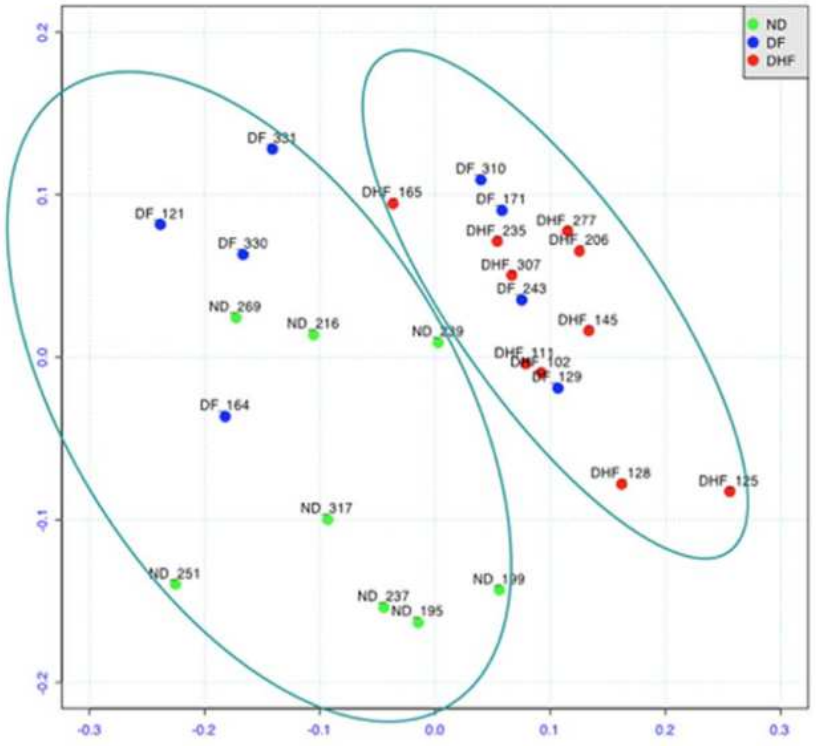
- Unspupervised learning is used to discover previously unknown classes.

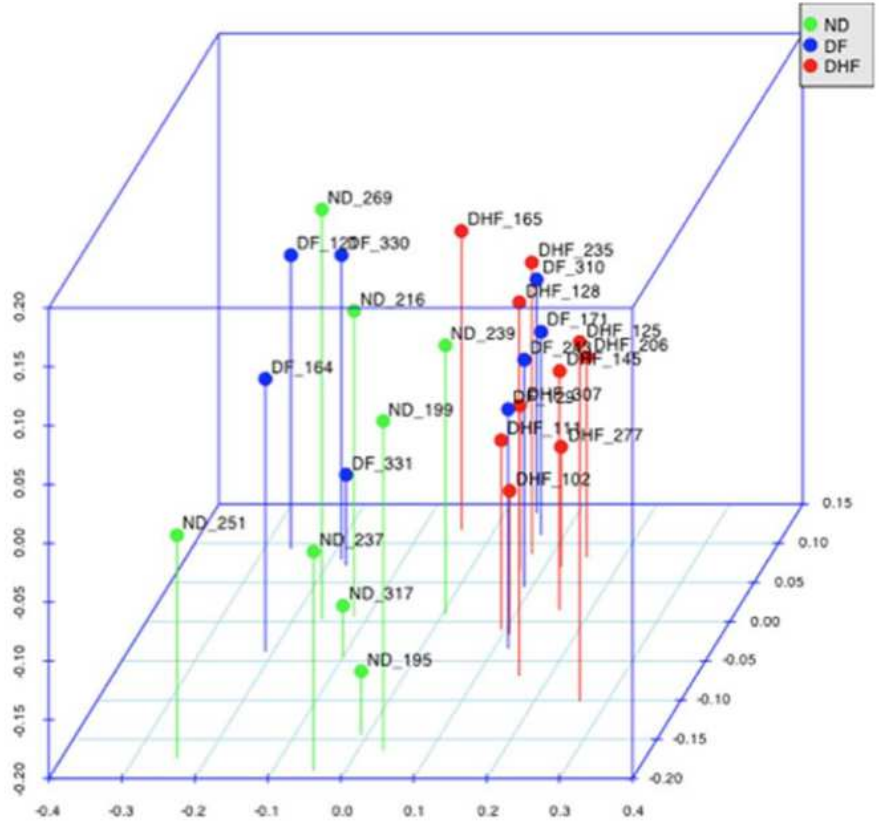


Two new cancer subtypes discovered by clustering

Example: Vizualization

- Unspupervised learning is used to project high-dimensional data into low-dimensional spaces.





(Adapted from Nascimento et al. "Gene Expression Profiling during Early Acute Febrile Stage of Dengue Infection Can Predict the Disease Outcome," PLoS ONE, 4(11), 2009)

K-Means Clustering

- Given data $S_n = \{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$, the objective is
  - Finding K cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ (K is given).
  - For each point $\mathbf{X}_i$, finding an assignment to one of the K clusters.
- Cluster assignment is made by vectors $\mathbf{r}_1, \ldots, \mathbf{r}_n$ where each $\mathbf{r}_i$ is a vector of size K with

$$\mathbf{r}_i = (0,\ 0,\ \cdots, 1, \cdots,\ 0,\ 0), \quad \text{for } i = 1, \ldots, n$$

  such that $\mathbf{r}_i(k) = 1$ if $\mathbf{X}_i$ belongs to cluster k, for $k = 1, \ldots, K$ (each point can belong to only one cluster).
- For example, with $K = 3$ and $n = 4$, we might have $\mathbf{r}_1 = (1, 0, 0)$, $\mathbf{r}_2 = (0, 0, 1)$, $\mathbf{r}_3 = (1, 0, 0)$, $\mathbf{r}_4 = (0, 1, 0)$, in which case $\mathbf{X}_1, \mathbf{X}_3$ are assigned to cluster 1, $\mathbf{X}_2$ is assigned to cluster 3, while $\mathbf{X}_4$ is assigned to cluster 2.

K-Means Problem Formulation

- The K-means algorithm seeks for the vectors $\{\boldsymbol{\mu}_i\}_{i=1}^{n}$ and $\{\mathbf{r}_i\}_{k=1}^{K}$ that minimize the distance criterion

$$J = \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbf{r}_i(k)||\mathbf{X}_i - \boldsymbol{\mu}_k||^2$$

- The solution can be obtained iteratively with two optimizations at each step:
  - Hold current values $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$ fixed, find $\{\mathbf{r}_i\}_{i=1}^{n}$ that minimizes J ("E-Step").
  - Hold current values $\{\mathbf{r}_i\}_{i=1}^{n}$ fixed, find $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$ that minimizes J ("M-Step").
- The nomenclature "E-step" and "M-step" is due to an analogy with the EM ("Expectation-Maximization") algorithm for Gaussian mixtures (more on this later).

K-Means "E-Step"

- With the current values $\{\boldsymbol{\mu}_k\}_{k=1}^K$ fixed, the values $\{\mathbf{r}_i\}_{i=1}^n$ that minimize J can be found by inspection.

$$\mathbf{r}_i(k) = \begin{cases} 1, & \text{if } k = \arg\min_{j=1,\ldots,K} ||\mathbf{X}_i - \boldsymbol{\mu}_j||^2, \\ 0 & \text{otherwise.} \end{cases}$$

  for $i = 1, \ldots, n$.

- In other words, we simply assign each point to the closest cluster mean.

K-Means "M-Step"

- With the current values $\{\mathbf{r}_i\}_{i=1}^n$ fixed, the values $\{\boldsymbol{\mu}_k\}_{k=1}^K$ that minimize J can be found by simple differentiation:

$$\frac{\partial J}{\partial \boldsymbol{\mu}_k} = 2\sum_{i=1}^{n} \mathbf{r}_i(k)||\mathbf{X}_i - \boldsymbol{\mu}_k|| = 0\,,$$

  which gives

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^n \mathbf{r}_i(k)\mathbf{X}_i}{\sum_{i=1}^n \mathbf{r}_i(k)}\,,$$

  for $k = 1, \ldots, K$.

- In other words, we simply assign to $\boldsymbol{\mu}_k$ the mean value of all training points assigned to cluster k in the previous "E-Step".

K-Means Algorithm

K-Means Clustering Algorithm

1: Initialize K and $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$.

2: **repeat**

3: E-Step: Assign each point to closest cluster mean.

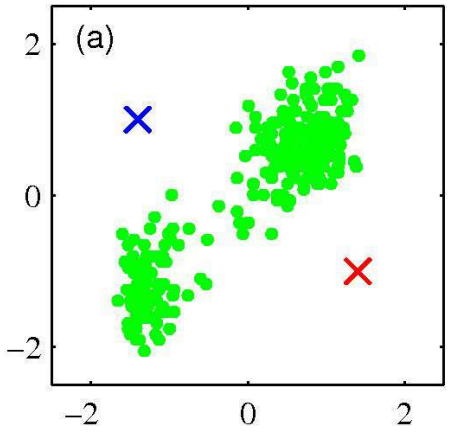
4: M-Step: Update cluster means.

5: **until** there is no signficant change to the criterion J.

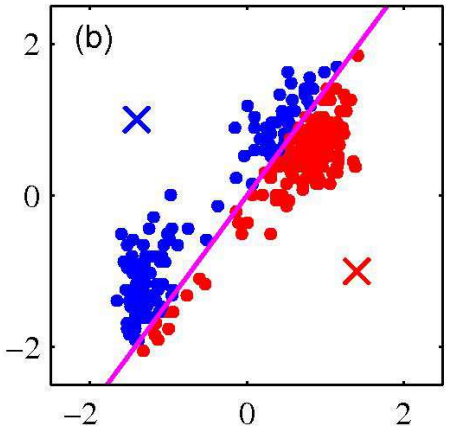
K-Means Algorithm Example

- "Old Faithful" data set (C. Bishop).

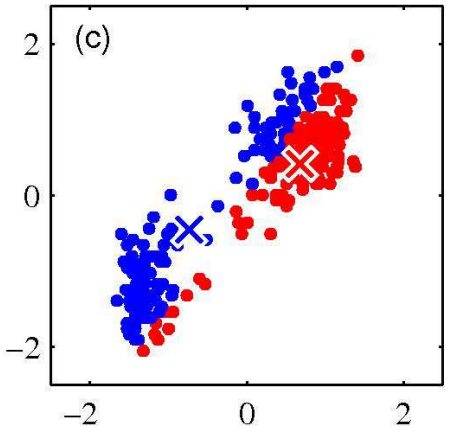
Original data



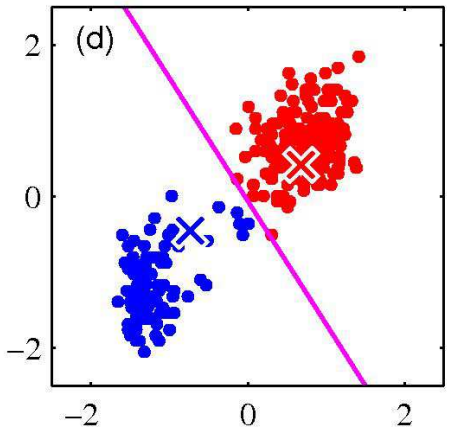
E-Step



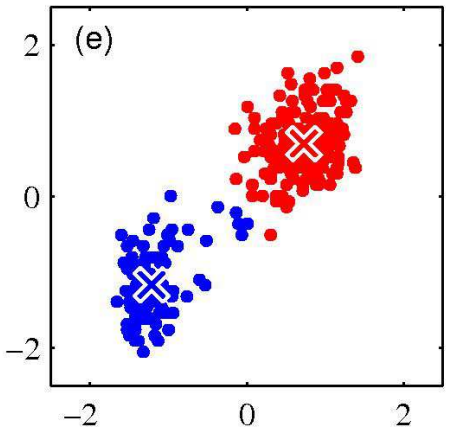
M-Step



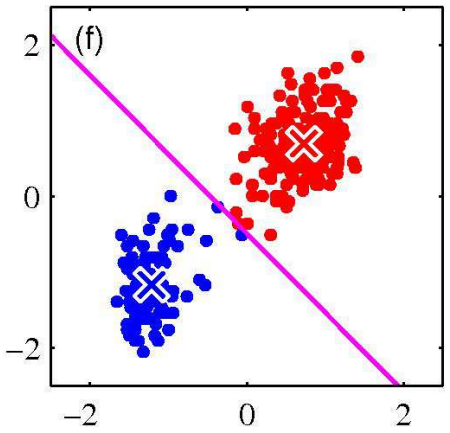
E-Step



M-Step

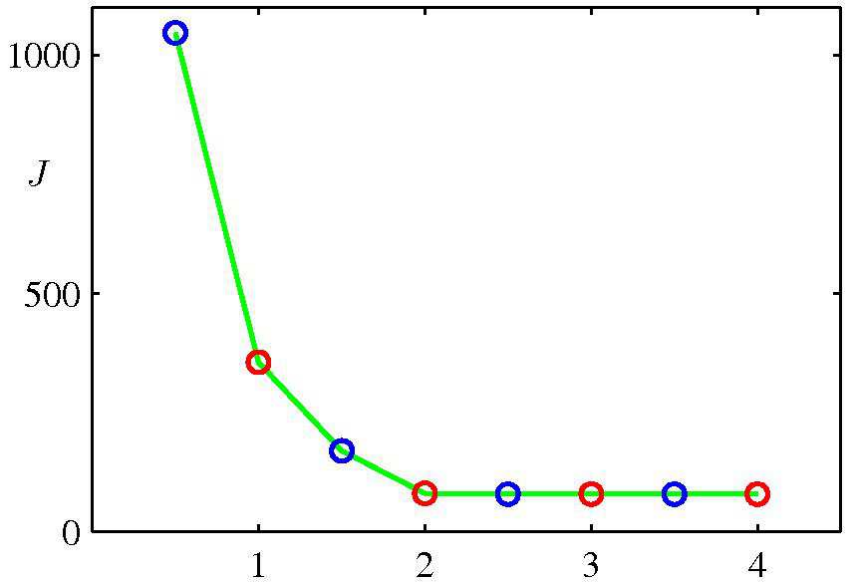


E-Step ...



K-Means Algorithm Example

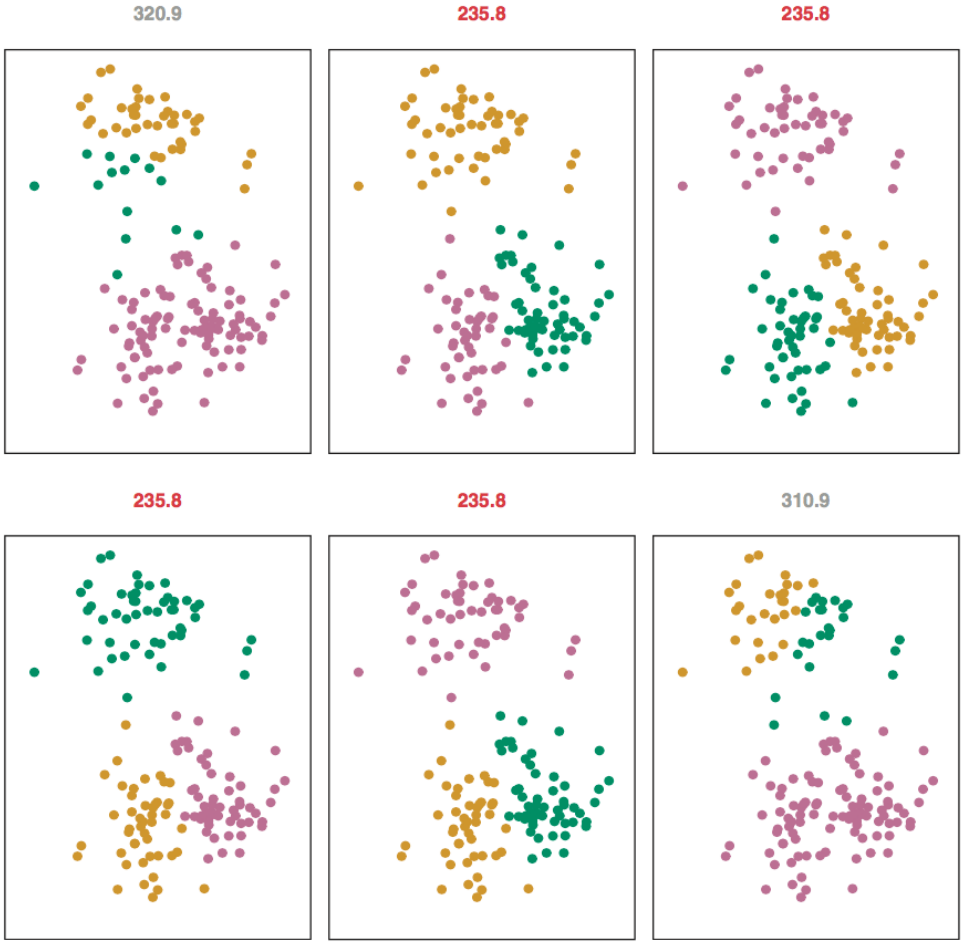
- Plot of cost function J as function of iteration number for previous example.
  E-step: red circle; M-step: blue circle.



- It is clear that the algorithm has converged after only two iterations.

Random Restarts

- By restarting the algorithm with multiple random initial values, local minima may be avoided.



(Source: James, Witten, Hastie, Tibshirani, 2009.

Fuzzy K-Means Clustering

- Also known as the "fuzzy c-means algorithm."
- Here, each vector $\mathbf{r}_i$ can assume nonnegative values such that $\sum_{k=1}^{K} \mathbf{r}_i(k) = 1$. The value $0 \leq \mathbf{r}_i(k) \leq 1$ gives the *degree of membership* of point $\mathbf{X}_i$ to cluster k.
- The algorithm seeks vectors $\{\boldsymbol{\mu}_i\}_{i=1}^{n}$ and $\{\mathbf{r}_i\}_{k=1}^{K}$ that minimize the distance criterion

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbf{r}_i(k)^s ||\mathbf{X}_i - \boldsymbol{\mu}_k||^2$$

  where $s \geq 1$ is a parameter that controls the "fuzziness" of the resulting clusters.
- This can be solved by a similar process to the usual K-means algorithm, with E and M steps.

Model-Based Clustering

- The K-means algorithm makes no assumption about the distribution of the data.
- A different approach is to assume a particular parametric shape for the distribution and estimate the parameters from the data (this is similar to parametric classification rules).
- The appropriate assumption for clustering is a *mixture* of probability densities.
- We will consider here the Gaussian Mixture Model (GMM) and maximum-likelihood parameter estimation.

Gaussian Mixture Model

- The GMM for the overall data distribution is:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)$$

where K is the desired number of clusters and $\pi_1, \ldots, \pi_K$ are nonnegative numbers, with $\sum_{i=1}^{K} \pi_k = 1$, called the *mixing parameters*.

- The mixing parameter π_k is simply the *a-priori* probability that a given random point $\mathbf{X}$ belongs to cluster C_k:

$$\pi_k = P(\mathbf{X} \in C_k)\,,$$

for $k = 1, \ldots, K$.

Cluster "Responsibilities"

- Bayes' theorem allows one to write:

$$\begin{aligned}\gamma_k(\mathbf{x}) &= P(\mathbf{X} \in C_k \mid \mathbf{X} = \mathbf{x}) \\ &= \frac{p(\mathbf{X} = \mathbf{x} \mid \mathbf{X} \in C_k) P(\mathbf{X} \in C_k)}{\sum_{k=1}^{K} p(\mathbf{X} = \mathbf{x} \mid \mathbf{X} \in C_k) P(\mathbf{X} \in C_k)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)},\end{aligned}$$

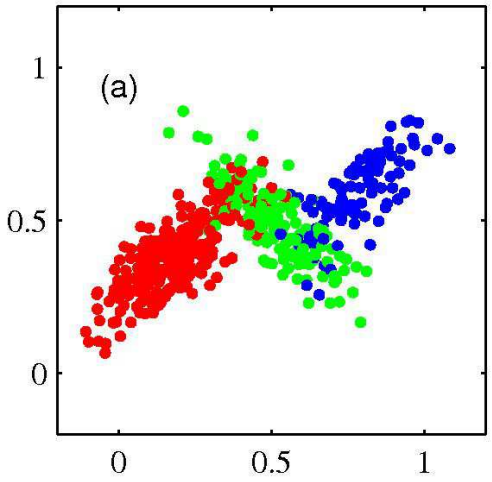
  for $k = 1, \ldots, K$.

- The number $\gamma_k(\mathbf{x})$ is the "responsibility" that cluster k takes for explaining the given point $\mathbf{x}$. Note the similarity with the posterior probability $\eta_k(\mathbf{x}) = P(Y = k \mid \mathbf{X} = \mathbf{x})$ in classification.

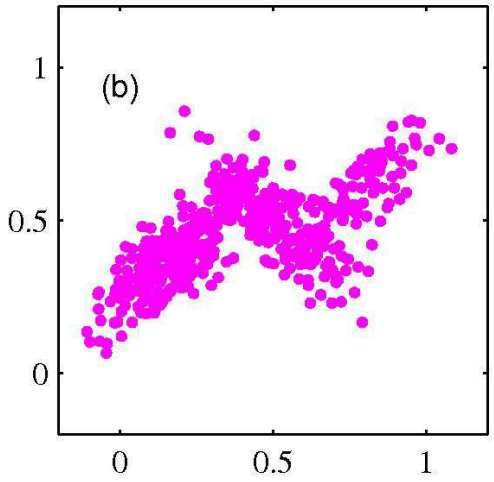
Gaussian Mixture Example

- Synthetic data, K = 3 (C. Bishop).

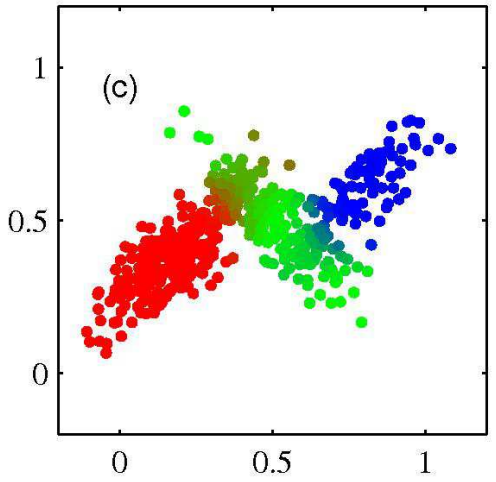
Data with cluster membership



Available data



Data with cluster responsibilities



- Just like in classification, “hard” cluster membership can be obtained by assigning to a point the cluster with the largest responsibility. Thus, clustering can be performed by estimating the cluster responsibilities.

Maximum-Likelihood Estimation

- Since $\{\gamma_k(\mathbf{X}_n)\}_{i=1}^n$ is a function of $\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{i=1}^K$, we need to find estimates of these parameters.
- Given independence, the likelihood function is

$$p(S_n \mid \{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{i=1}^K) = \Pi_{i=1}^n \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{X}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \right).$$

- Therefore, the log-likelihood function can be written as:

$$L = \ln p(S_n \,|\, \{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{i=1}^K) = \sum_{i=1}^n \ln \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{X}_i \,|\, \boldsymbol{\mu}_k, \Sigma_k) \right).$$

- The maximum-likelihood parameter estimates are:

$$\{\hat{\pi}_k, \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k\}_{i=1}^K = \arg\max \ \ln p(S_n \mid \{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{i=1}^K).$$

M-step for Gausian Means

- The values $\{\boldsymbol{\mu}_k\}_{k=1}^K$ that maximize the log-likelihood L can be found by simple differentiation:

$$\frac{\partial L}{\partial \boldsymbol{\mu}_k} = \sum_{i=1}^{n} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)}}_{\gamma_k(\mathbf{X}_i)!} \Sigma_k^{-1}(\mathbf{X}_i - \boldsymbol{\mu}_k) = 0$$

which gives (by plugging in current estimates):

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^{n} \hat{\gamma}_k(\mathbf{X}_i)\mathbf{X}_i}{\sum_{i=1}^{n} \hat{\gamma}_k(\mathbf{X}_i)} .$$

for $k = 1, \ldots, K$.

M-step for Gausian Covariance

- The values $\{\Sigma_k\}_{k=1}^K$ that maximize the log-likelihood L can be found as in the previous slide, by setting $\frac{\partial L}{\partial \Sigma_k} = 0$, which gives (by plugging in current estimates):

$$\hat{\Sigma}_k = \frac{\sum_{i=1}^n \hat{\gamma}_k(\mathbf{X}_i)(\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)^T}{\sum_{i=1}^n \hat{\gamma}_k(\mathbf{X}_i)},$$

for $k = 1, \ldots, K$.

M-step for Mixing Parameters

- The values $\{\pi_k\}_{k=1}^K$ that maximize the log-likelihood L can be found by a slightly more complex optimization process, using Lagrance multipliers (due to the constraints $\pi_k \geq 0$ and $\sum_k \pi_k = 1$. The solution turns out to be simply (by plugging in current estimates):

$$\hat{\pi}_k = \frac{1}{n} \sum_{i=1}^{n} \hat{\gamma}_k(\mathbf{X}_i)$$

for $k = 1, \ldots, K$.

E-step

- The E-step corresponds simply to updating the responsibility estimates given the estimates $\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{i=1}^K$ obtained in the M-step:

$$\hat{\gamma}_k(\mathbf{X}_i) \;=\; \frac{\hat{\pi}_k \mathcal{N}(\mathbf{X}_i \mid \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k)}{\sum_{k=1}^K \hat{\pi}_k \mathcal{N}(\mathbf{X}_i \mid \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k)} \,,$$

for $i = 1, \ldots, n$ and $k = 1, \ldots, K$.

GMM Clustering Algorithm

GMM Clustering Algorithm

1: Initialize K and $\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{i=1}^{K}$.

2: **repeat**

3: E-Step: Update $\hat{\gamma}_k(\mathbf{X}_i)$, for $i = 1, \ldots, n$.

4: M-Step: Update $\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{i=1}^{K}$.

5: **until**
there is no significant change in the log-likelihood L.

6: Assign $\mathbf{X}_i$ to the cluster with the largest responsibility $\hat{\gamma}_k(\mathbf{X}_i)$, for $i = 1, \ldots, n$.

Estimation Singularities

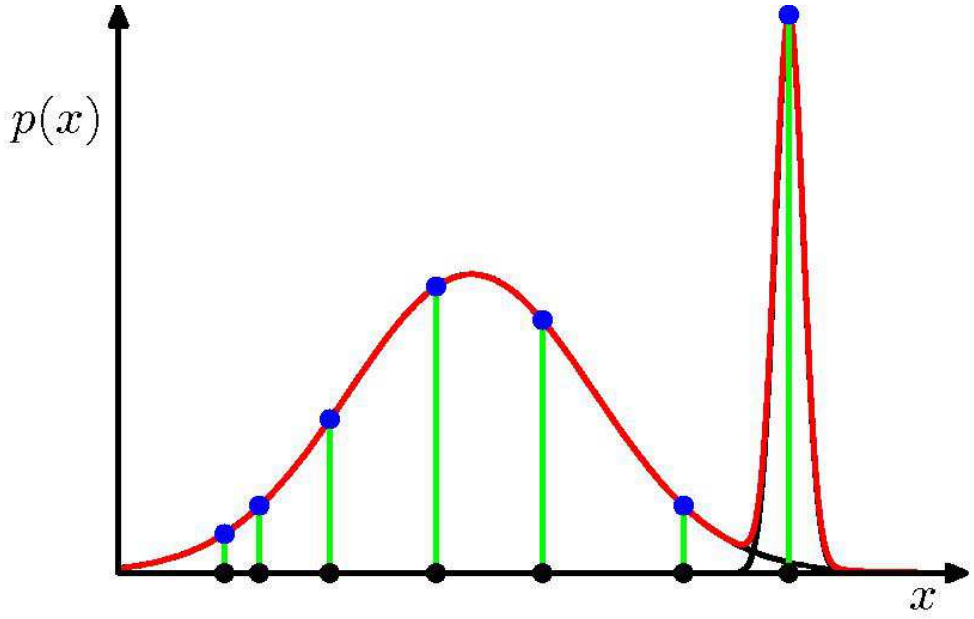
- If covariance matrices need to be estimated, then singularities may occur.
- To see this, assume that one of the mean estimates coincides with training point $\mathbf{X}_1$. The the log-likelihood becomes

$$L = \frac{1}{(2\pi)^{d/2}|\hat{\Sigma}_1|^{1/2}} + \sum_{i=2}^{n} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{X}_i \,|\, \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k)\right).$$

By letting $|\hat{\Sigma}_1| \rightarrow \infty$, one can increase L without bound, so that no meaningful estimate of Σ_1 results.

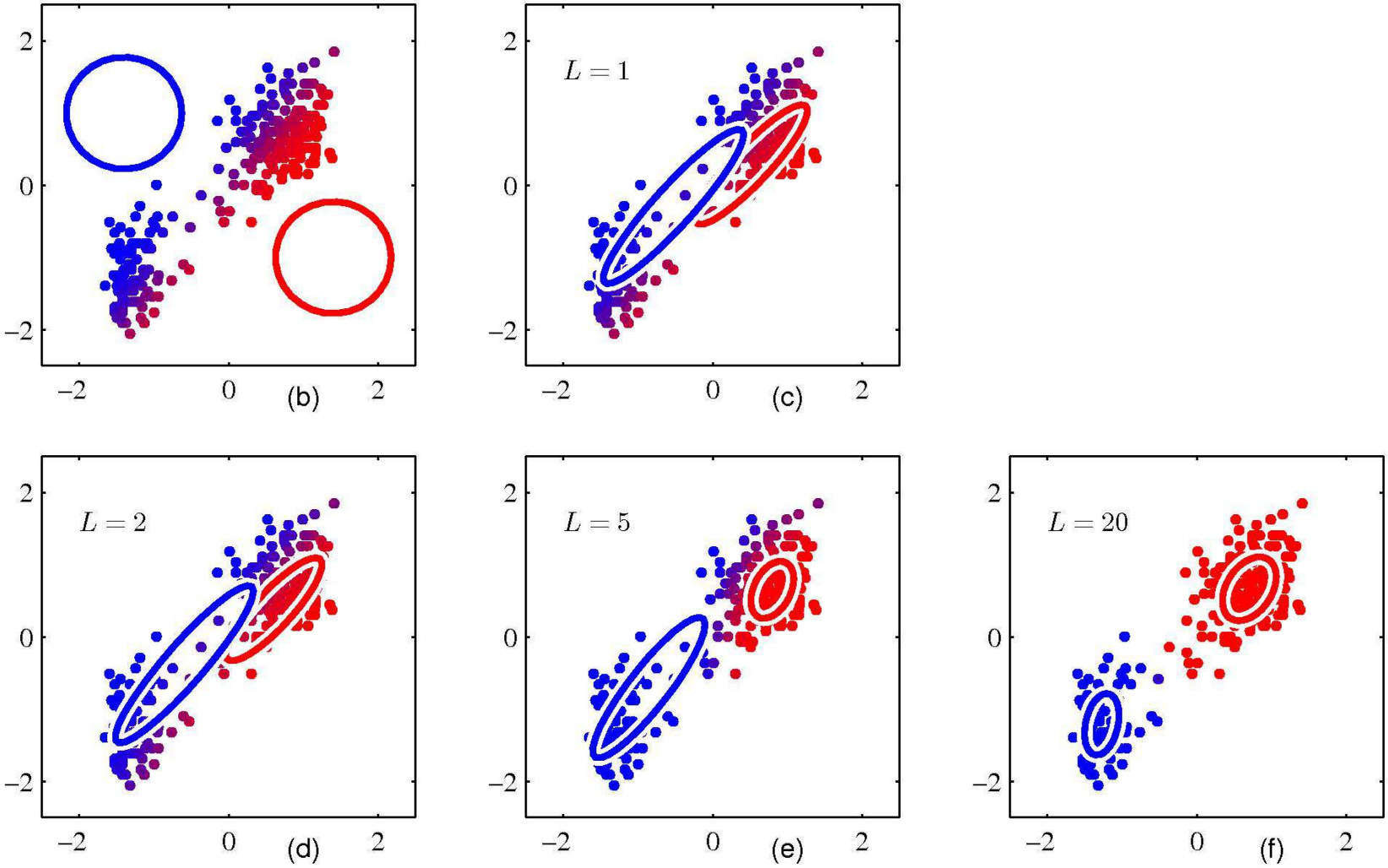
Estimation Singularities - II

- In order to avoid that, the GMM algorithm needs to check whether any of the cluster covariances is "collapsing" and if so, reinitialize the mean and covariance means of that cluster.
- Graphical Example (C. Bishop).



GMM Fitting Example

- "Old Faithful" data set (C. Bishop).



Relationship to K-Means Algorithm

- The relationship can be revealed by considering the case where $\Sigma_k = \sigma^2 I$ (spherical covariance matrices).
- In this case, the cluster reponsibilities for $\mathbf{X}_i$ become:

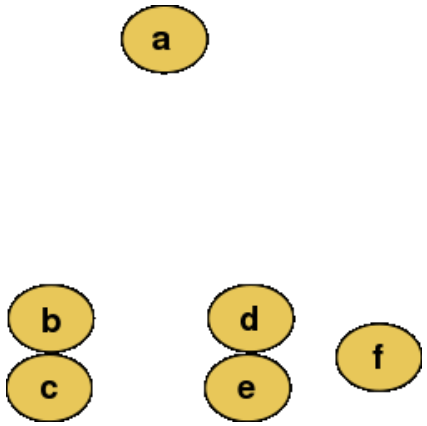
$$\gamma_k(\mathbf{X}_i) = \frac{\pi_k \exp(-||\mathbf{x} - \boldsymbol{\mu}_k||^2/2\sigma^2)}{\sum_{k=1}^{K} \pi_k \exp(-||\mathbf{x} - \boldsymbol{\mu}_k||^2/2\sigma^2)},$$

  for $k = 1, \ldots, K$.
- Let $\boldsymbol{\mu}_j$ be the mean vector closest to $\mathbf{X}_i$. Then it is easy to see that, if one lets $\sigma \to 0$, then $\gamma_j(\mathbf{X}_i) \to 1$, while all other responsibilities go to zero.
- In other words, $\gamma_k(\mathbf{X}_i) \to \mathbf{r}_i(k)$, and cluster assignment is done as in the K-means algorithm.

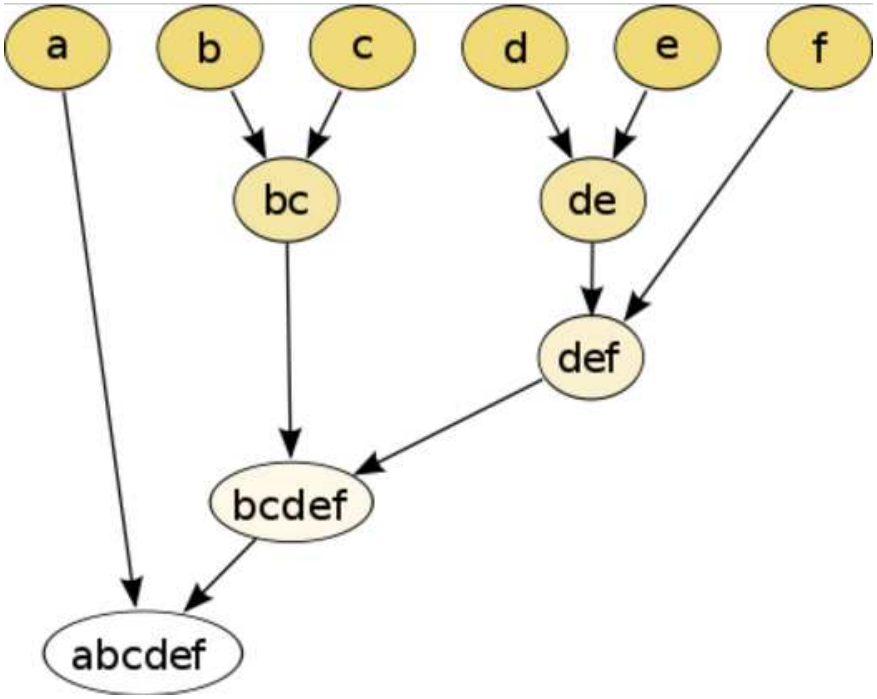
Hierachical Clustering

- Here different levels of clustering are obtained by adopting an iterative process of cluster creation.
- The process could be
  - **Agglomerative** (Bottom-up): start with each point in a separate cluster and iteratively merge clusters based on a *pairwise similarity metric*.
  - **Divisive** (Top-down): start with all the data in a single cluster and iteratively split clusters.
- Agglomerative clustering is far more common, and so we will concentrate on it.

Hierachical Clustering Example



Original data

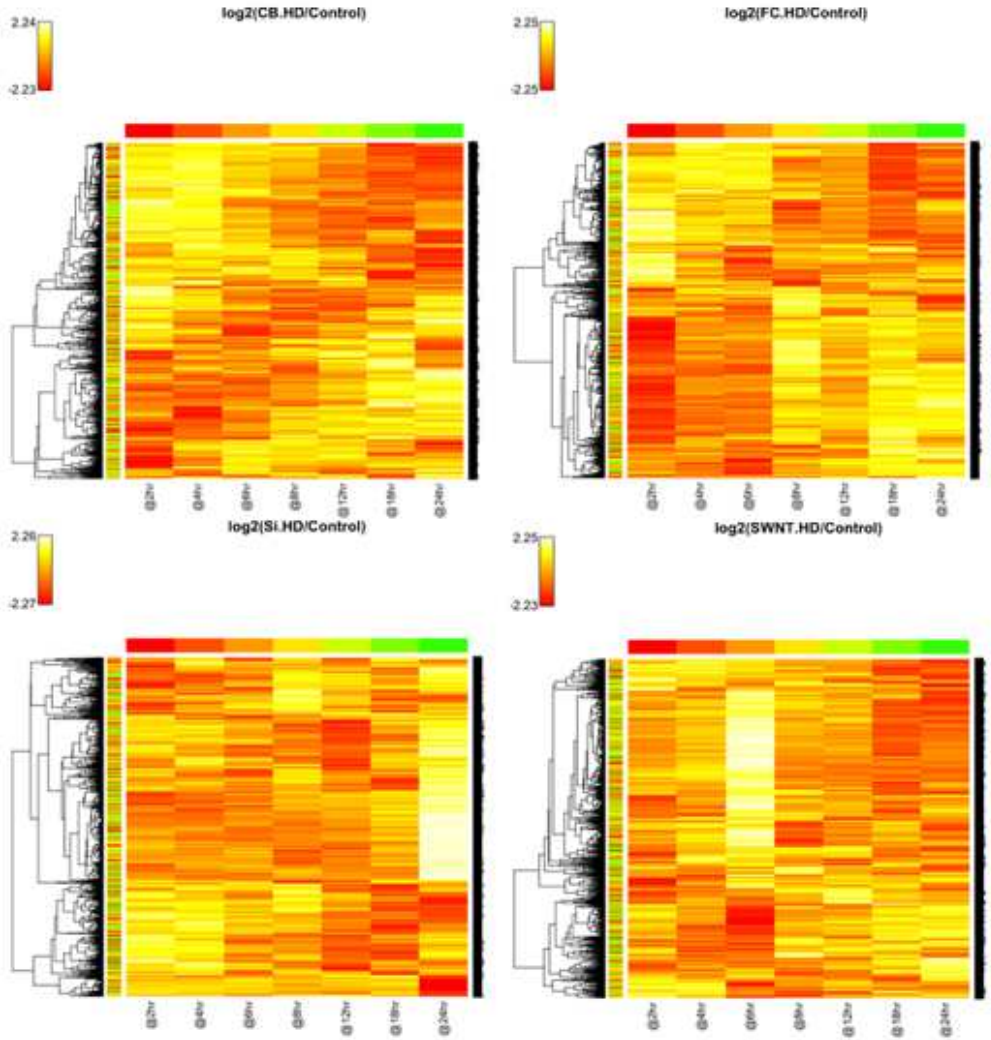


Dendrogram

(Source: wikipedia).

Hierarchical Clustering and "Heatmaps"

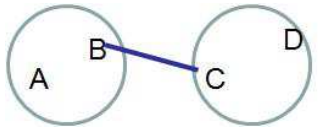
- Toxicology gene expression (Zollanvari et al. 2009).



Linkage Types

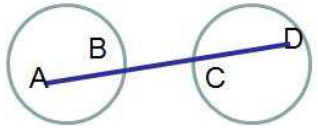
- According to the type of similarity metric, one may have:
  - Single-Linkage Hierarchical Clustering:

$$d_s(C_i, C_j) = \min\{d(x, y) \mid x \in C_i, y \in C_j\}$$



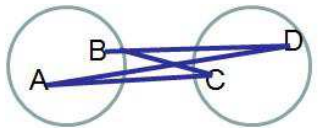
  - Complete-Linkage Hierarchical Clustering:

$$d_c(C_i, C_j) = \max\{d(x, y) \mid x \in C_i, y \in C_j\}$$



  - Average-Linkage Hierarchical Clustering:

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



(Graphics from Murphy, 2012).

Hierarchical Clustering Computation

- Example: Single-linkage with six points (Webb).
- Dissimilarity Matrix:

	1	2	3	4	5	6
1	0	4	13	24	12	8
2		0	10	22	11	10
3			0	7	3	9
4				0	6	18
5					0	8.5
6						0

Hierarchical Clustering Computation

- First iteration.

	1	2	(3,5)	4	6
1	0	4	12	24	8
2		0	10	22	10
(3,5)			0	6	8.5
4				0	18
6					0

- Second iteration.

	(1,2)	(3,5)	4	6
(1,2)	0	10	22	8
(3,5)		0	6	8.5
4			0	18
6				0

Hierarchical Clustering Computation

- Third iteration.

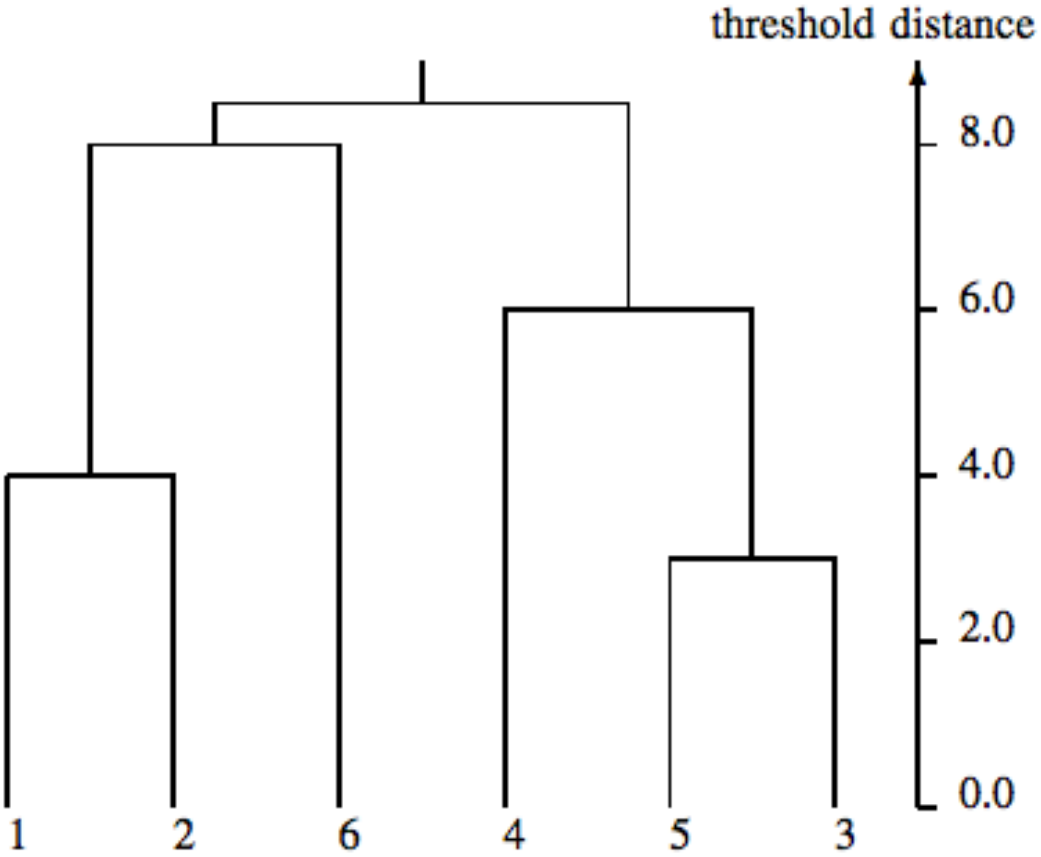
	(1,2)	(3,4,5)	6
(1,2)	0	10	8
(3,4,5)		0	8.5
6			0

- Fourth (last) iteration.

	(1,2,6)	(3,4,5)
(1,2,6)	0	8.5
(3,4,5)		0

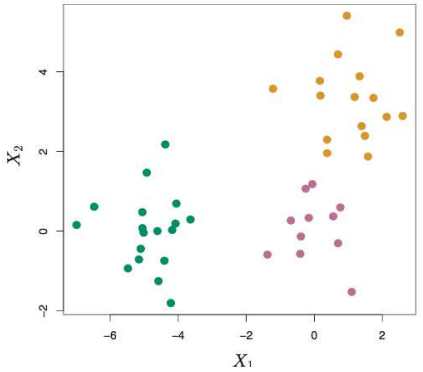
Hierachical Clustering Computation

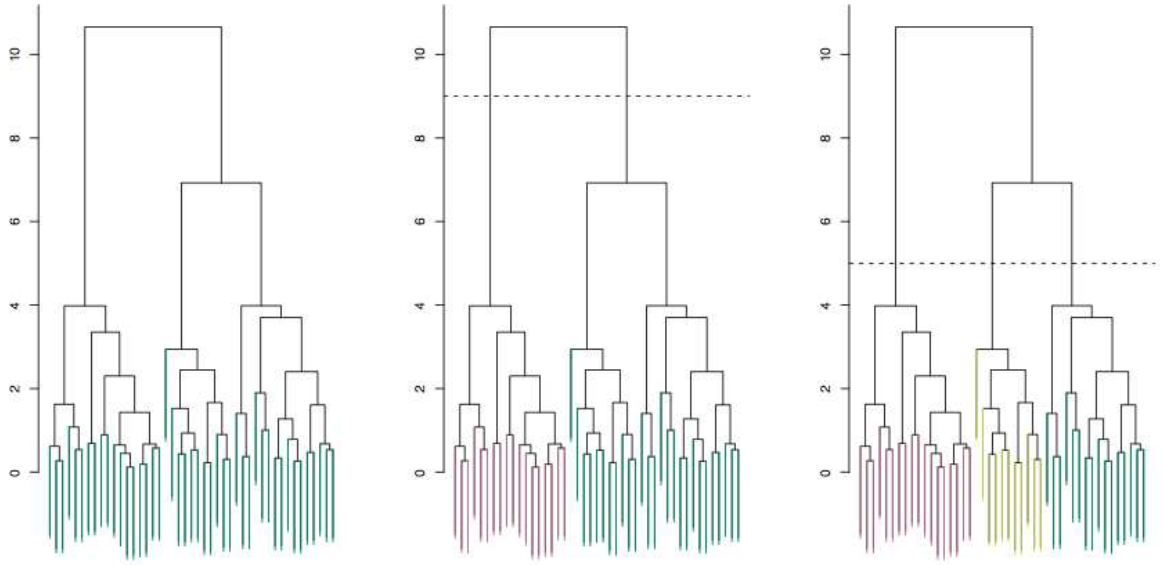
- Resulting Dendrogram.



Dendrogram Cutting

- Cutting the dendrogram produces multiple clusterings. Example with complete linkage and Euclidean distance (James at al.).

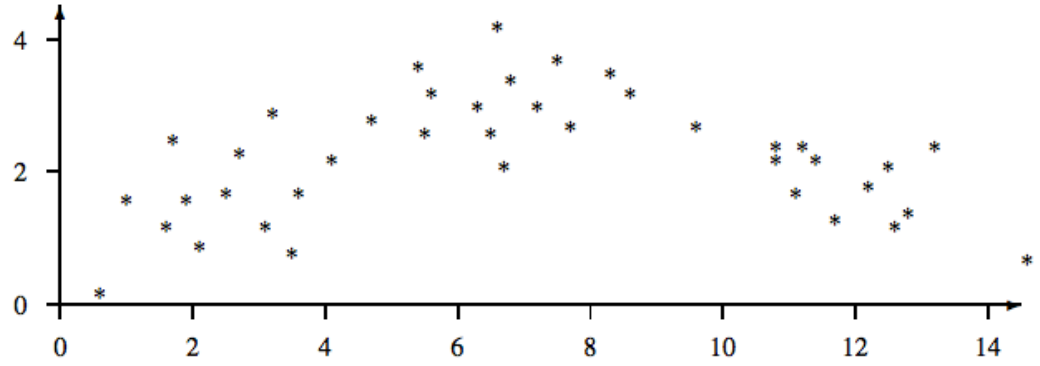




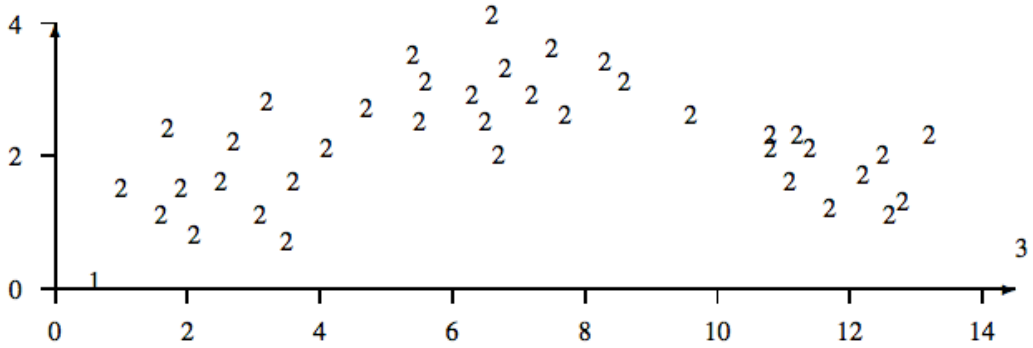
Issue with Single Linkage

- Single linkage produces “chaining,” resulting in elongated clusters. Example (Webb):

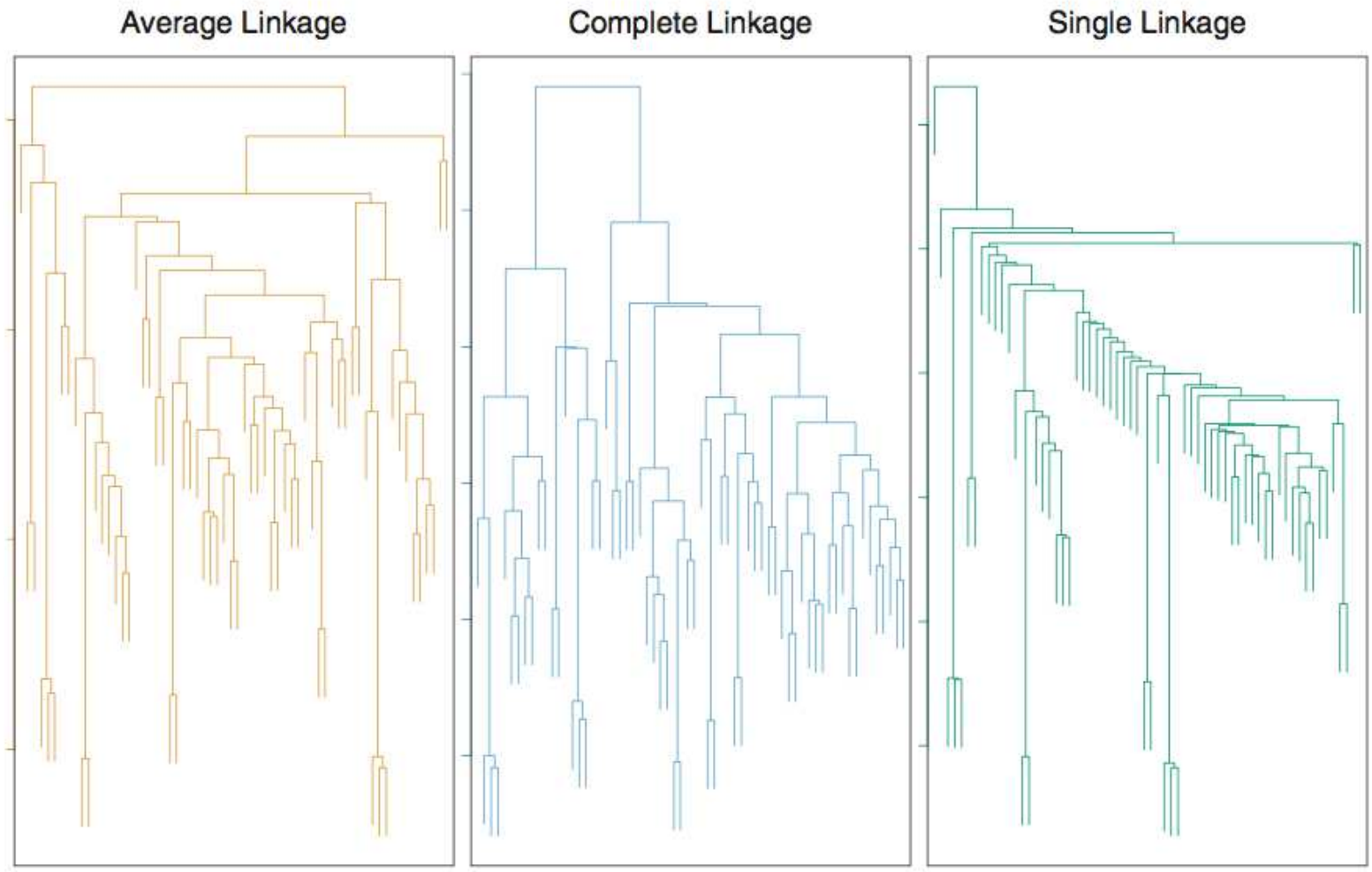
Original Data



Clusters



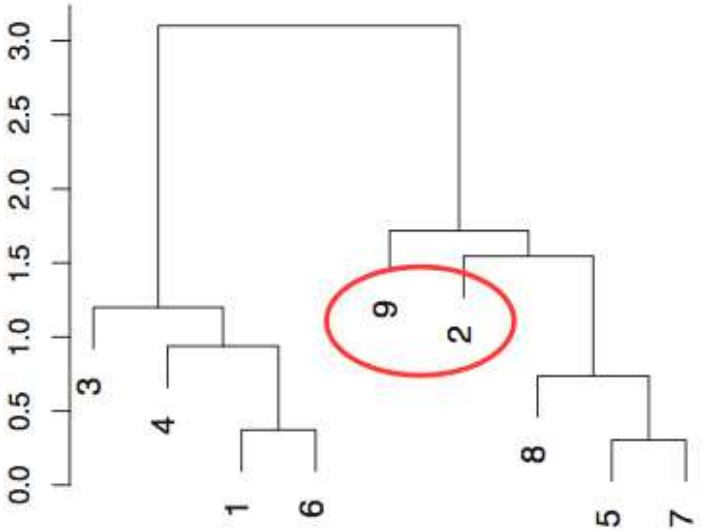
Dendrogram Comparison

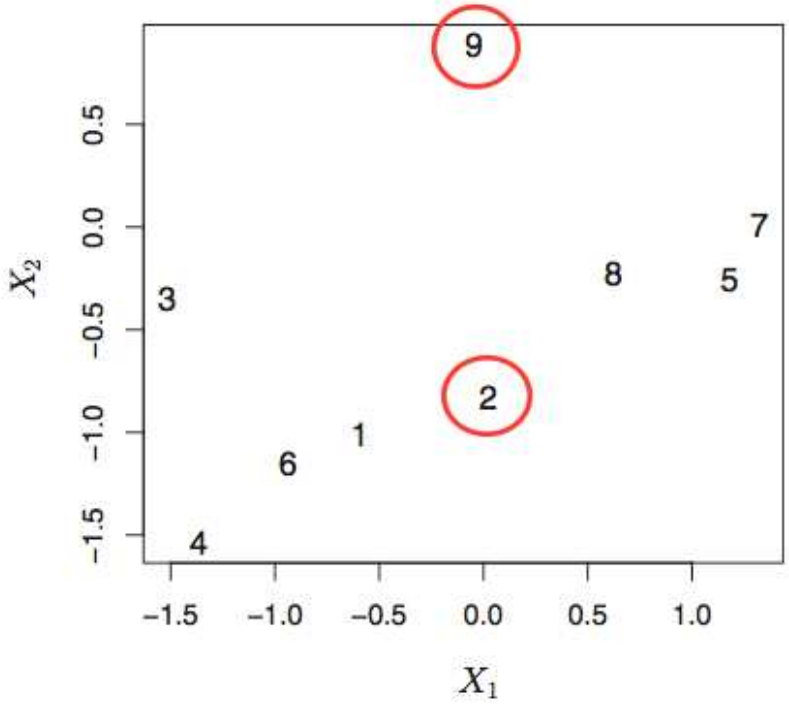


(Figure from James et al., 2009).

Caveat on Reading Dendrogram

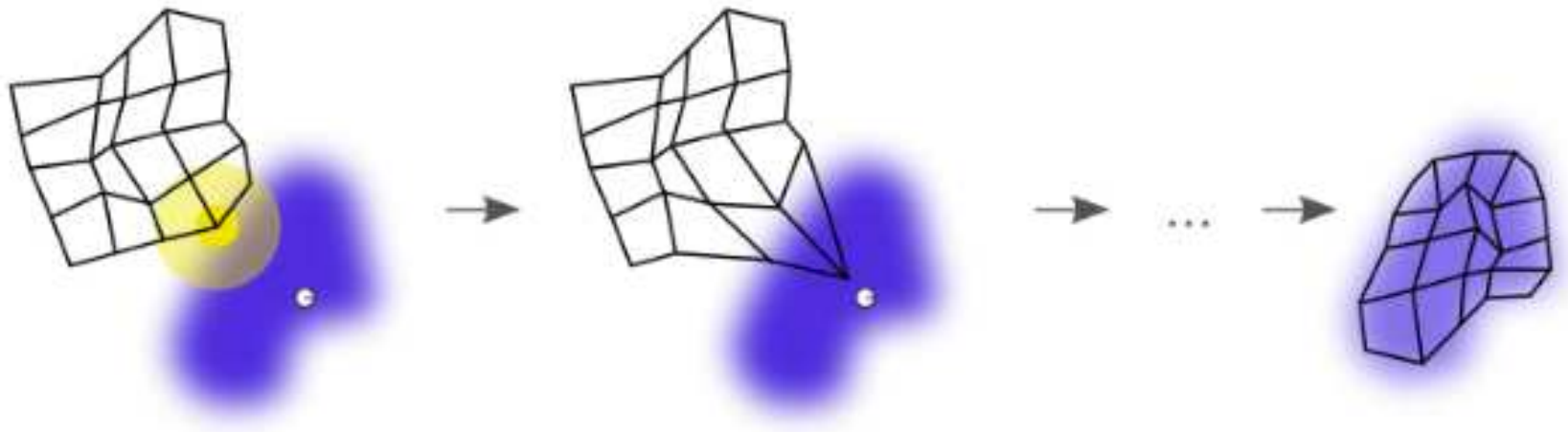
- A common mistake is to think two points are similar because they are near on the dendrogram. Example (James at al.).





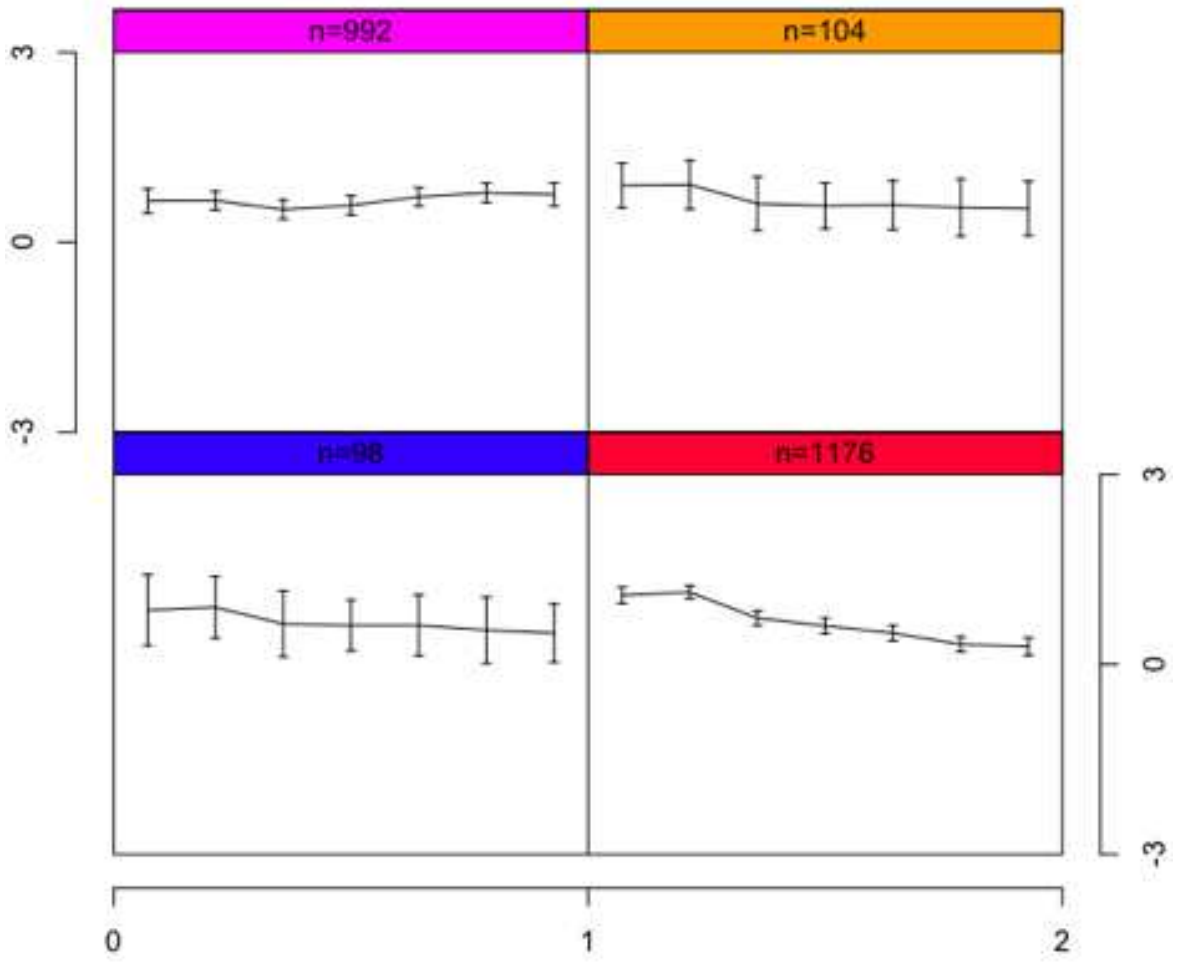
Self-Organizing Maps (SOM)

- SOM is a popular algorithm, which itereatively adapts a grid of nodes to the data.
- It can be seen as a neural network where each node of the grid is a neuron. Each node is supposed to “respond" similarly to its neighbors.
- Graphical representation (wikipedia):



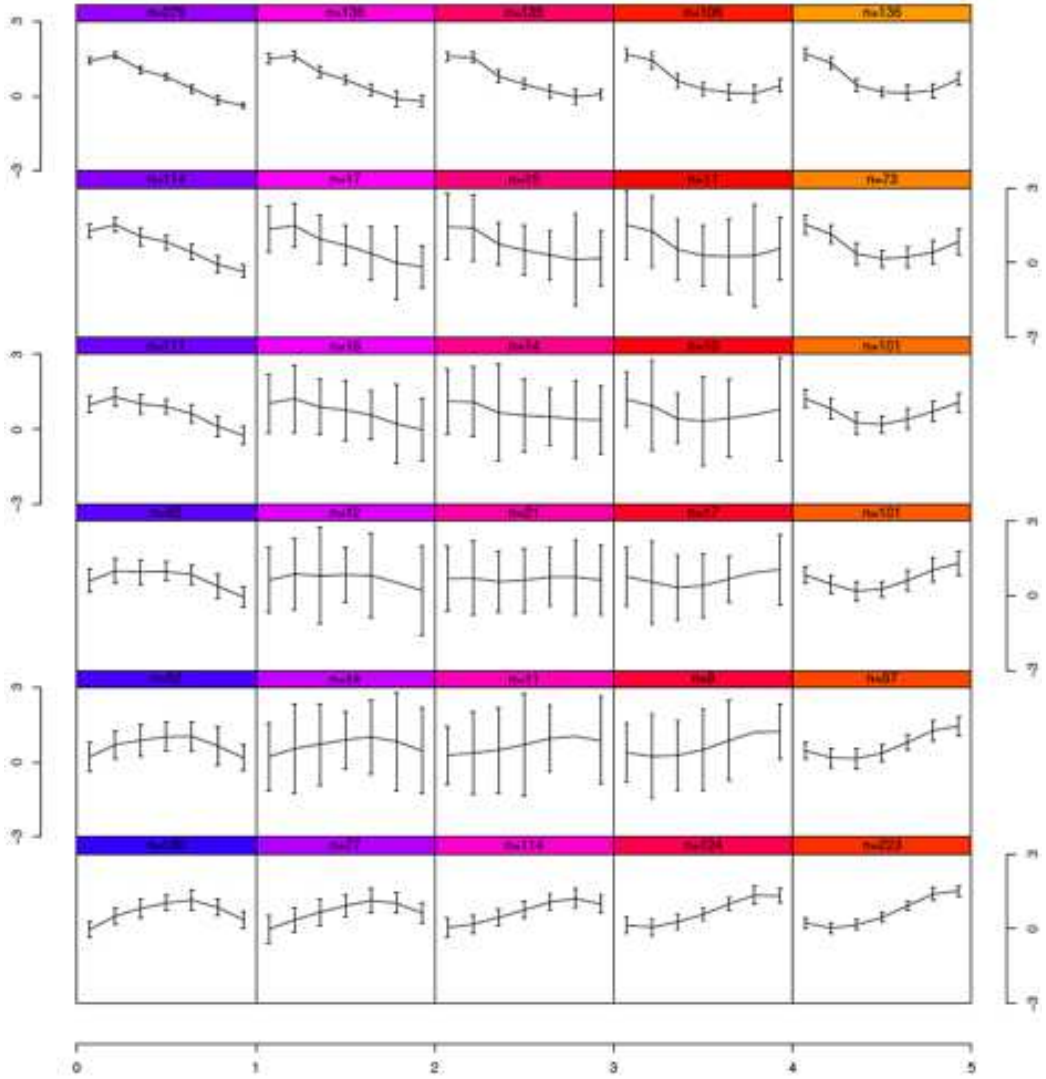
SOM Example

- SOM with 2x2 grid (Toxicology data):



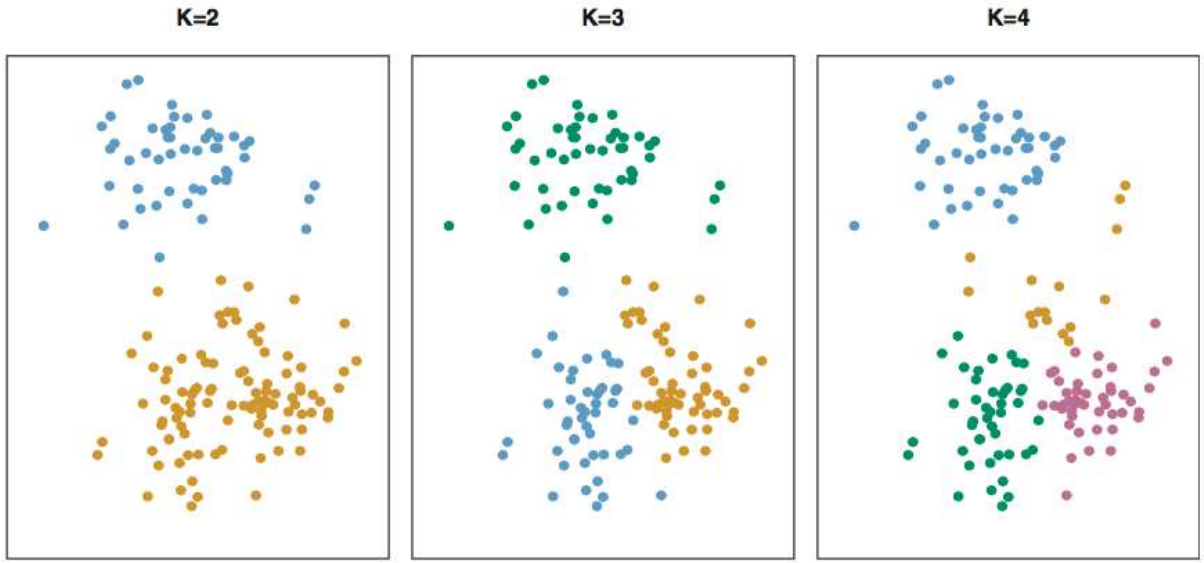
SOM Example

- SOM with 5x6 grid (Toxicology data):



Cluster Validity

- In addition to $J = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbf{r}_i(k) ||\mathbf{X}_i - \boldsymbol{\mu}_k||^2$ (the within-cluster error metric), there are many other validity metrics: Dunn index, Davis-Bouldin index, the root mean-square standard deviation (RMSSTD), and more.
- How to pick the number of clusters K?



- One could use validity indices to compare the results, or use a Bayesian approach.

Principal Component Analysis (PCA)

- PCA is based on the previously-mentioned heuristic according to which low-variance features should be avoided. Here, an extra step of feature decorrelation is applied first.
- After the decorrelation step, the first d individual (transformed) features with the largest variance are retained.
- Therefore, PCA uses the best individual features approach, with uncorrelated features.
- As with K-means clustering, we will discuss the basic (deterministic) version of PCA, and then extend it to a probabilistic setting based on a Latent Variable model (this is imilar to the Gaussian Mixture model).

Discrete Karhunen-Loève Transform

- Given $\mathbf{X} \in R^d$, we can always find a set of d orthonormal eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_d$ for the covariance matrix $\Sigma_{\mathbf{X}}$, corresponding to nonnegative eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$.

- Let $\mathbf{Z} = (Z_1, \ldots, Z_d)$ be given by the linear transformation

$$\mathbf{Z} = U^T(\mathbf{X} - \boldsymbol{\mu})$$

where $U = [\mathbf{u}_1 \ldots \mathbf{u}_d]$ and $\boldsymbol{\mu} = E[\mathbf{X}]$.

- Clearly,

$$E[\mathbf{Z}] = E\left[\mathbf{U}^T(\mathbf{X} - \boldsymbol{\mu})\right] = U^T\left(E[\mathbf{X}] - \boldsymbol{\mu}\right) = 0$$

so that the Z_i are all zero-mean, for $i = 1, \ldots, d$.

Discrete Karhunen-Loève Transform

- It follows that

$$\Sigma_{\mathbf{Z}} = E[\mathbf{Z}\mathbf{Z}^T] = E\left[U^T(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^T U\right]$$
$$= U^T E\left[(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^T\right] U = U^T \Sigma_{\mathbf{X}} U = \Lambda$$

  where Λ is the diagonal matrix of eigenvalues $\lambda_1, \ldots, \lambda_d$.

- Therefore,

$$E[Z_i Z_j] = 0, \quad \text{for } i \neq j$$

  that is, the Z_i are uncorrelated, and

$$\sigma_i^2 = \text{Var}(Z_i) = E[Z_i^2] = \lambda_i$$

  so the variance of Z_i is given by the corresponding eigenvalue λ_i.

Discrete Karhunen-Loève Transform

- The equations

$$Z_i = \mathbf{u}_i^T (\mathbf{X} - \boldsymbol{\mu}) , \;\; i = 1, \ldots, d$$

subject to

$$\Sigma_{\mathbf{X}} \, \mathbf{u}_i = \sigma_i^2 \, \mathbf{u}_i , \;\; i = 1, \ldots, d$$

define the discrete Karhunen-Loève transform.

- The discrete KL transform produces zero-mean, uncorrelated transformed features. This is similar to the *whitening* transformation:

$$\mathbf{Z} = \Lambda^{-\frac{1}{2}} U^T (\mathbf{X} - \boldsymbol{\mu})$$

except that the whitening transformation also normalizes all variances to unity.

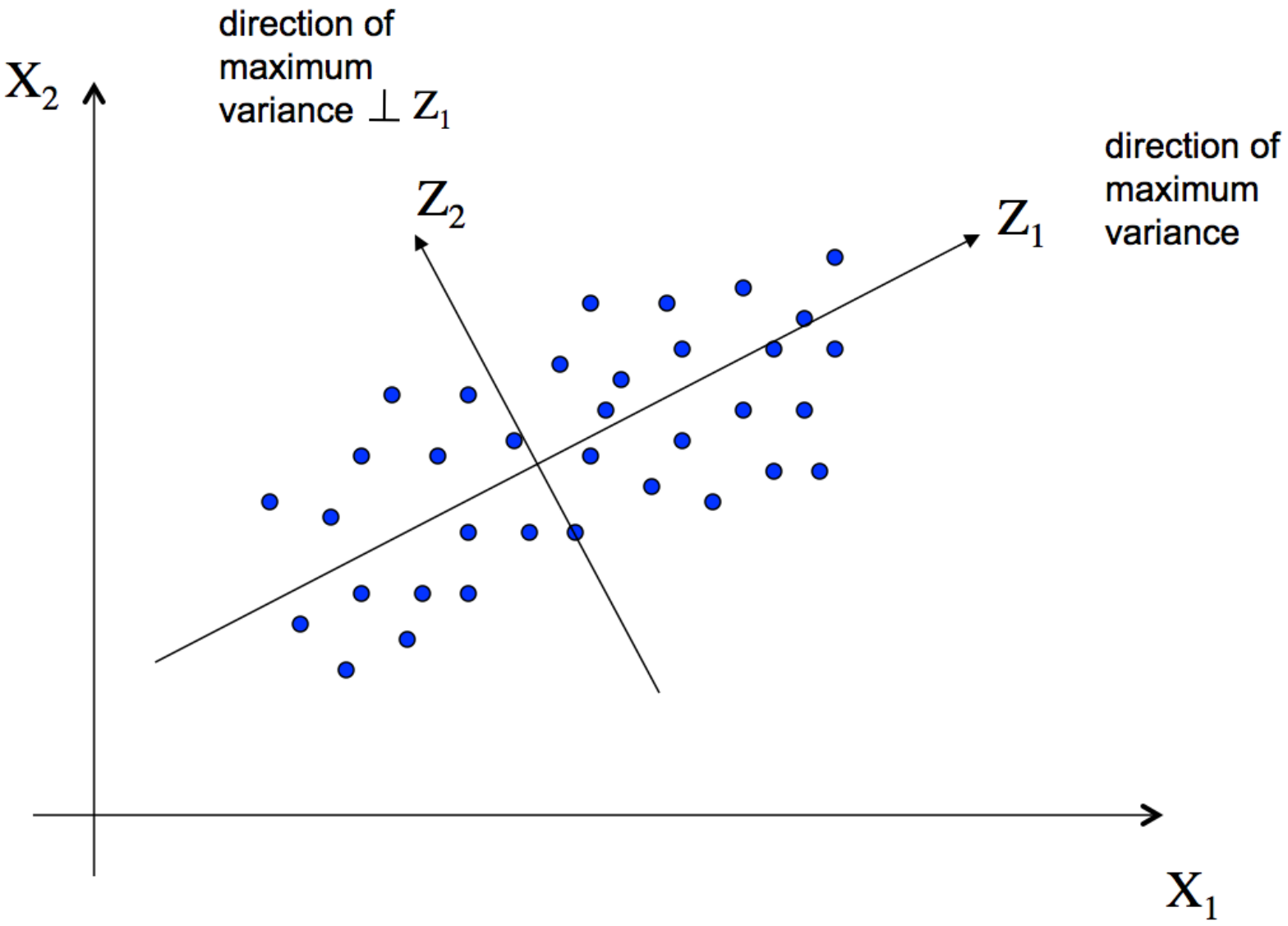
PCA Transform - Maximum Variance

- The component Z_i is the i-th *principal component*.
- The first PC Z_1 has the maximal variance λ_1, and the eigenvector u_1 points to the direction of maximal variation. The second PC Z_2 has the maximal variance in a direction perpendicular to u_1, while Z_3 has the maximal variance perpendicular to u_1 and u_2, and so on.
- The PCA transform $\mathbf{Z} = T(\mathbf{X})$ consists of applying the discrete KL transform and then keeping the first p principal components $\mathbf{Z} = (Z_1, \ldots, Z_p)$. In other words

$$\mathbf{Z} = W^T(\mathbf{X} - \boldsymbol{\mu})$$

where $W = [\mathbf{u}_1 \cdots \mathbf{u}_p]$ is a rank-p matrix (therefore PCA is not in general invertible and lossy with respect to the Bayes error criterion).

PCA Graphical Example

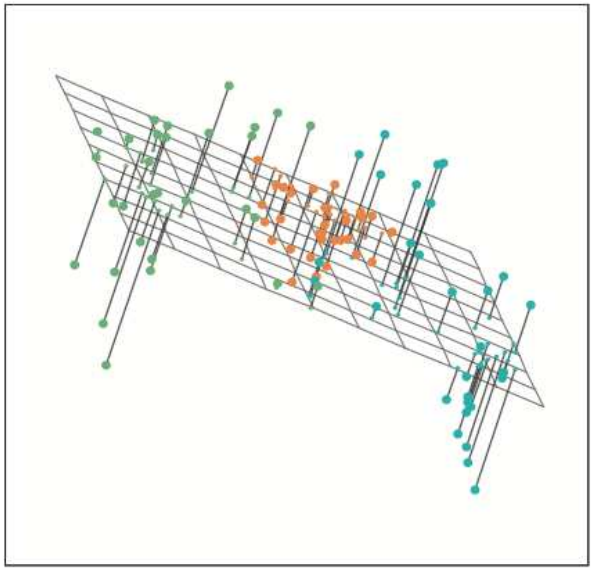


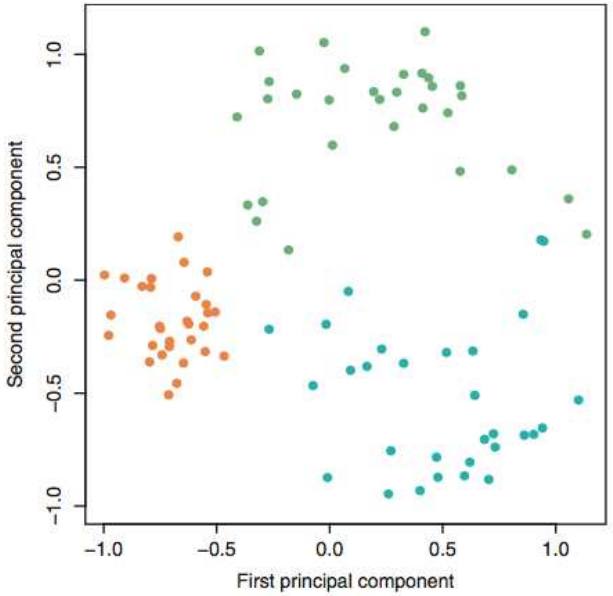
PCA Transform - Minimum Error

- An alternative interpretation of the PCA transform is that it is the linear projection $T : R^d \rightarrow R^p$ that minimizes

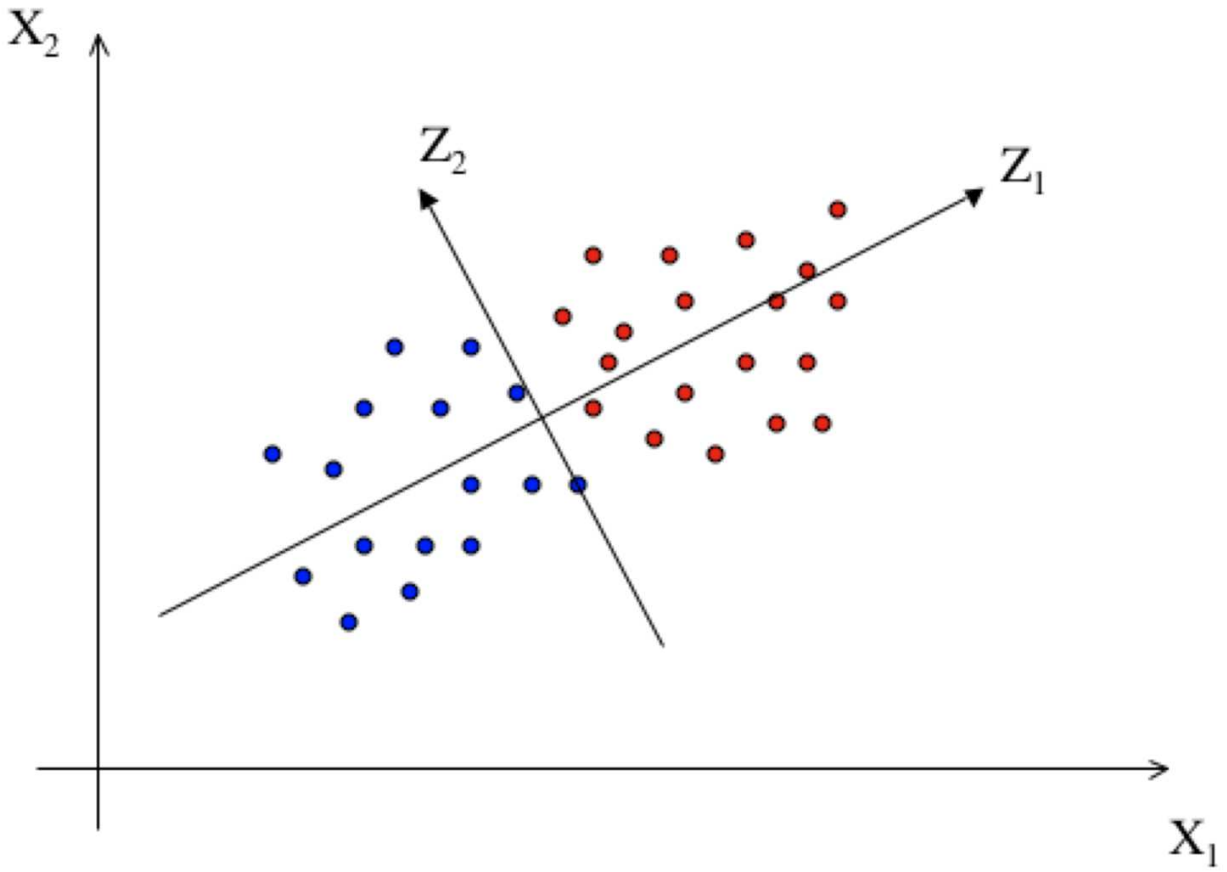
$$J = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{X}_i - T(\mathbf{X}_i)||^2$$

In fact, it can be shown that $J = \sum_{i=p+1}^{d} \lambda_i$ (the sum of discarded eigenvalues). Example (James et al., 2009):



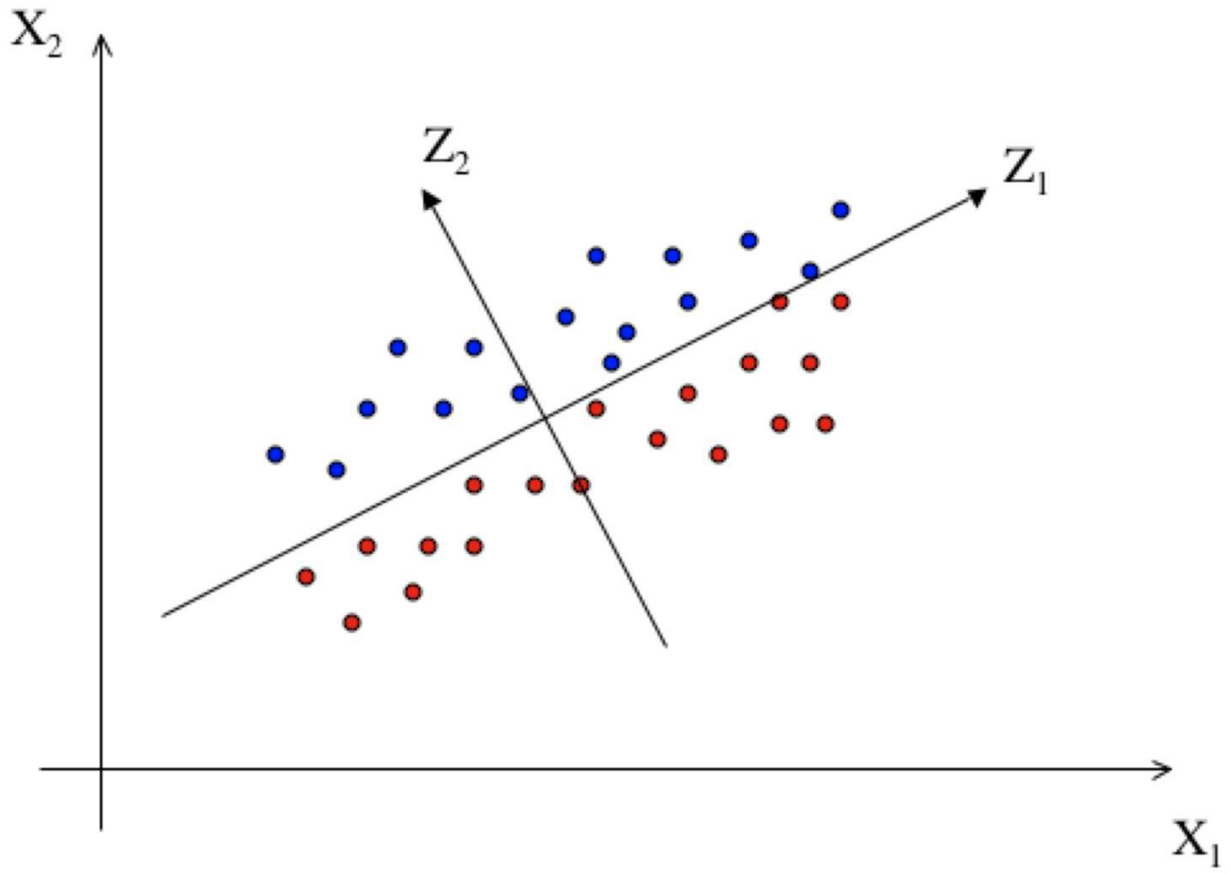


PCA - Group Structure



The first principal component Z_1 alone contains most of the discrimination information.

PCA - Group Structure



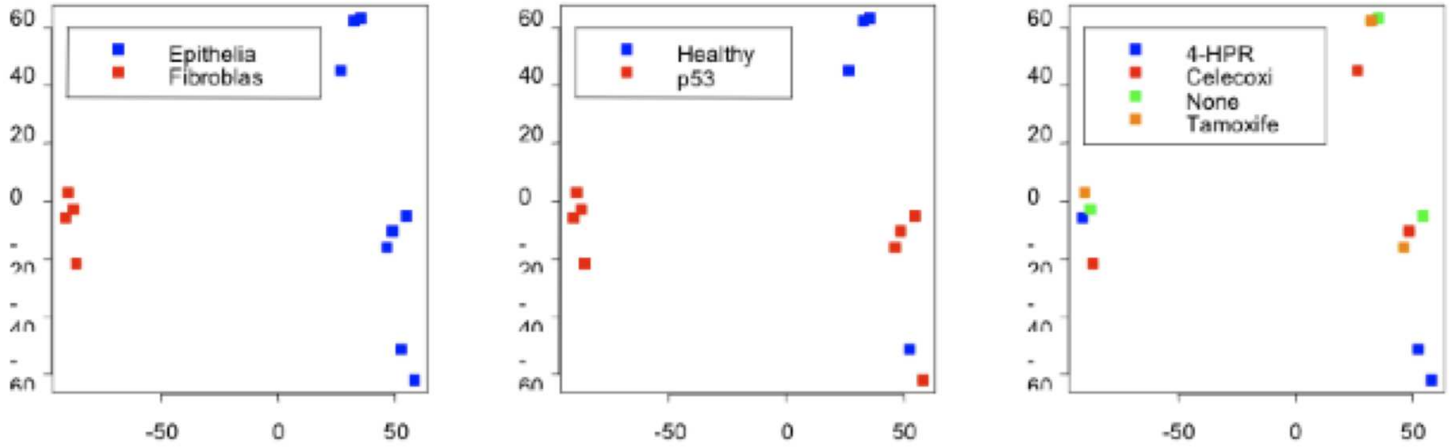
Here, the discrimination information is contained in the second principal component Z_2!

Real-Data 2-D PCA Example

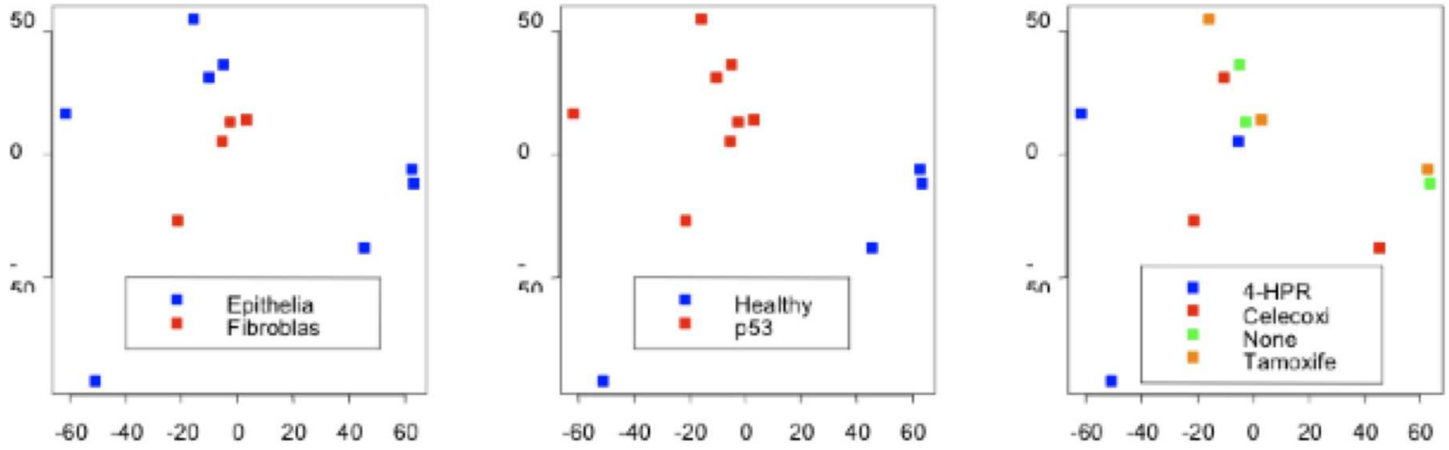
- Cancer chemotherapy study
- Gene expression data with 12 samples.
- Reduction from 12,573 initial genes to 2 features.
- Three groupings:
  - Cell type: Epithelial cells (8) vs. fibroblasts (4)
  - p53 status: “Healthy” patients (4) vs. p53-mutant patients (8)
  - Treatment: 4-HPR (3) vs. Tamoxifen (3) vs. Celecoxib (3) vs. none (3)
- Data produced by Louise Strong’s group, processed by Kevin Coombes – MD Anderson Cancer Center.

Real-Data 2-D PCA Example - II

- First PC (x axis) vs. Second PC (y axis)



- Second PC (x axis) vs. Third PC (y axis)



PCA Computation Issues

- In practice, the mean vector μ and covariance matrix $\Sigma_{\mathbf{X}}$ are approximated respectively by the sample mean and sample covariance matrix computed from data.
- In small-sample cases, the sample covariance matrix is a poor estimator of the true $\Sigma_{\mathbf{X}}$.
- In high-dimensional spaces, calculating the sample covariance matrix and diagonalizing is an intractable computational problem.
- Algorithms that try to avoid dealing directly with the sample covariance matrix can be beneficial, such as Probabilistic PCA (next).

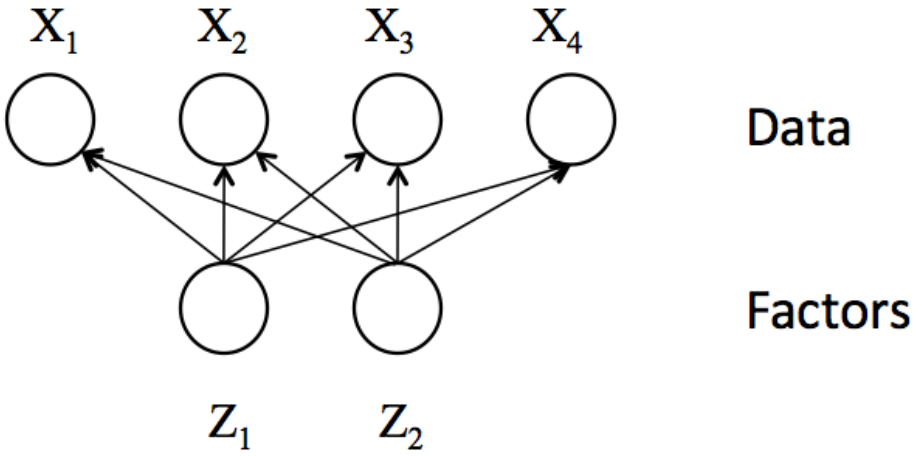
Factor Analysis

- Probabilistic PCA is a special case of Factor Analysis.

- Suppose one inverts the point of view of PCA, and consider the generation of the data from the components:

$$\mathbf{X} = W\mathbf{Z} + \boldsymbol{\mu}$$

where $W = [\mathbf{u}_1 \cdots \mathbf{u}_p]$ as before.



Factor Analysis

- In the Factor Analysis model, one considers a general rank-p $d \times p$ matrix C, called the "factor loading" matrix.
- To account for uncertainty, one adds a Gaussian error term ϵ and makes $\mathbf{Z}$ an isotropic Gaussian vector:

$$\mathbf{X} = C\mathbf{Z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon}$$

  where $\mathbf{Z} \sim \mathcal{N}(0, I_p)$ are the vector of factors, and $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \Psi)$ is a zero-mean error term.
- Clearly, the generative model is Gaussian, with

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, CC^T + \Psi)$$

- For probabilistic PCA, we assume that $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I)$.

EM Algorithm for Probabilistic PCA

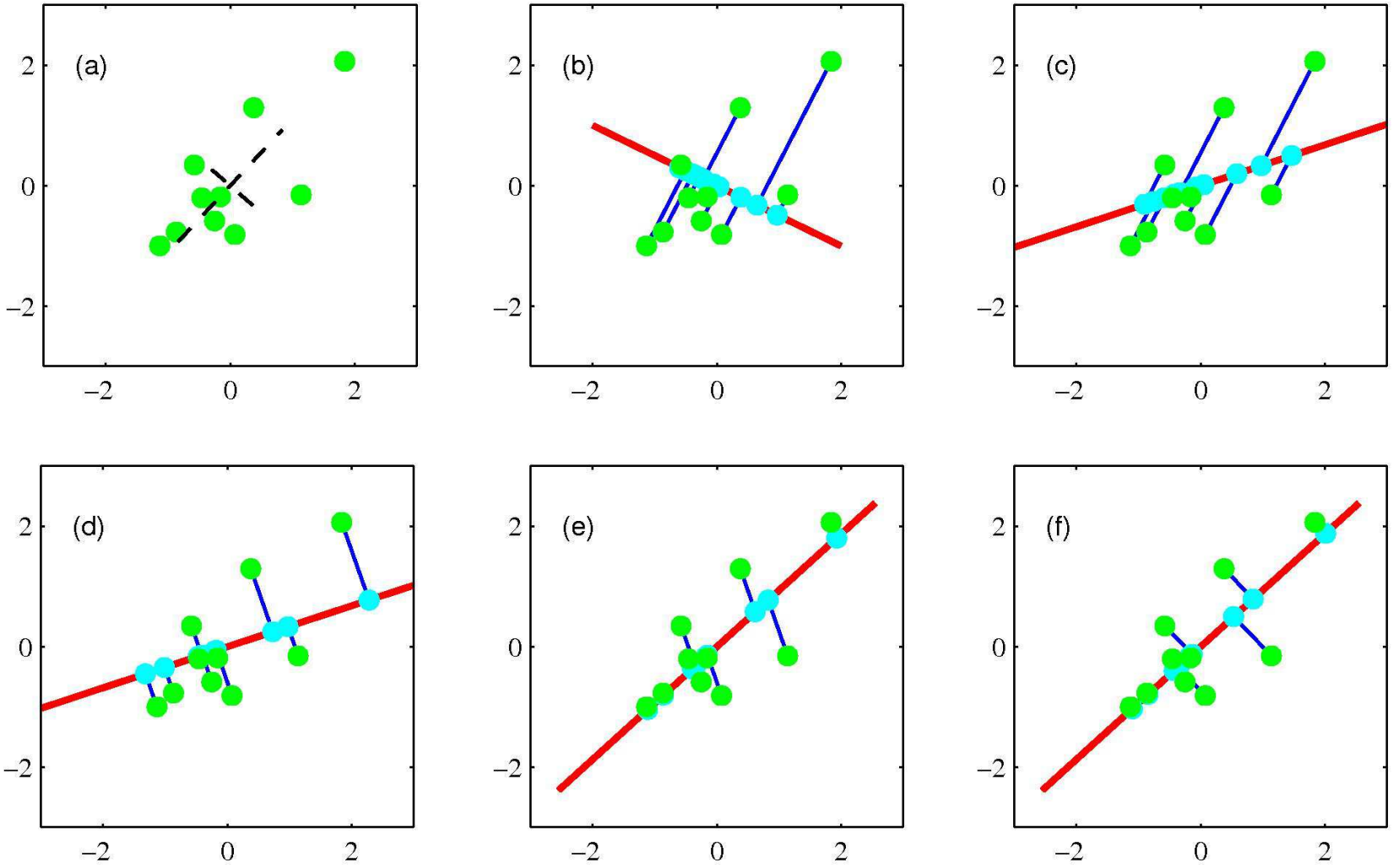
- The parameters C, $\boldsymbol{\mu}$, and σ need to be estimated from data.
- It can be shown that the maximum-likelihood solution is similar to the traditional PCA one. In particular, it requires finding the eigenvectors of the sample covariance matrix.
- However, the solution can also be obtained iteratively, using the EM algorithm, which *does not* require finding the eigenvectors of the sample covariance matrix.

EM Algorithm for Usual PCA

- As $\sigma \to 0$, Probabilistic PCA becomes usual PCA.
- One can use the EM algorithm with $\sigma = 0$ and find the usual PCA solution. Let $\mathbf{X} = [\mathbf{X}_1 \cdots \mathbf{X}_n]$ be the $d \times n$ data matrix (we assume zero mean data for simplicity) and let $\mathbf{Z} = [\mathbf{Z}_1 \cdots \mathbf{Z}_n]$ be a $p \times n$ matrix of projections of the data into the (current) principal subspace.
- Algorithm:
  - Choose an initial value $C_{(0)}$.
  - ("E-step") Project data: $\mathbf{Z}_{(n)} = (C_{(n)}^T C_{(n)})^{-1} C_{(n)}^T \mathbf{X}$
  - ("M-step") Update space: $C_{(n+1)} = \mathbf{Z}_{(n)} \mathbf{X}^T (\mathbf{X}\mathbf{X}^T)^{-1}$
  - End when $||C_{(n+1)} - C_{(n)}||_2^2$ is close enough to zero. The PCA matrix is $W = C_{(n+1)}$.

Example

- “Spring and Rods" physical simulation (C. Bishop).



Multidimensional Scaling

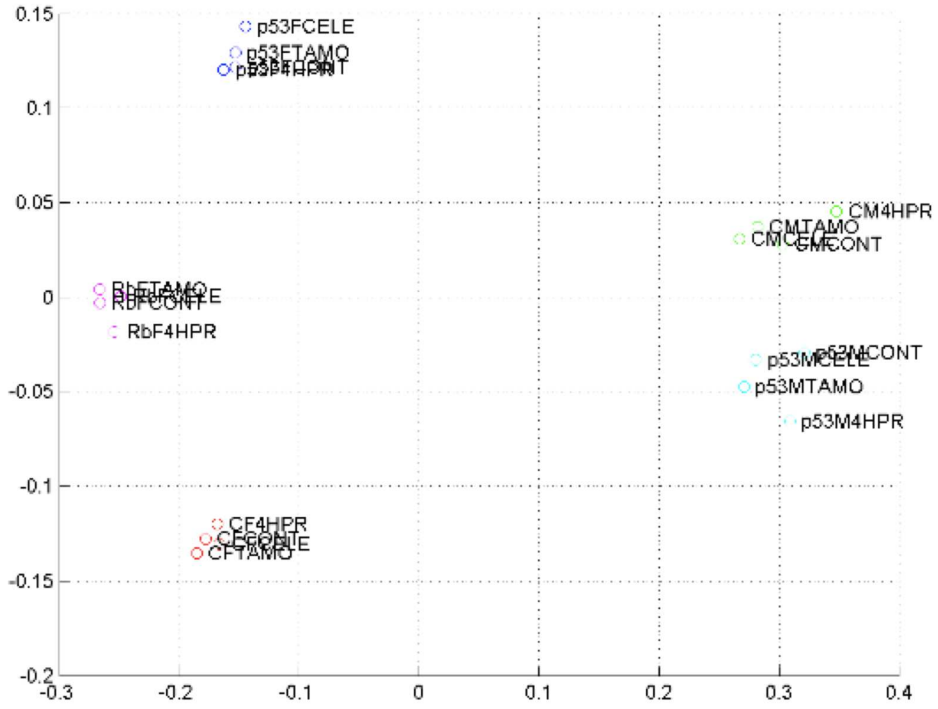
- The main idea is to find points in R^d that best approximate pairwise dissimilarities (e.g., Euclidean distances, $1-$correlation) in the original space R^p.
- If δ_{ij} and d_{ij} are the dissimilarities between original and transformed points, respectively, the goodness of fit can be measured by the *stress* (values < 10% are good):

$$S = \sqrt{\frac{\sum_{i,j}(\delta_{ij}^2 - d_{ij}^2)}{\sum_{i,j} d_{ij}^2}}$$

- This is nonlinear feature extraction, which can be advantageous over linear methods such as PCA.
- The main issues are that it is unsupervised, and it is not simple to express $T(X)$ to apply to a new sample point.

Real-Data 2-D MDS Example

- Data from previous cancer study (with 8 new samples).
- Reduction from 904 initial genes to 2 features.
- Processed by our group. Stress = 4.64%



Real-Data 3-D MDS Example

- Reduction of same data to 3 features. Stress = 1.83%