Clustering: Part I

MATH 6312
Department of mathematics, UTA

ESL 14.3.4-8, 14.3.10

Optional reading: ESL 14.3.1-2

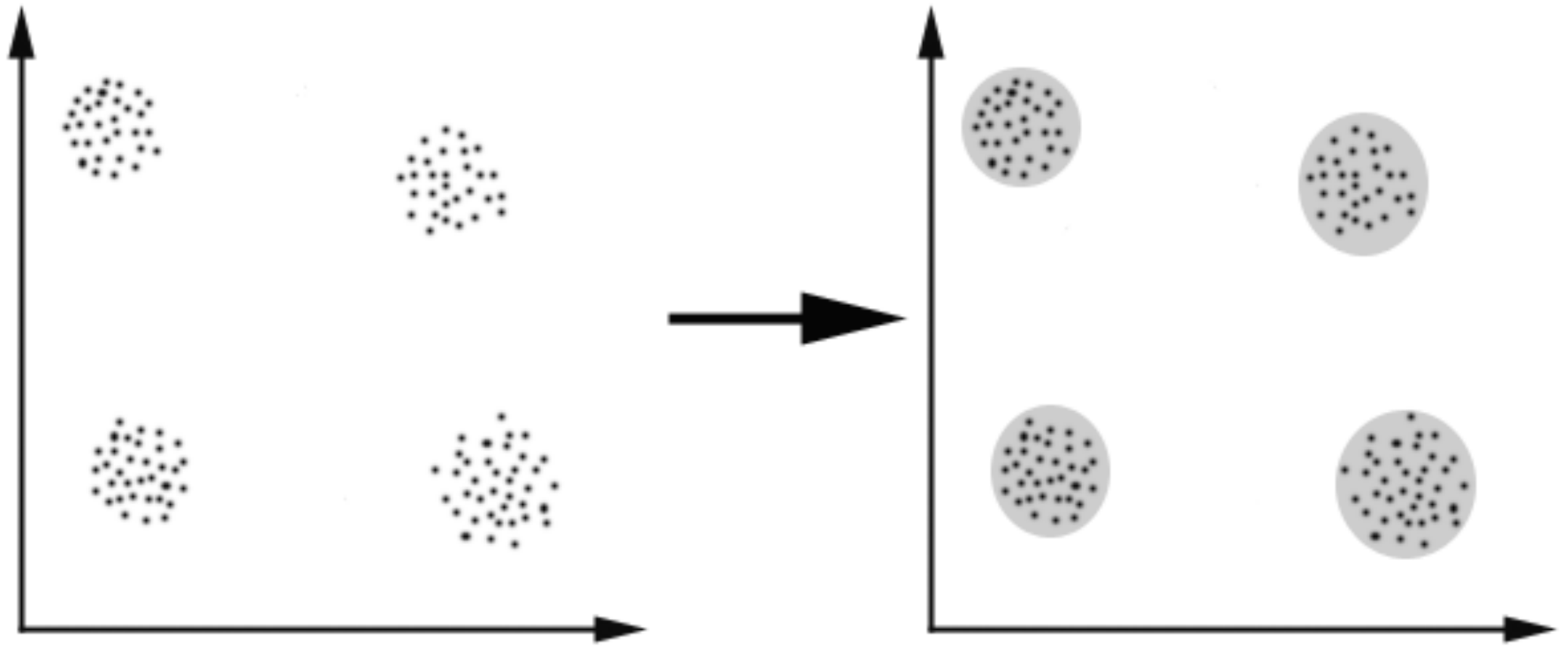
What/Why is clustering?

- Clustering: task of dividing up data into groups (clusters), so that points in any one group are more similar to each other than to points outside the group

- Why cluster? Two main uses
  - Summary: deriving a reduced representation of the full data set.
  - Discovery: looking for new insights into the structure of the data. e.g., finding groups of Netflix users with similar movie preference.

- Other reasons are
  - Investigating the validity of pre-existing group assignments
  - Helping with prediction (classification or regression)

Which points are similar?



- Credit: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/

Clustering vs classification

- Classification (learning with teacher):
  - We have data for which the groups are known, and we try to learn what differentiates these groups (classifier) to properly classify future data.
- Clustering (learning without teacher):
  - We look at data for which groups are unknown and undefined, and try to learn the groups themselves, as well as what differentiates them.

Clustering algorithms

- Suppose the number of clusters K < N is pre-specified. C(i) = k ∈ {1,2,…,K} is an encoder that assigns the i*th* observation to the k*th* cluster.

- We seeks the particular encoder that minimizes the within-point scatter (i.e. sum dissimilarities with clusters)

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

- This is computationally infeasible as there are many possible cluster assignments.

- Feasible strategies are based on iterative greedy descent (examining a small fraction of all possible assignments)

  - At each step, the cluster assignments are changed in such a way that the value of the criterion is improved from its previous value.

  - When the prescription is unable to provide an improvement, the algorithm terminates with the current assignments as its solution.

K-means

- K-means clustering starts with guesses (cluster centers), and it repeats the following steps:

- For each data point, the closest cluster center (in Euclidean distance) is identified;

- Each cluster center is replaced by the average of all data points that are closest to it.

- Stop when the cluster assignment does not change.

- Credit: http://shabal.in/visuals/kmeans/2.html
- Visualization of K-means convergence with 4 starting points.

Details on K-means

- The K-means algorithm is one of the most popular iterative descent clustering methods.

- Squared distance is used as the dissimilarity measure

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

- Within-point scatter (N_k: # of points in k*th* cluster):

$$\begin{aligned} W(C) &= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2 \\ &= \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2, \end{aligned}$$

- The cluster assignment is obtained by solving the enlarged optimization:

$$\bar{x}_S = \underset{m}{\operatorname{argmin}} \sum_{i \in S} ||x_i - m||^2. \tag{14.32}$$

$$\min_{C, \{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - m_k||^2. \tag{14.33}$$

Algorithm 14.1 *K-means Clustering.*

1. For a given cluster assignment C, the total cluster variance (14.33) is minimized with respect to $\{m_1, \ldots, m_K\}$ yielding the means of the currently assigned clusters (14.32).

2. Given a current set of means $\{m_1, \ldots, m_K\}$, (14.33) is minimized by assigning each observation to the closest (current) cluster mean. That is,

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} ||x_i - m_k||^2. \tag{14.34}$$

3. Steps 1 and 2 are iterated until the assignments do not change.

Remarks on K-means

- Within-point scatter decreases with each iteration of the algorithm. (why?)
- The final clustering depends on the initial cluster centers.
  - We typically run K-means multiple times with random initial centers, then choose among from collection of centers based on which one gives the smallest within-point scatter.
- The algorithm is not guaranteed to deliver the clustering that globally minimizes within-cluster variation.

Soft K-means

- Goal: to make probabilistic (rather than deterministic) assignments of points to cluster.

- K-means is closely related to the EM (expectation-maximization) algorithm to solve the Gaussian mixture.

Gaussian mixture

- Suppose there are two clusters (K=2). Let $\Delta_i \in \{0,1\}$ be the class membership variable.

- Given Δ_i, $y_i \in \mathbb{R}^p$ (data point) follow a normal distribution with parameters unknown.

$$[y_i \mid \Delta_i = 0] \sim N(\mu_1, \Sigma_1)$$
$$[y_i \mid \Delta_i = 1] \sim N(\mu_2, \Sigma_2)$$

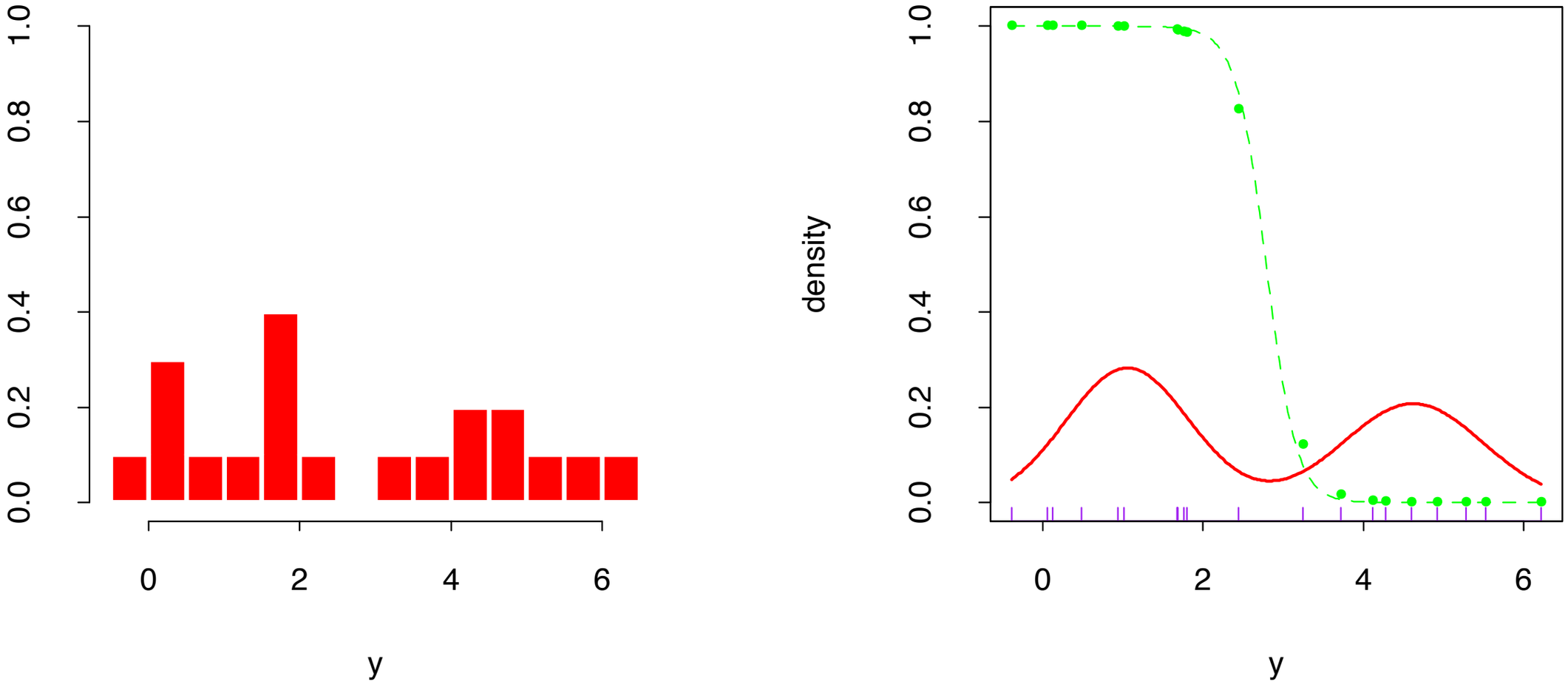
- The class membership Δ_i is not observable. Then y follows the two component Gaussian mixture:

$$p(y) = p(\Delta = 0)p(y \mid \Delta = 0) + p(\Delta = 1)p(y \mid \Delta = 1)$$
$$= (1-\pi)\cdot\phi_{\theta_1}(y) + \pi\cdot\phi_{\theta_2}(y)$$

-

$$\theta = (\pi, \theta_1, \theta_2) = (\pi, \mu_1, \Sigma_1, \mu_2, \Sigma_2)$$
$$\pi = P(\Delta = 1)$$
$$\phi_{\theta_k}(\cdot) \text{ is the Guassian density}$$

Gaussian mixture



❖ FIGURE 8.5. Mixture example. (Left panel:) Histogram of data. (Right panel:) Maximum likelihood fit of Gaussian densities (solid red) and responsibility (dotted green) of the left component density for observation y, as a function of y.

- Let Z = (y_1,y_2,…,y_N). Then, θ in Gaussian mixture can be estimated by maximizing the (observed) log likelihood:

$$\ell(\theta; \mathbf{Z}) = \sum_{i=1}^{N} \log[(1-\pi)\phi_{\theta_1}(y_i) + \pi\phi_{\theta_2}(y_i)].$$

- Maximization is difficult numerically, because of the sum of terms inside the logarithm (try to calculate the 1st derivative).

- Instead, we argument data with unobserved variable $\Delta = (\Delta_1,\Delta_2,\ldots,\Delta_N)$. Then, the (complete) log-likelihood is

$$\begin{aligned}\ell_0(\theta;\mathbf{Z},\boldsymbol{\Delta}) &= \sum_{i=1}^{N}\left[(1-\Delta_i)\log\phi_{\theta_1}(y_i)+\Delta_i\log\phi_{\theta_2}(y_i)\right] \\ &\quad + \sum_{i=1}^{N}\left[(1-\Delta_i)\log(1-\pi)+\Delta_i\log\pi\right],\end{aligned}$$

- Since the values of Δ_i's are actually unknown, we substitute for each Δ_i its expected value (also called responsibility)

$$\gamma_i(\theta) = \mathrm{E}(\Delta_i|\theta,\mathbf{Z}) = \Pr(\Delta_i = 1|\theta,\mathbf{Z})$$

- On It is easy to maximize the complete likelihood to estimate θ. Wait, we do not know θ to calculate the expected value…

Algorithm 8.1 *EM Algorithm for Two-component Gaussian Mixture.*

1. Take initial guesses for the parameters $\hat{\mu}_1, \hat{\sigma}_1^2, \hat{\mu}_2, \hat{\sigma}_2^2, \hat{\pi}$ (see text).

2. *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \ i = 1, 2, \ldots, N. \tag{8.42}$$

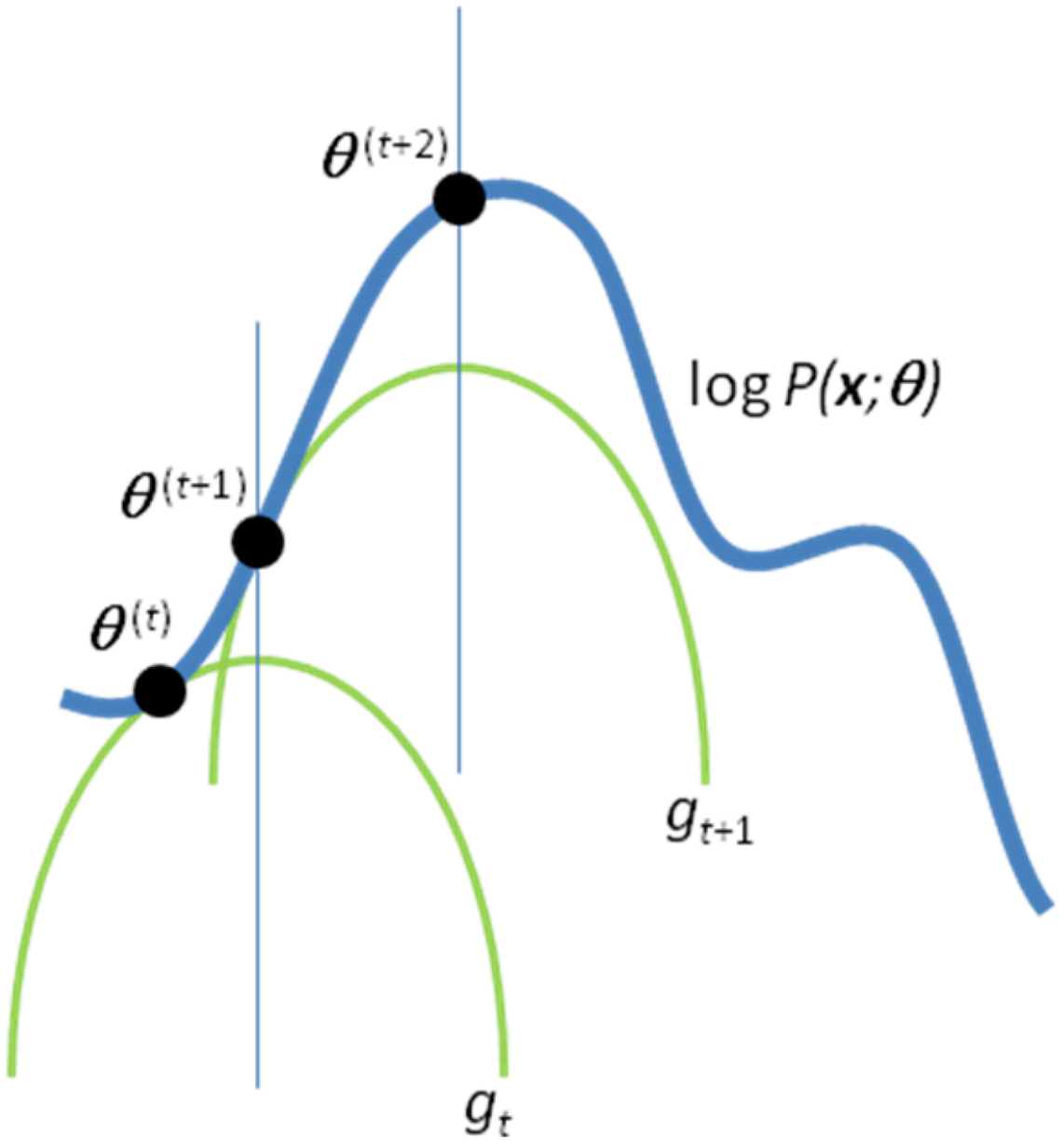
3. *Maximization Step*: compute the weighted means and variances:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i) y_i}{\sum_{i=1}^N (1-\hat{\gamma}_i)}, \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i)(y_i - \hat{\mu}_1)^2}{\sum_{i=1}^N (1-\hat{\gamma}_i)},$$

$$\hat{\mu}_2 = \frac{\sum_{i=1}^N \hat{\gamma}_i y_i}{\sum_{i=1}^N \hat{\gamma}_i}, \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^N \hat{\gamma}_i (y_i - \hat{\mu}_2)^2}{\sum_{i=1}^N \hat{\gamma}_i},$$

   and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

4. Iterate steps 2 and 3 until convergence.



Supplementary Figure 1 Convergence of the EM algorithm. Starting from initial parameters $\theta^{(t)}$, the E-step of the EM algorithm constructs a function g_t that lower-bounds the objective function $\log P(\boldsymbol{x};\boldsymbol{\theta})$. In the M-step, $\theta^{(t+1)}$ is computed as the maximum of g_t. In the next E-step, a new lower-bound g_{t+1} is constructed; maximization of g_{t+1} in the next M-step gives $\theta^{(t+2)}$, etc.

Image from http://www.nature.com/nbt/journal/v26/n8/extref/nbt1406-S1.pdf

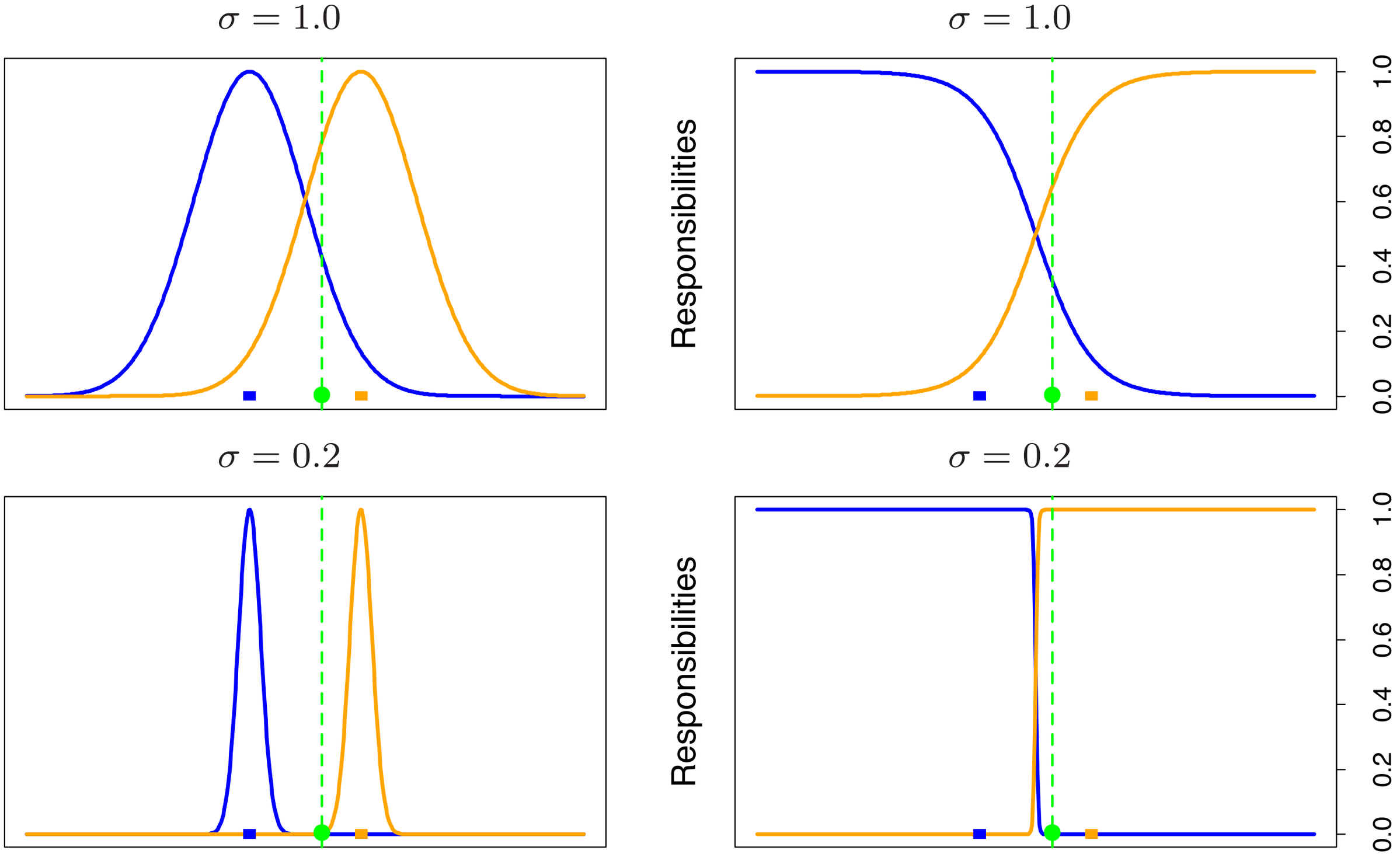
EM and soft K-means

- This set-up of EM is a soft version of K-means clustering if $\Sigma_1=\Sigma_2=\sigma^2\cdot I_p$.

- The E-step assigns responsibilities for each point, while the M-step recomputes the parameters based on current responsibilities.

$$\gamma_i(\theta) = \mathrm{E}(\Delta_i|\theta, \mathbf{Z}) = \Pr(\Delta_i = 1|\theta, \mathbf{Z})$$

- As σ goes to 0, responsibilities become 0 and 1, and then the EM becomes equivalent to K-means.

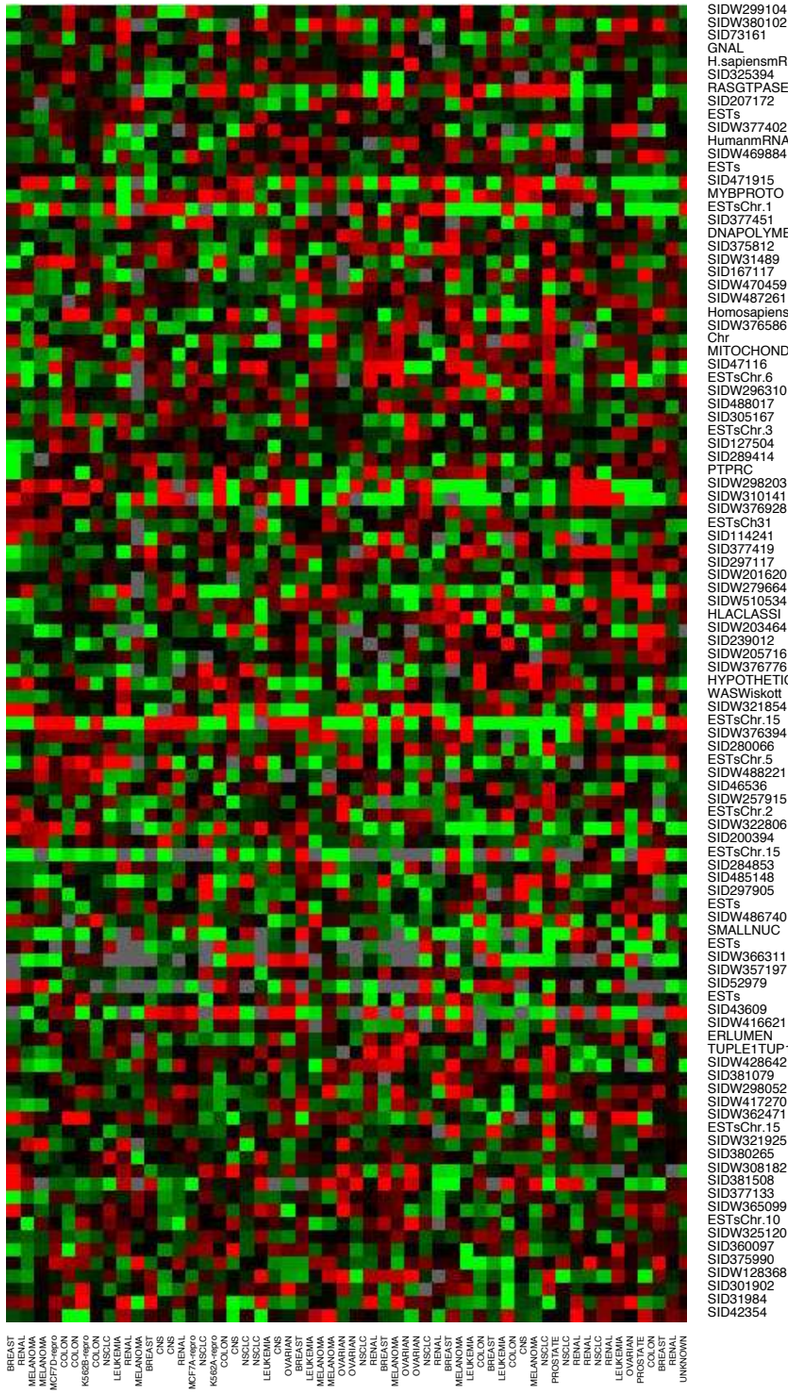


- ESL 14.7: soft (top) vs hard (bottom) assignments. At the x=0.5 (green line), the responsibility becomes 1 with a small variance.

Human tumor microarray data

- 6830 genes in row and 64 samples in column.
- K-means clustering is applied with K=3.

Cluster	Breast	CNS	Colon	K562	Leukemia	MCF7
1	3	5	0	0	0	0
2	2	0	0	2	6	2
3	2	0	7	0	0	0
Cluster	Melanoma	NSCLC	Ovarian	Prostate	Renal	Unknown
1	1	7	6	2	9	1
2	7	2	0	0	0	0
3	0	0	0	0	0	0



Shortcoming of K-means

- As the number of clusters K is changed, the cluster memberships can change in arbitrary ways.
  - That is, with say four clusters, the clusters need not be nested within the three clusters above.
- For these reasons, hierarchical clustering is probably preferable for this application.

K-medoids clustering

- K-means requires all variables to be quantitative variables (Euclidean distance).

- Using the squared Euclidean distance for the dissimilarity measure D(x,x′), K-means is sensitive to outliers.

- What if we use D(x,x′) other than the squared Euclidean distance?

Algorithm 14.2 *K-medoids Clustering.*

1. For a given cluster assignment C find the observation in the cluster minimizing total distance to other points in that cluster:

$$i_k^* = \underset{\{i:C(i)=k\}}{\operatorname{argmin}} \sum_{C(i')=k} D(x_i, x_{i'}). \tag{14.35}$$

   Then $m_k = x_{i_k^*}$, $k = 1, 2, \ldots, K$ are the current estimates of the cluster centers.

2. Given a current set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \underset{1\le k\le K}{\operatorname{argmin}}\, D(x_i, m_k). \tag{14.36}$$

3. Iterate steps 1 and 2 until the assignments do not change.

Remark on K-medoids

- The K-medoids algorithm repeat the following steps:
  - 1. Cluster (label) each point based on the closest center;
  - 2. Replace each center by the medoid of points in its cluster
- K-medoids is computationally harder than K-means (because of computing the medoid is harder than computing the average).
- K-medoids has the (potentially important) property that the centers are located among the data points themselves.

Take home messages

- In clustering we segment our data points into clusters. We want points in any one group to be more similar to each other than to other points.

- Fixing the number of clusters K, the task of exactly minimizing the within-point scatter is not feasible. The K-means algorithm approximately minimizes this by iterating two simple steps.

- The K-medoids algorithm is an alternative where the centers are chosen among the points themselves.

- K-medoids and K-means depend on the starting configuration. Hence for either algorithm, one should run algorithm several times with different starts.

Clustering: part II

MATH 6312
Department of mathematics, UTA

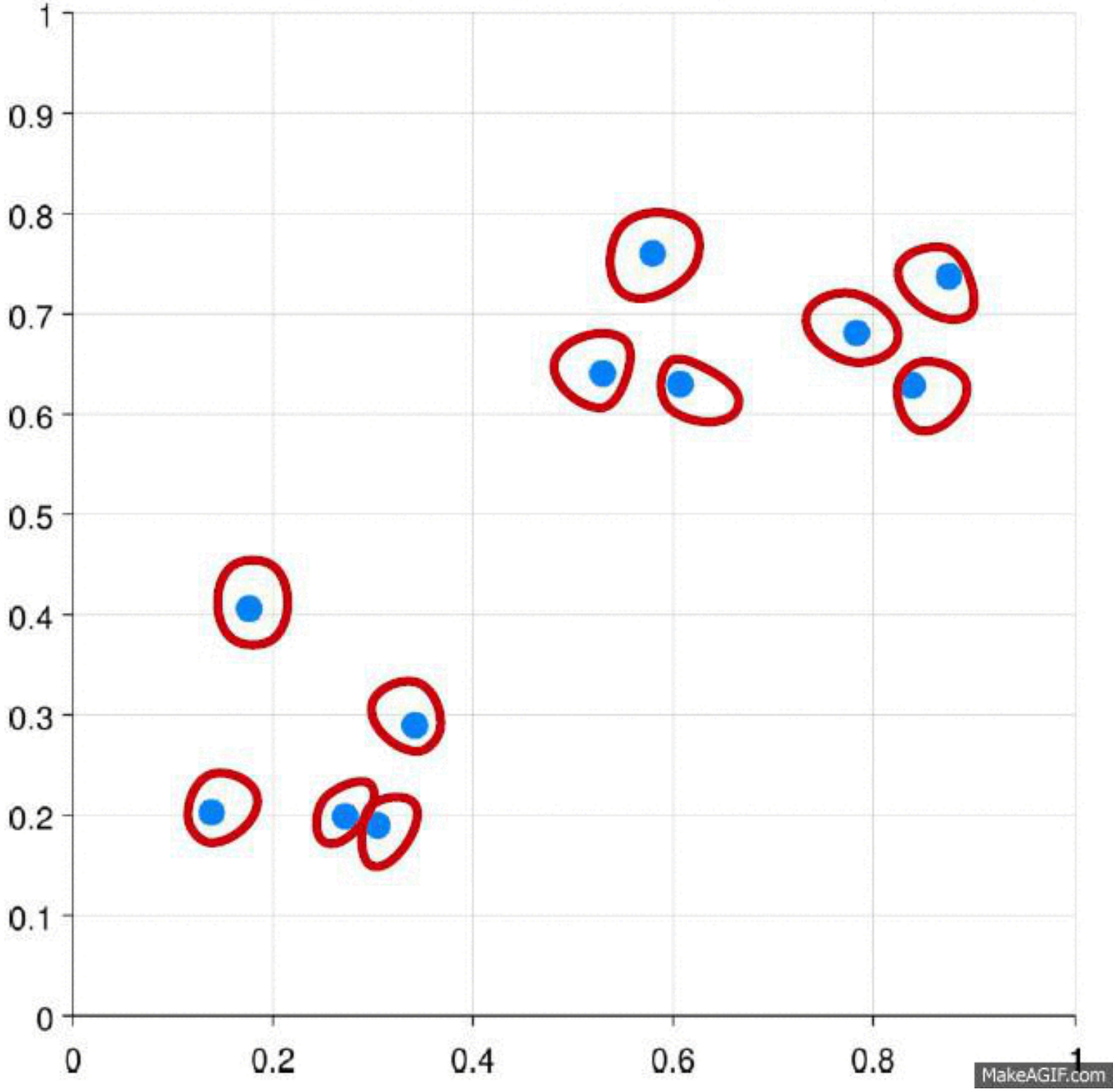
ESL 14.3

Hierarchical clustering

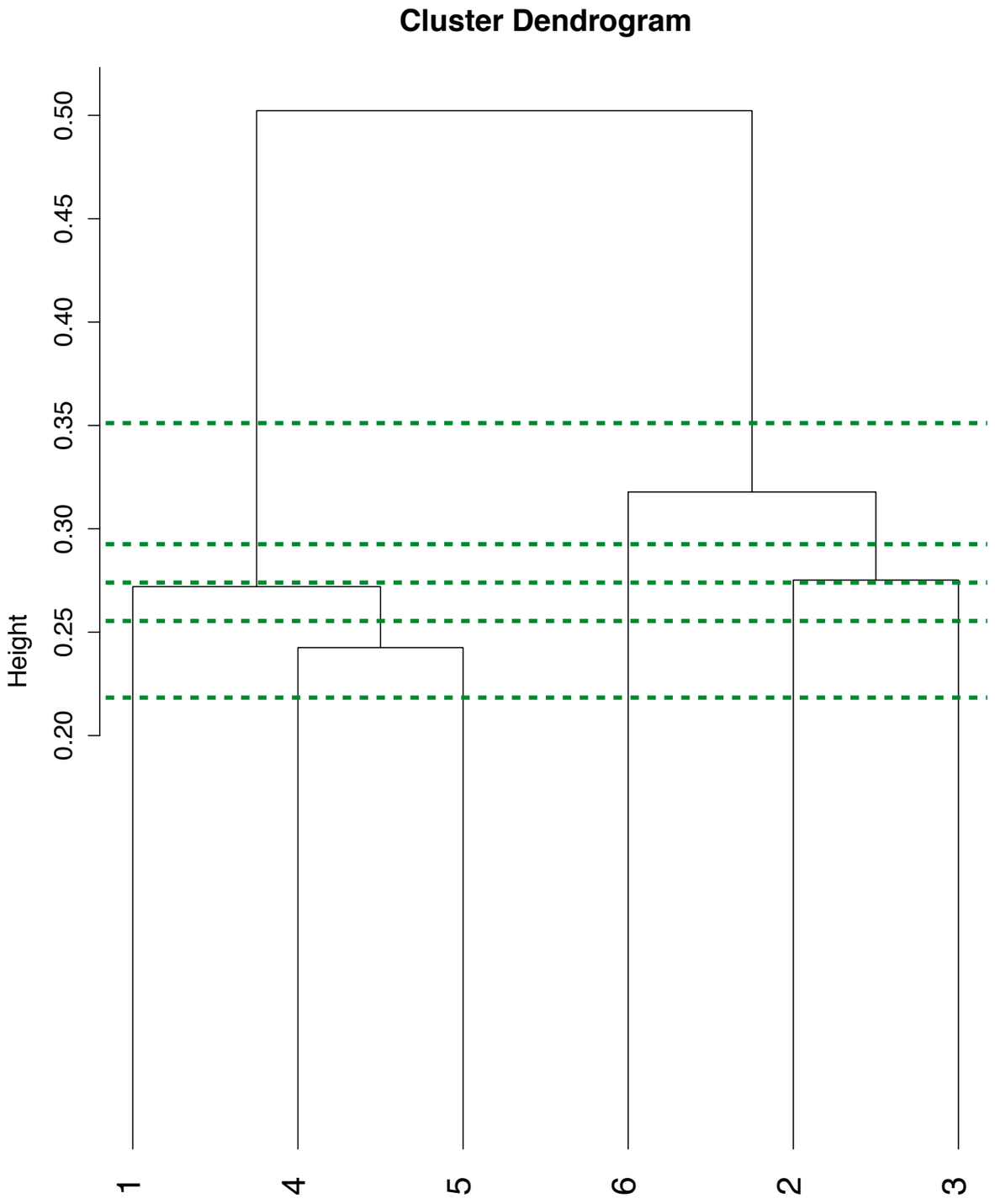
- Recall two properties of K-means (K-medoids) clustering:
  - 1. It fits exactly K clusters (as pre-specified)
  - 2. Final clustering assignment depends on the chosen initial cluster centers
- Given pairwise dissimilarities between data points, hierarchical clustering produces a consistent result, without the need to choose initial starting positions (number of clusters)
- We need to choose a way to measure the dissimilarity between groups, called the linkage
- Given the linkage, hierarchical clustering produces a sequence of clustering assignments. At one end, all points are in their own cluster, at the other end, all points are in one cluster

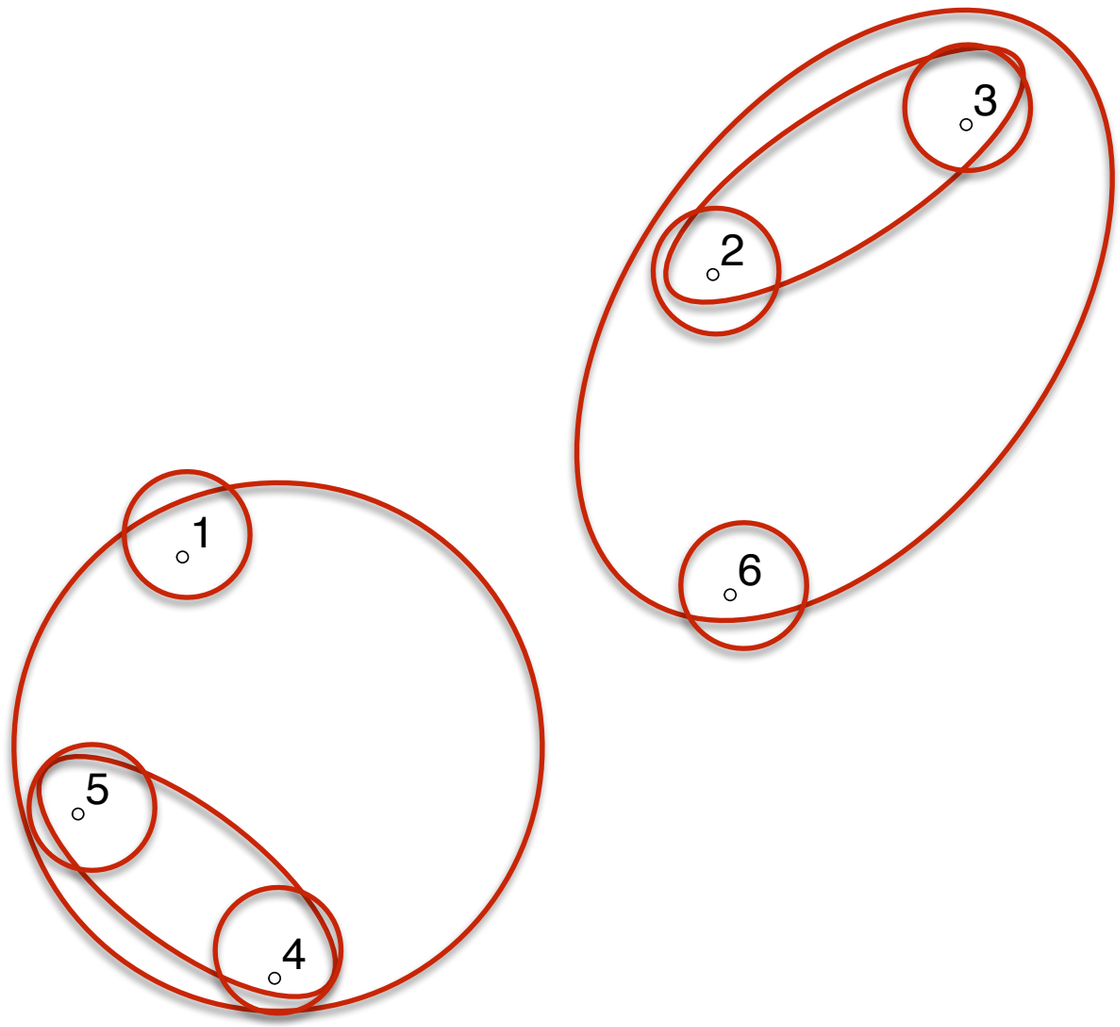
Agglomerative vs divisive

- Two types of hierarchical clustering algorithms
- Agglomerative (bottom-up):
  - Start with all points in their own group
  - Until there is only one cluster, repeatedly: merge the two groups that have the smallest dissimilarity
- Divisive (top-down):
  - Start with all points in one cluster
  - Until all points are in their own cluster, repeatedly: split the group into two resulting in the biggest dissimilarity



- Credit: http://makeagif.com/i/DkJOLy
- Illustration of agglomerative hierarchical clustering





- We can also represent the sequence of clustering assignments as a dendrogram.
- Note that cutting the dendrogram horizontally partitions the data points into clusters

What's a dendrogram?

- Dendrogram: convenient graphic to display a hierarchical sequence of clustering assignments. This is simply a tree where:
  - Each node represents a group
  - Each leaf node is a singleton (i.e., a group containing a single data point)
  - Root node is the group containing the whole data set
  - Each internal node has two daughter nodes (children), representing the groups that were merged to form it
- Remember: the choice of linkage determines how we measure dissimilarity between groups of points

Linkages

- At any level, clustering assignments can be expressed by sets G = $\{i_1, i_2, \ldots, i_r\}$, giving indices of points in this group.
- Let n_G be the size of G (here $N_G = r$). Bottom level: each group looks like G = {i}, top level: only one group, G = {1,...n}
- Linkage: function d(G,H) that takes two groups G,H and returns a dissimilarity score between them
- Agglomerative clustering, given the linkage:
  - Start with all points in their own group
  - Until there is only one cluster, repeatedly: merge the two groups G,H such that d(G,H) is smallest

Three linkages

- In single linkage, the dissimilarity between G and H is calculated based on the closest (least dissimilar) pair.

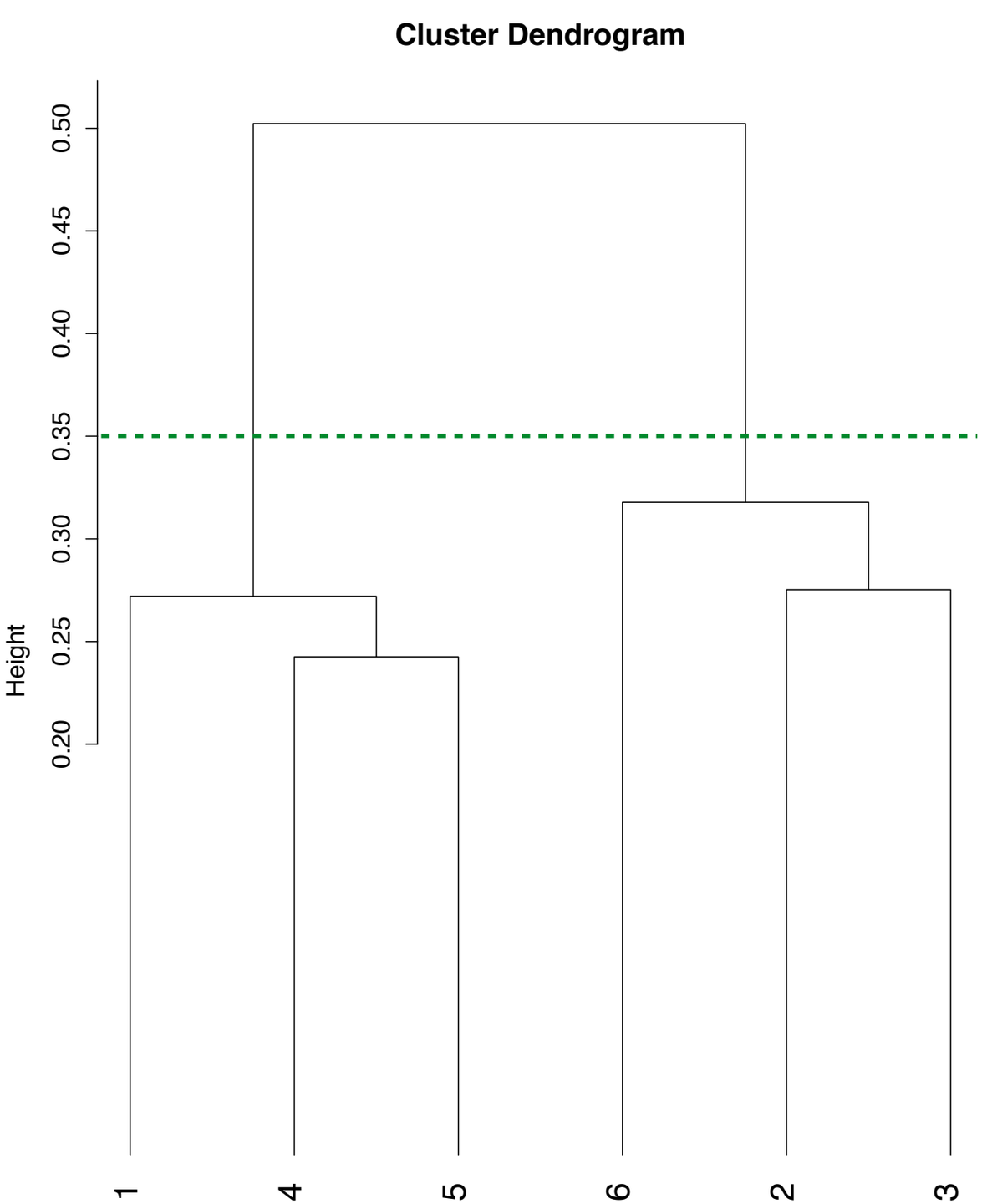
$$d_{SL}(G,H) = \min_{\substack{i\in G \\ i'\in H}} d_{ii'}.$$

- In complete linkage, dissimilarity between G and H is calculated based on the farthest (most dissimilar) pair.

$$d_{CL}(G,H) = \max_{\substack{i\in G \\ i'\in H}} d_{ii'}.$$

- In average linkage, the dissimilarity between G and H is the average dissimilarity over all points in opposite groups.

- $$d_{GA}(G,H) = \frac{1}{N_G N_H} \sum_{i\in G} \sum_{i'\in H} d_{ii'}$$



- Cut-interpretation:
  - Simple linkage: for each point X_i, there is another point X_j in its cluster with $d_{ij} \leq 0.35$
  - Complete linkage: for each point X_i, every other point X_j in its cluster satisfies $d_{ij} \leq 0.35$
  - Average linkage: ???

Remarks on linkages

- Single, complete, average linkage share the following properties:

- These linkages operate on dissimilarities $d_{ii'}$, and don't need the points $X_1,\ldots,X_n$ to be in Euclidean space

- Running agglomerative clustering with any of these linkages produces a dendrogram with no inversions

  - Dissimilarity scores between merged clusters only increases as we run the algorithm

  - It means that we can draw a proper dendrogram, where the height of a parent is always higher than height of its daughters

Shortcomings of single, complete linkage

- Single and complete linkage can have some practical problems:
  - Single linkage suffers from chaining. In order to merge two groups, only need one pair of points to be close, irrespective of all others. Therefore clusters can be too spread out, and not compact enough
  - Complete linkage avoids chaining, but suffers from crowding. Because its score is based on the worst-case dissimilarity between pairs, a point can be closer to points in other clusters than to points in its own cluster. Clusters are compact, but not far enough apart
- Average linkage tries to strike a balance. It uses average pairwise dissimilarity, so clusters tend to be relatively compact and relatively far apart

Shortcomings of average linkage

- Average linkage is not perfect, it has its own problems:

  - It is not clear what properties the resulting clusters have when we cut an average linkage tree at given height h. Single and complete linkage trees each had simple interpretations

  - Results of average linkage clustering can change with a monotone increasing transformation of dissimilarities $d_{ii'}$. i.e., if h is such that $h(x) \leq h(y)$ whenever $x \leq y$, and we used dissimilarities $h(d_{ii'})$ instead of $d_{ii'}$, then we could get different answers

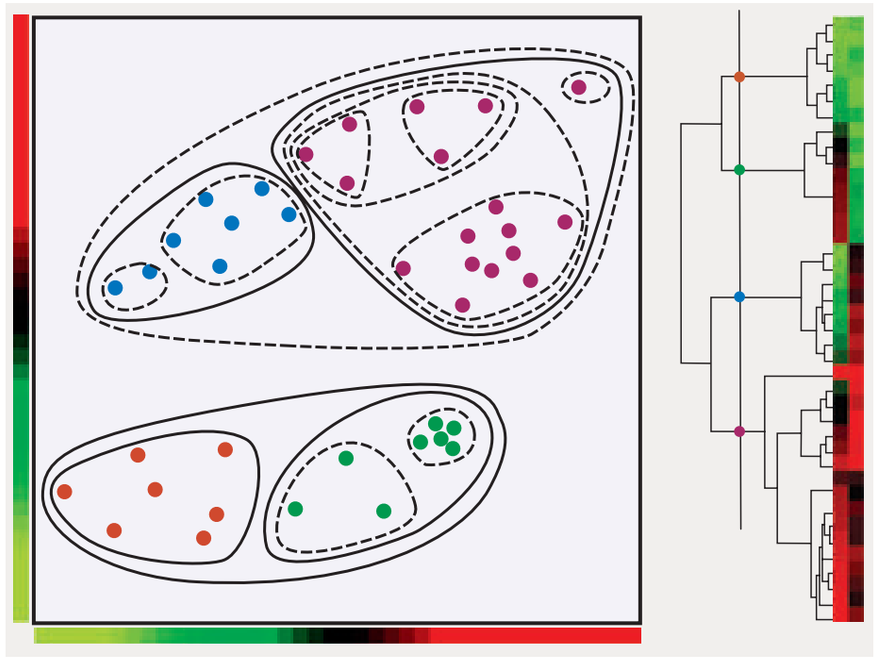
Summary: three linkages

- Three linkages: single, complete, average linkage. Properties:
  - Single and complete linkage can have problems with chaining and crowding, respectively, but average linkage doesn't
  - Cutting an average linkage tree provides no interpretation, but there is a nice interpretation for single, complete linkage trees
  - Average linkage is sensitive to a monotone transformation of the dissimilarities $d_{ij'}$, but single and complete linkage are not
  - All three linkages produce dendrograms with no inversions

Centroid linkages

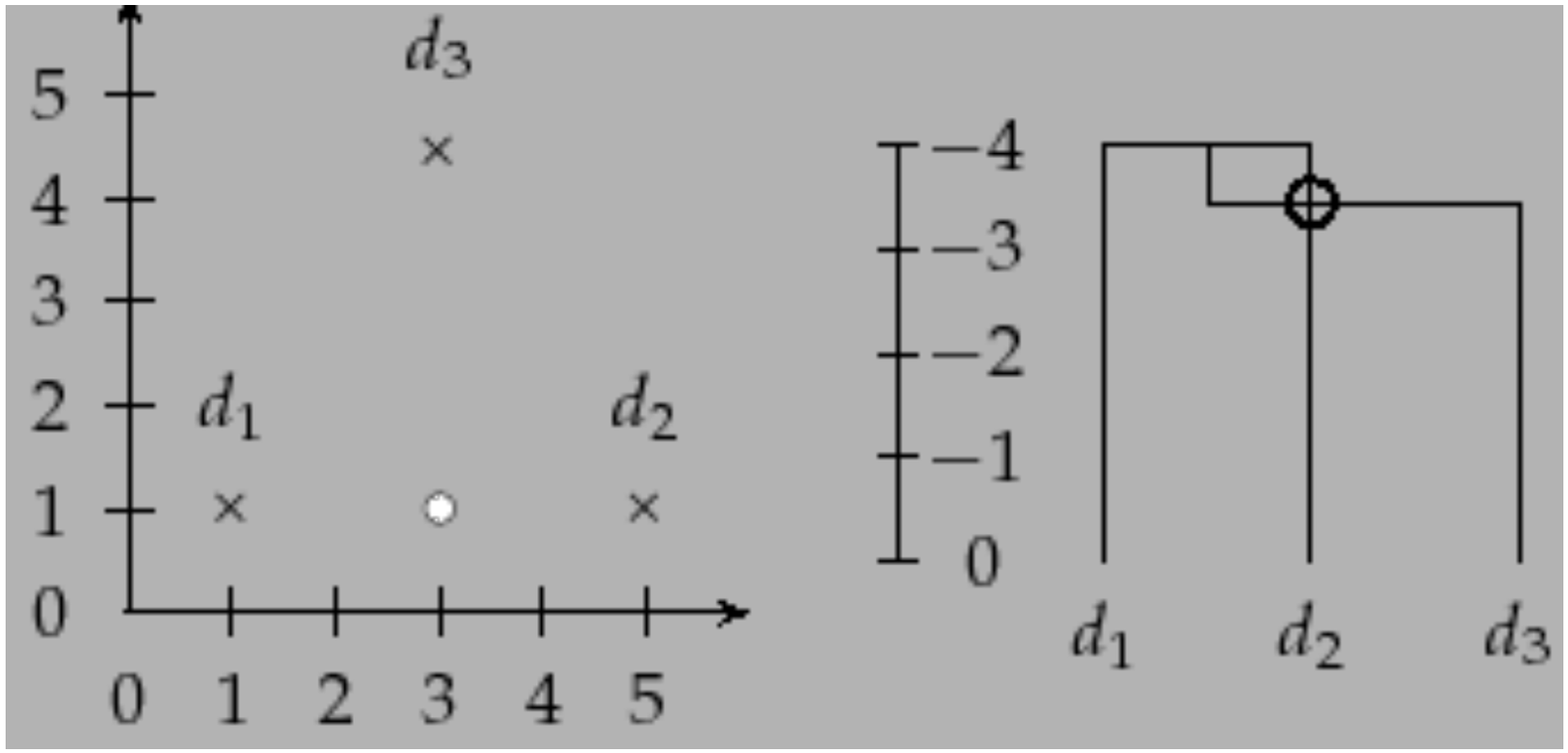
- In centroid linkage, the dissimilarity between G and H is calculated based the dissimilarity of two centroids:

$$d_{centroid}(G,H) = \| \bar{X}_G - \bar{X}_H \|_2$$



D'haeseleer P., 2005. *Nature*

- It is conceptually simple and easy to implement. The centroid linkage is commonly used for hierarchical clustering in biomedical studies.



- Example of inversions (problem of centroid linkage)
  - In the first merge, the similarity (negative distance) of d_1 and d_2 is -4.
  - In the second merge, the similarity of the centroid of d_1 and d_2 (the circle) and d_3 is > -4.
- Dissimilarities decreases (similarity increases) in this sequence of two clustering steps. This Violates the fundamental assumption that small clusters are more coherent than large clusters.
- The dotted circles in the dendrogram above indicate inversions: The horizontal merge bar is lower than the bar of a previous merge.

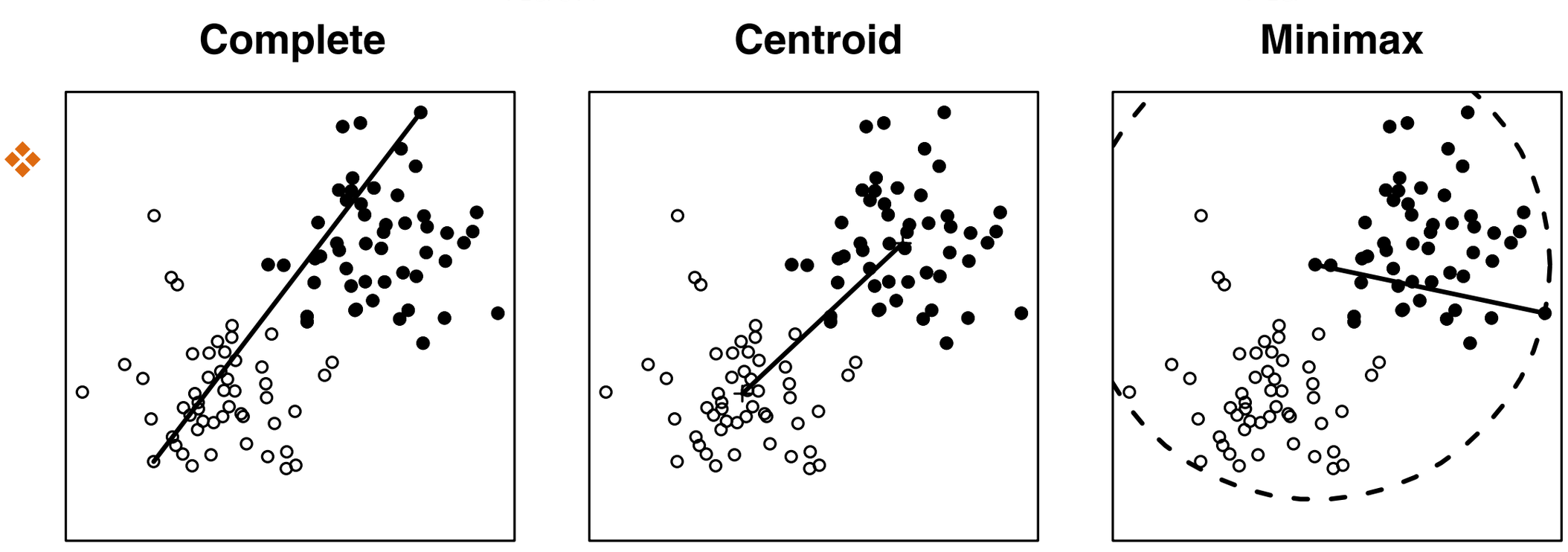
Shortcomings of centroid linkage

- Inversions can occur.
- No interpretation for the clusters resulting from cutting the tree
- Answers change with a monotone transformation of the dissimilarity measure
  - For example, changing $d_{ii'} = ||X_i - X_{i'}||$ to $d_{ii'} = ||X_i - X_{i'}||^2$ would give a different clustering

Minimax linkage

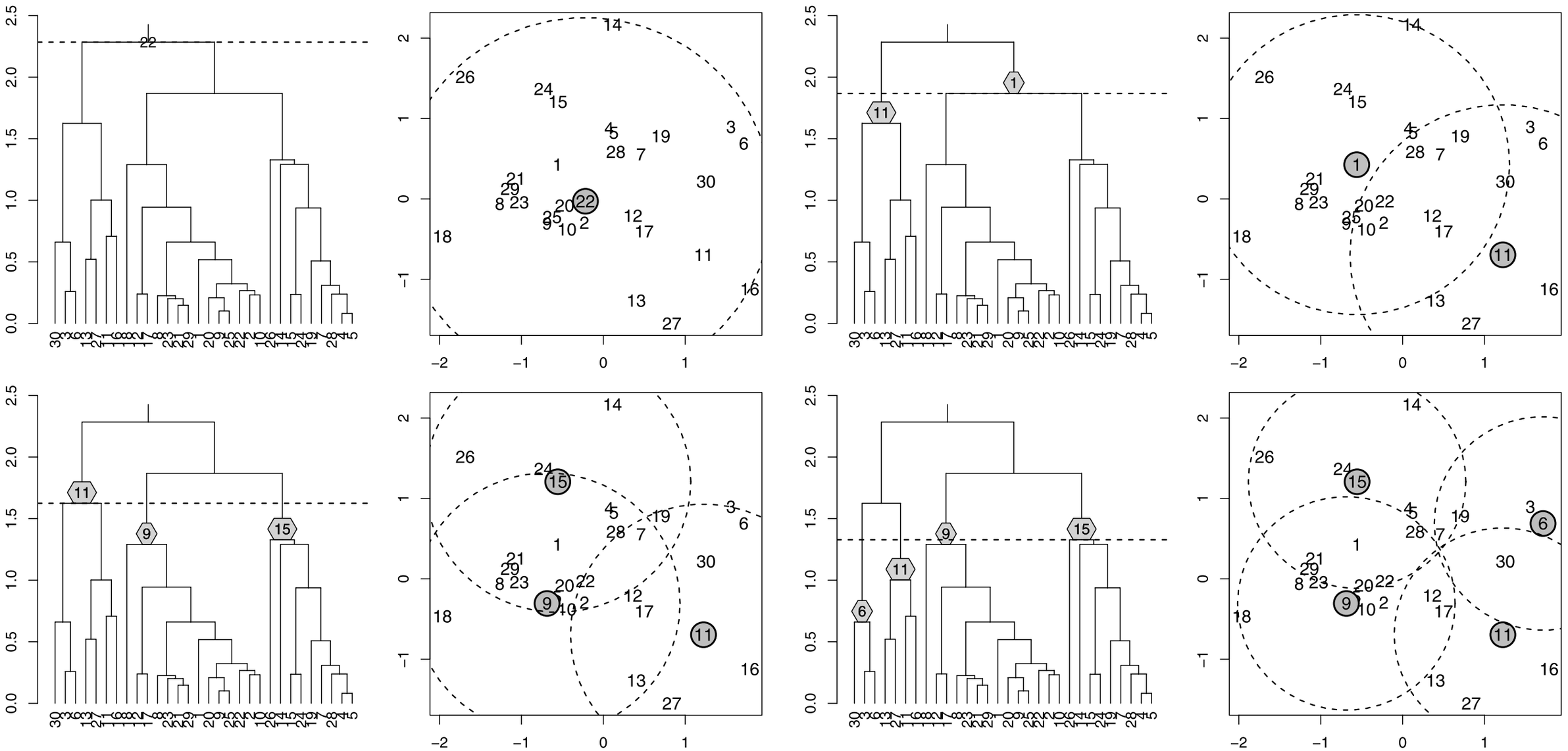
- In minimax linkage, the dissimilarity between G and H is calculated based the smallest radius encompassing all points in G and H.

$$d_{\text{mini}}(G,H) = \min_{i \in G \cup H} r(X_i, G \cup H) \text{ where } r(X_i, G) = \max_{i' \in G} d_{ii'}$$



-

Bien and Tibshirani 2011. *JASA*



- Bien and Tibshirani 2011. *JASA*: Successive cuts of a minimax tree with clusters.
- Cutting at height *h* yields a set of clusters such that every element of the dataset is covered by the set of balls of radius *h* centered at the cluster.
- Cut interpretation: each point X_i belongs to a cluster whose center X_c satisfies $d_{ic} \leq h$

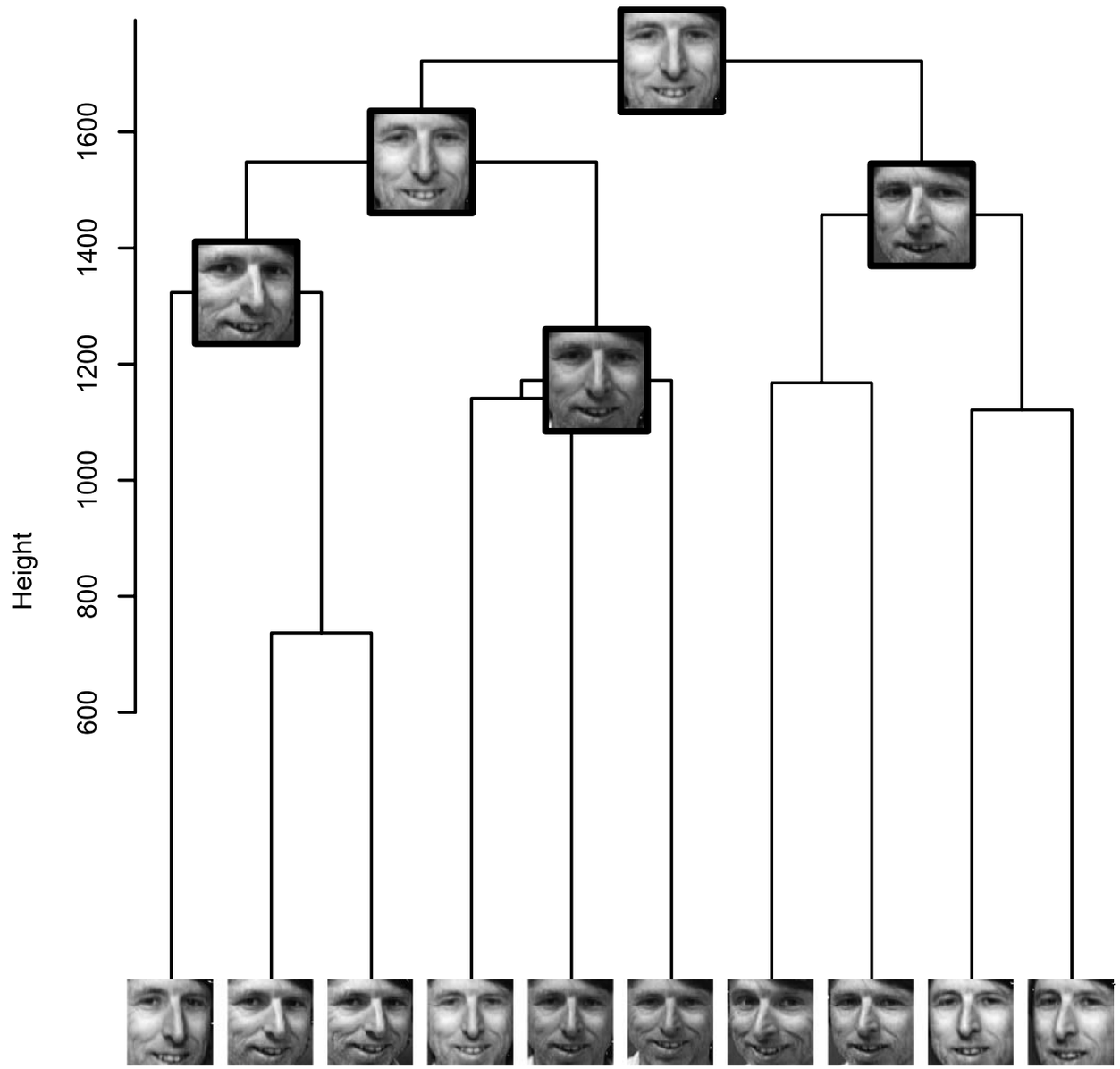
Remarks on minimax linkage

- Cutting a minimax tree at a height h a nice interpretation: each point is ≤ h in dissimilarity to the center of its cluster.

- Produces dendrograms with no inversions

- Unchanged by monotone transformation of dissimilarities

- Produces clusters whose centers are chosen among the data points themselves. Depending on the application, this can be an important property. (Hence minimax clustering is the analogy to K-medoids in the world of hierarchical clustering)



- Olivetti Faces Dataset: The dataset contains 10 images each of 40 distinct people. The minimax linkage hierarchical clustering is performed.

- The figure shows a branch of the minimax linkage dendrogram.

Clustering: part III

STAT 6312
Department of mathematics, UTA

ESL 14.3, 14.5

Optional reading: ESL

How to specify K?

- Sometimes, using K-means, K-medoids, or hierarchical clustering, we might have no problem specifying the number of clusters K ahead of time

  - Segmenting a client database into K clusters for K salesman

- Other times, K is implicitly defined by cutting a hierarchical clustering tree at a given height, e.g., placing cell phone towers

- But in most exploratory applications, the number of clusters K is unknown. So we are left asking the question: what is the right value of K?

It’s hard but important

- Determining the number of clusters is a hard task for humans to perform (unless the data are low-dimensional). Not only that, it’s just as hard to explain what it is we’re looking for.

- Why is it important?

  - It might mean a big difference scientifically if we were convinced that there were K = 2 subtypes of breast cancer vs. K = 3 subtypes

  - One of the (larger) goals of data mining/statistical learning is automatic inference; choosing K is certainly part of this

Back to K-means

- We're going to focus on K-means, but most ideas will carry over to other settings

- Recall: given the number of clusters K, the K-means algorithm approximately minimizes the within-cluster variation over clustering assignments C:

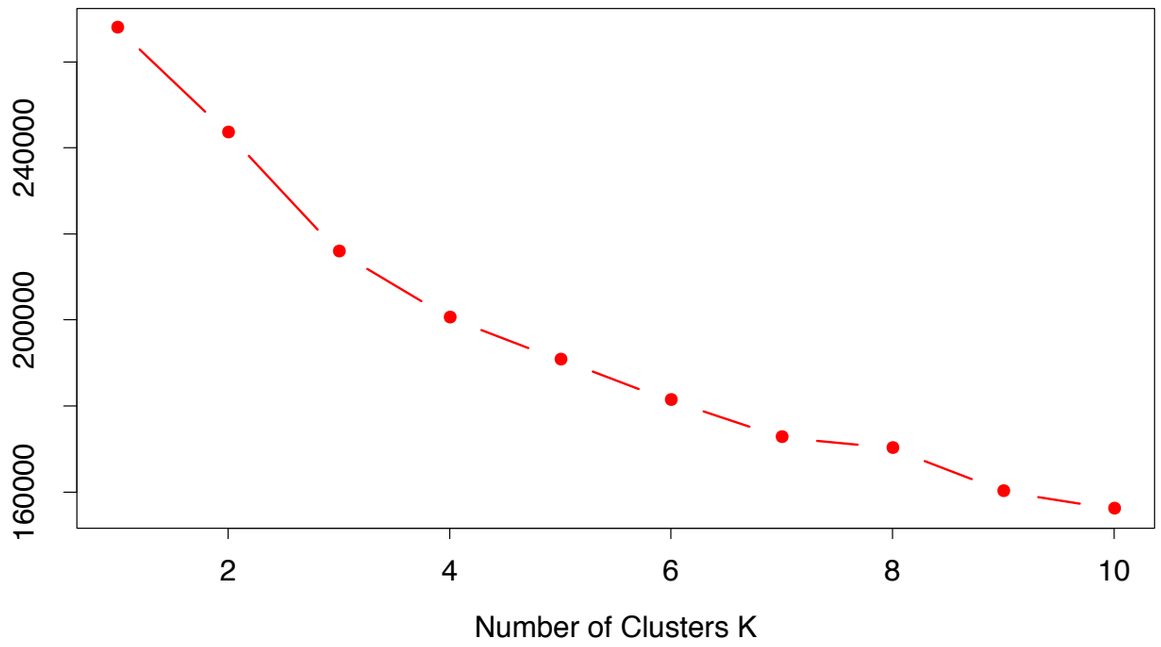
$$W = \sum_{k=1}^{K} N_k \sum_{C(i)=k} \| x_i - \bar{x}_k \|^2$$

- Clearly a lower value of W is better. So why not just run K-means for a bunch of different values of K, and choose the value of K that gives the smallest W(K)?

Not going to work…

- W(K) is a decreasing function of K. In fact, taking K=N (one observation in each cluster) yields W(K) =0.

- Thus, we need to balance W(K) and K by minimizing W(K) + λK. The question is how to choose λ.



ESL 14.8: Total within-cluster variation for K-means clustering applied to the human tumor microarray data.

Gap statistics

- The gap statistic compares the observed W(K)'s with what would be expected from a sample with no cluster structure.

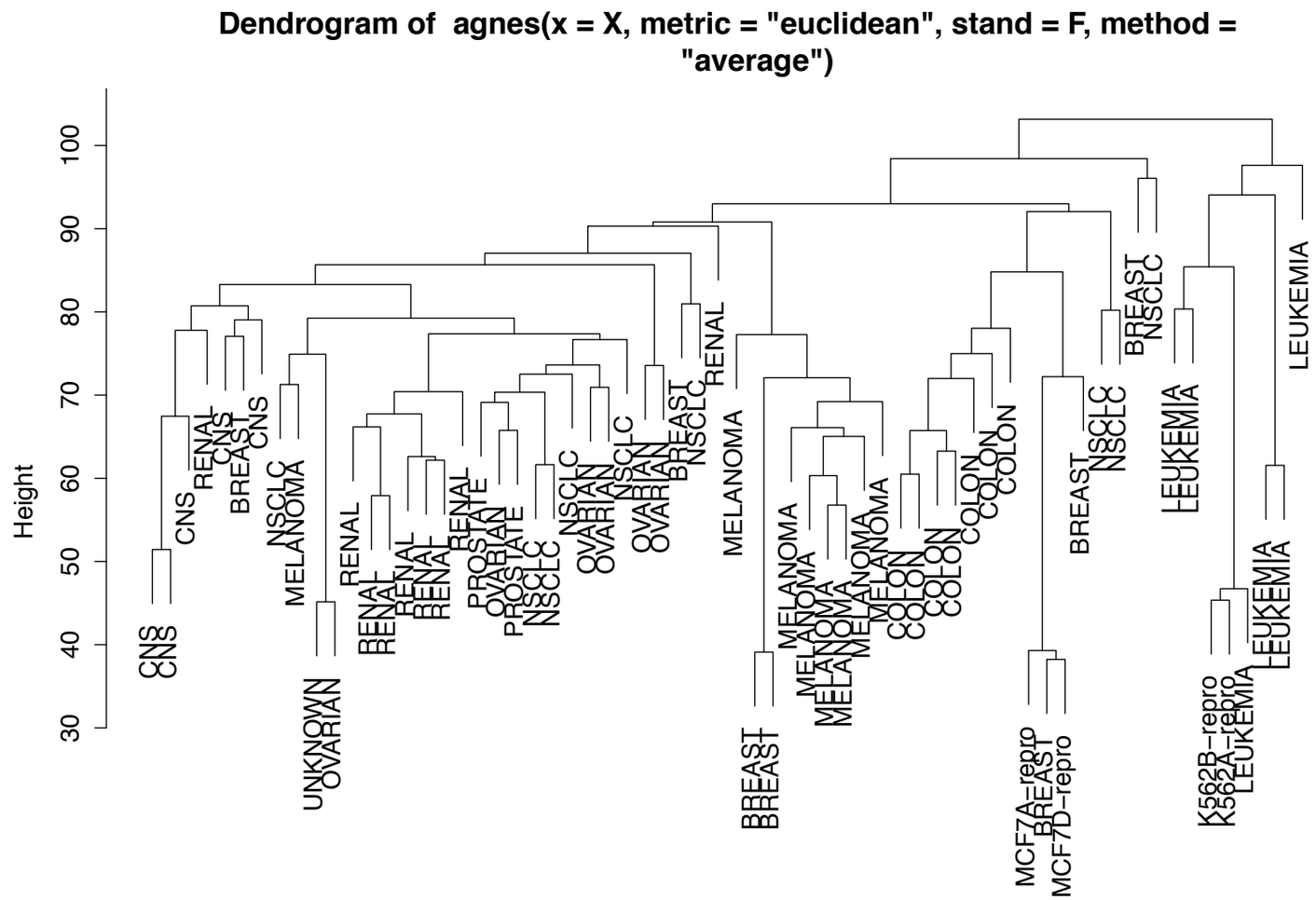
$$G(K) = E_0[\log W(K)] - \log W(K)$$

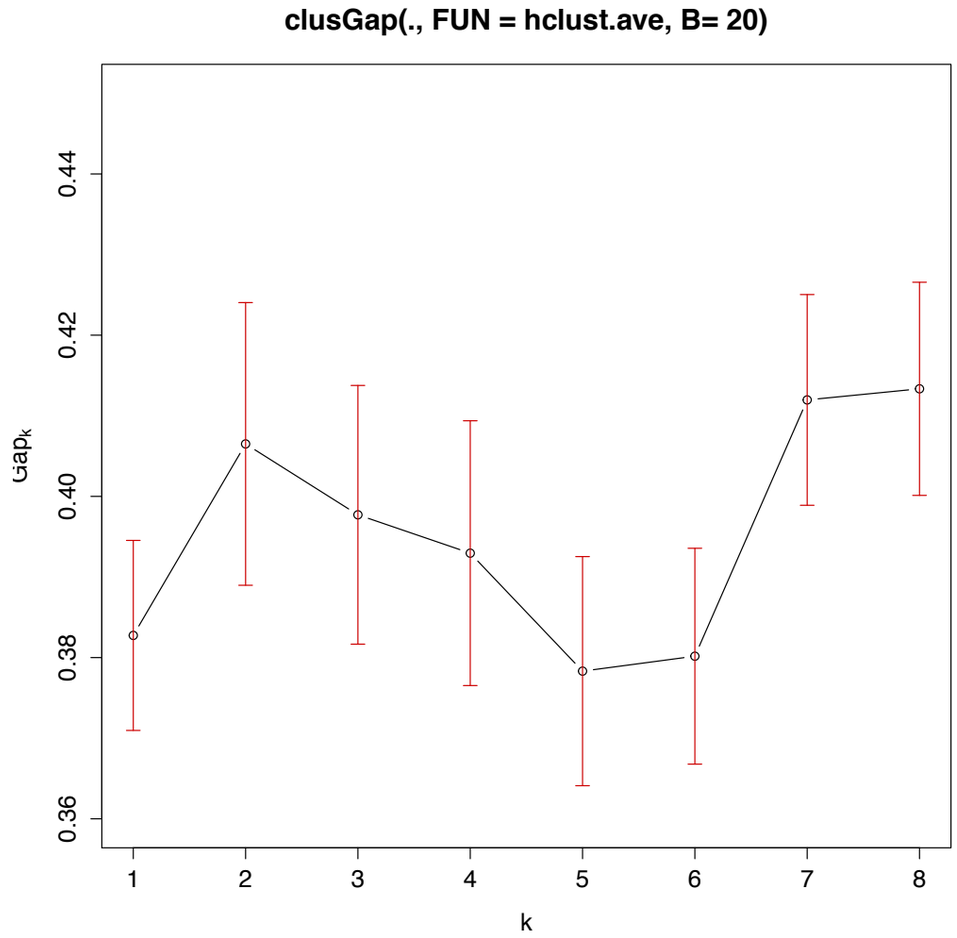
- Tibshirani suggests taking a uniform distribution over the range of the data. The quantity E_0[log W(K)] and its standard error s(K) are computed by simulation.

- A large value of G(K) indicates that the observed clustering is substantially better than what would be expected if there were no clusters. Thus we look for a K with a large gap G(K).

- $$\hat{K} = \min\{K \in \{1,\ldots,K_{\max}\} : G(K) \geq G(K+1) - s(K+1)\}$$

NCI microarray data





- Hierarchical clustering with centroid linkage is applied on the NCI microarray data.
- Which K should we choose based on the gap statistic?

Silhouettes

- Another measure of clustering efficacy is the silhouette.

- The silhouette of an observation i measures how well it fits in its own cluster versus how well it fits in its next closest cluster.

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

- Recall C(i)=k, and suppose group l has the next-closest group mean to x_i. Adapted to K-means, we have

$$a(i) = \| x_i - \bar{x}_k \|^2 \text{ and } b(i) = \| x_i - \bar{x}_l \|^2$$

- Q. What are minimum and maximum values of the silhouette?

High silhouettes are good!

- If the observation is equal to its group mean, its silhouette is 1. If it is halfway between the two group means, its silhouette is 0.

- For other clusterings (e.g. K-medoids), the silhouettes can range from -1 to 1, but usually stay above 0, or at least do not go much below.

- High silhouettes are good, so that the average silhouette is a measure of goodness for the clustering.

Silhouette plot

- K-means is applied with K=2,…,7 on NCI microarray data, and Silhouettes are computed.

- The silhouette plots are resented in the next slides. The observations (along the vertical axis) are arranged by group and, within group, by silhouettes.

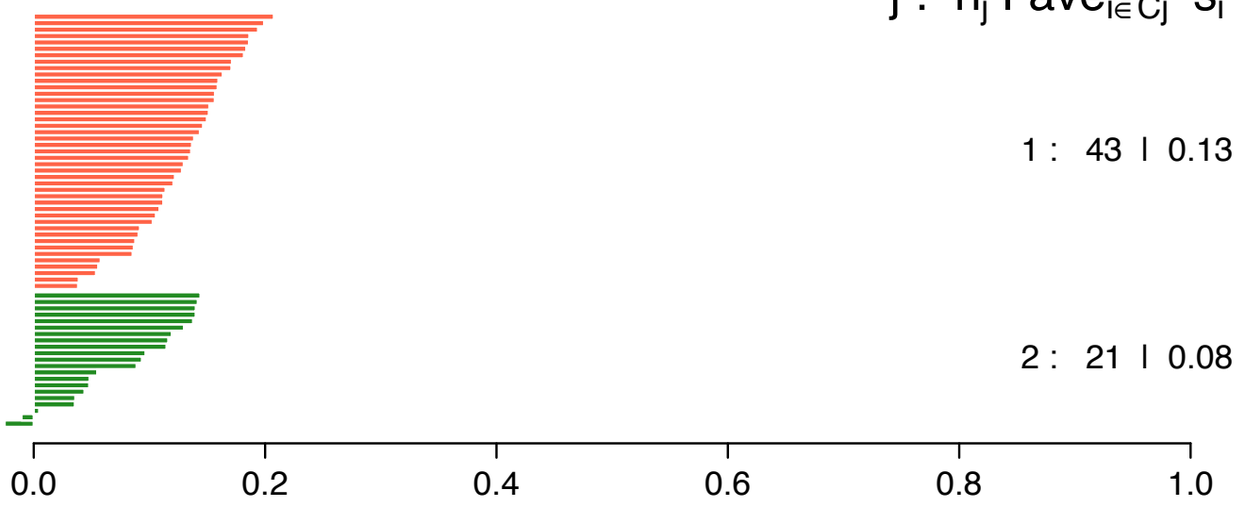
- This arrangement allows one to compare the clusters. Here we can see that clusters have different silhouettes.

k = 2

j : n_j | $ave_{i \in Cj}$ s_i

1 : 43 | 0.13

2 : 21 | 0.08



Average silhouette width : 0.11 Silhouette width s_i

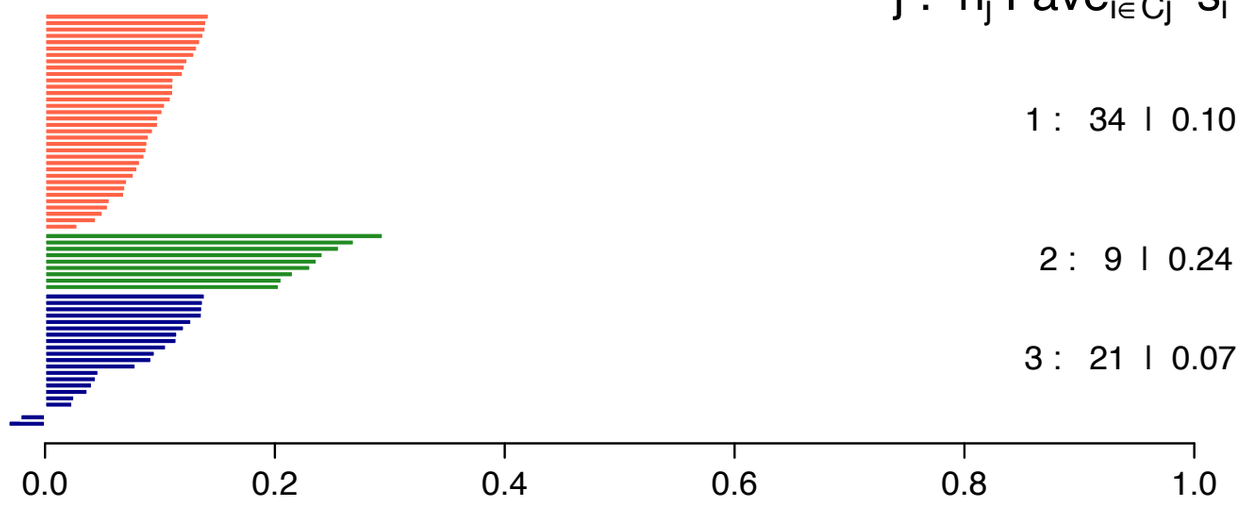
k = 3

j : n_j | $ave_{i \in Cj}$ s_i

1 : 34 | 0.10

2 : 9 | 0.24

3 : 21 | 0.07



Average silhouette width : 0.11 Silhouette width s_i

k = 4

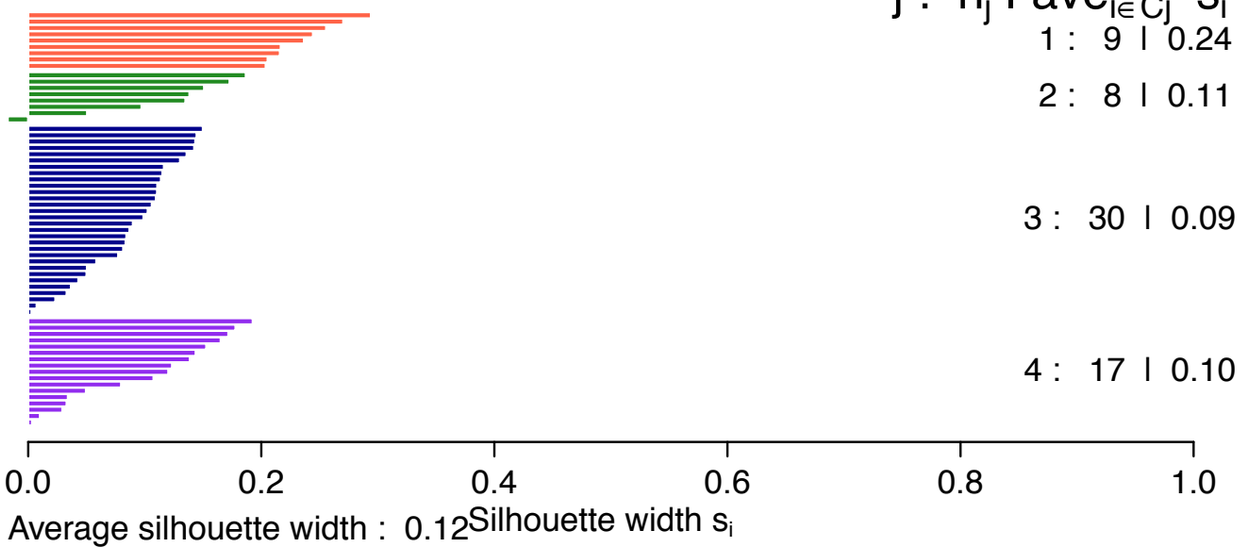
j : n_j | $ave_{i \in Cj}$ s_i

1 : 9 | 0.24

2 : 8 | 0.11

3 : 30 | 0.09

4 : 17 | 0.10



Average silhouette width : 0.12 Silhouette width s_i

k = 5

j : n_j | $ave_{i \in Cj}$ s_i

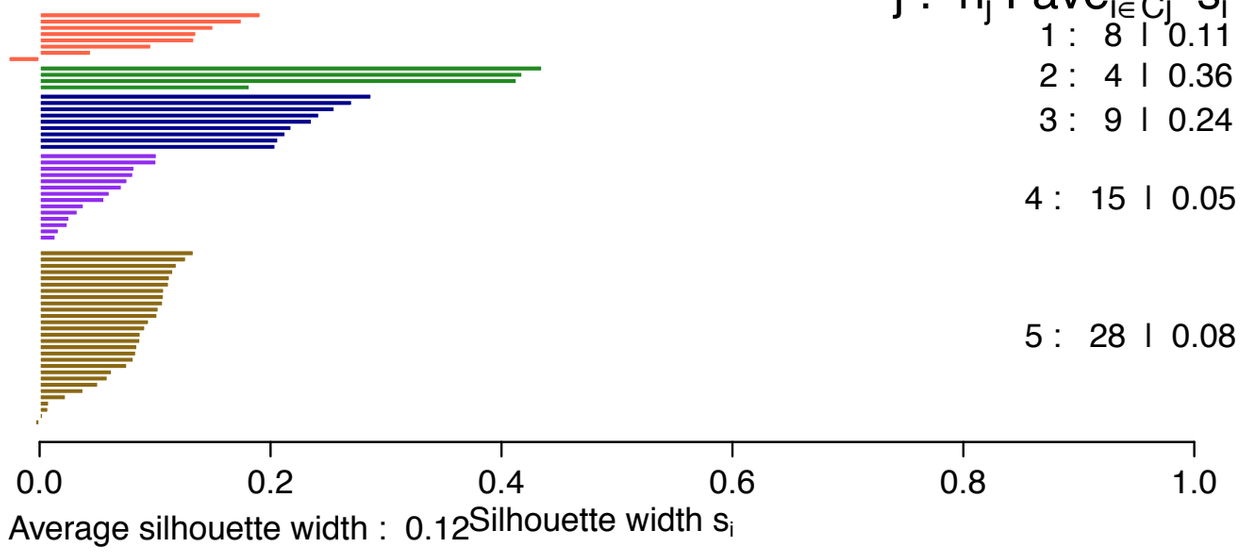
1 : 8 | 0.11

2 : 4 | 0.36

3 : 9 | 0.24

4 : 15 | 0.05

5 : 28 | 0.08



Average silhouette width : 0.12 Silhouette width s_i

k = 6

j : n_j | $ave_{i \in Cj}$ s_i

1 : 9 | 0.24

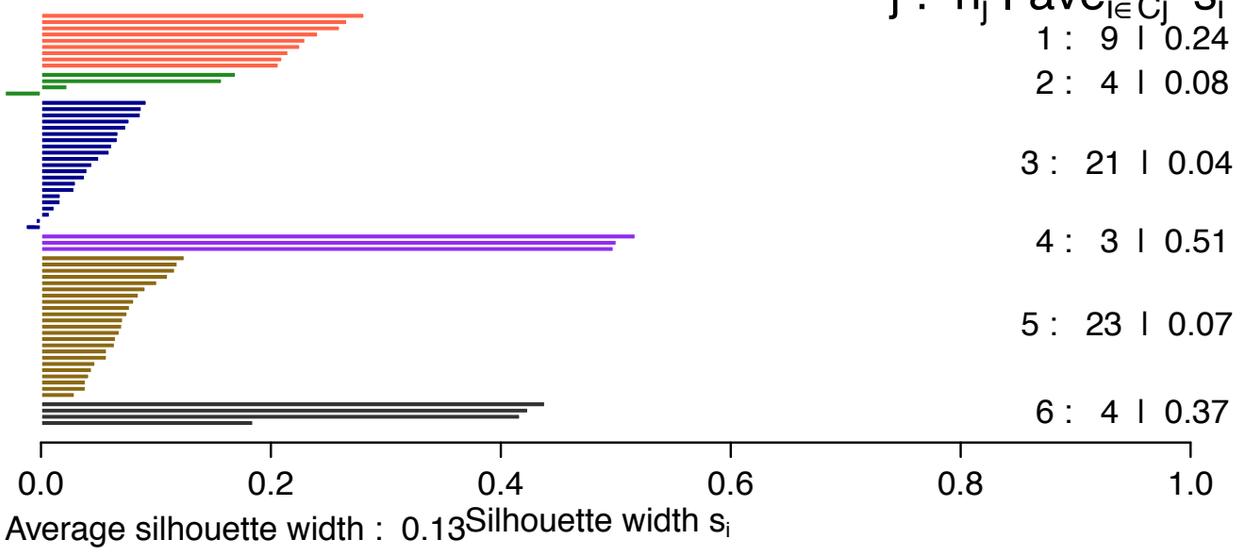
2 : 4 | 0.08

3 : 21 | 0.04

4 : 3 | 0.51

5 : 23 | 0.07

6 : 4 | 0.37



Average silhouette width : 0.13 Silhouette width s_i

k = 7

j : n_j | $ave_{i \in Cj}$ s_i

1 : 9 | 0.22

2 : 23 | 0.03

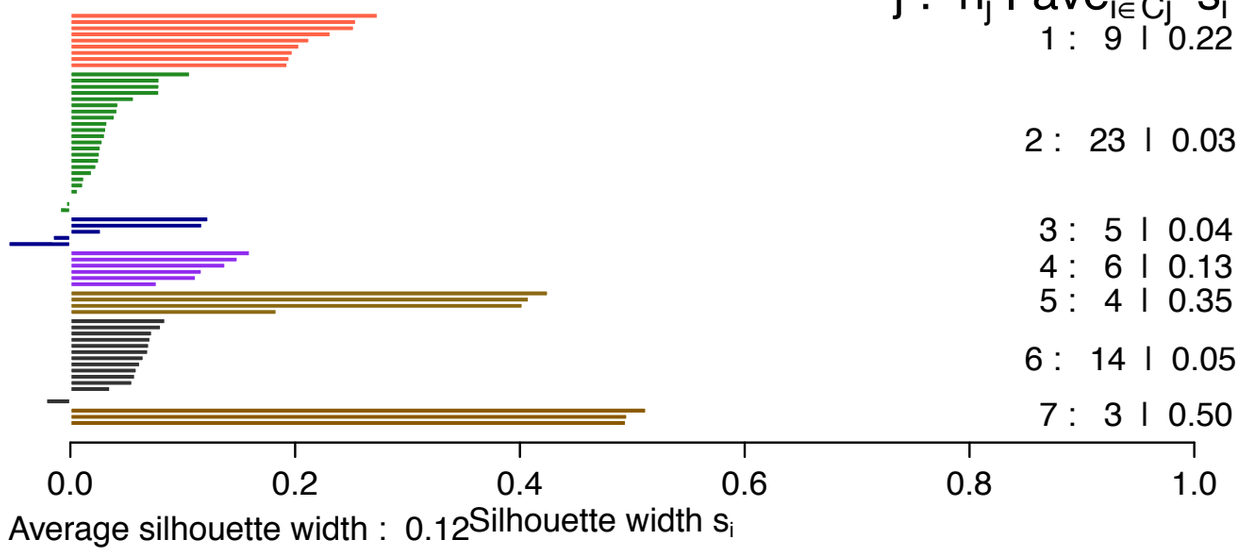
3 : 5 | 0.04

4 : 6 | 0.13

5 : 4 | 0.35

6 : 14 | 0.05

7 : 3 | 0.50



Average silhouette width : 0.12 Silhouette width s_i

Within/between-cluster variation

- Within-cluster variation W(K) measures how tightly clusters are grouped. As we increase the number of clusters K, this just keeps going down.
- Between-cluster variation B(K) measures how spread apart the groups are from each other.
- Again, adapted to K-means, we have

$$W = \sum_{k=1}^{K} N_k \sum_{C(i)=k} \| x_i - \bar{x}_k \|^2 \qquad B = \sum_{k=1}^{K} N_k \| \bar{x}_k - \bar{x} \|^2$$

- Q. Is B(K) an increasing function of K?

CH index

- Ideally we'd like our clustering assignments C to simultaneously have a small W and a large B
- This is the idea behind the CH index. For clustering assignments coming from K clusters, we record CH score:

$$CH(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)}$$

- To choose K, just pick some maximum number of clusters to be considered K_{max}, and choose the value of K with the largest score CH(K).
- $$\hat{K} = \underset{K=2,\ldots,K_{max}}{\operatorname{argmax}} CH(K)$$

Take home messages

- Centroid linkage is commonly used in biology. It measures the distance between group averages, and is simple to understand and to implement. But it also has some drawbacks (inversions!)

- Minimax linkage is a little more complex. It asks the question: “which point’s furthest point is closest?”, and defines the answer as the cluster center.

- Determining the number of clusters is both a hard and important problem. Three methods for choosing K:

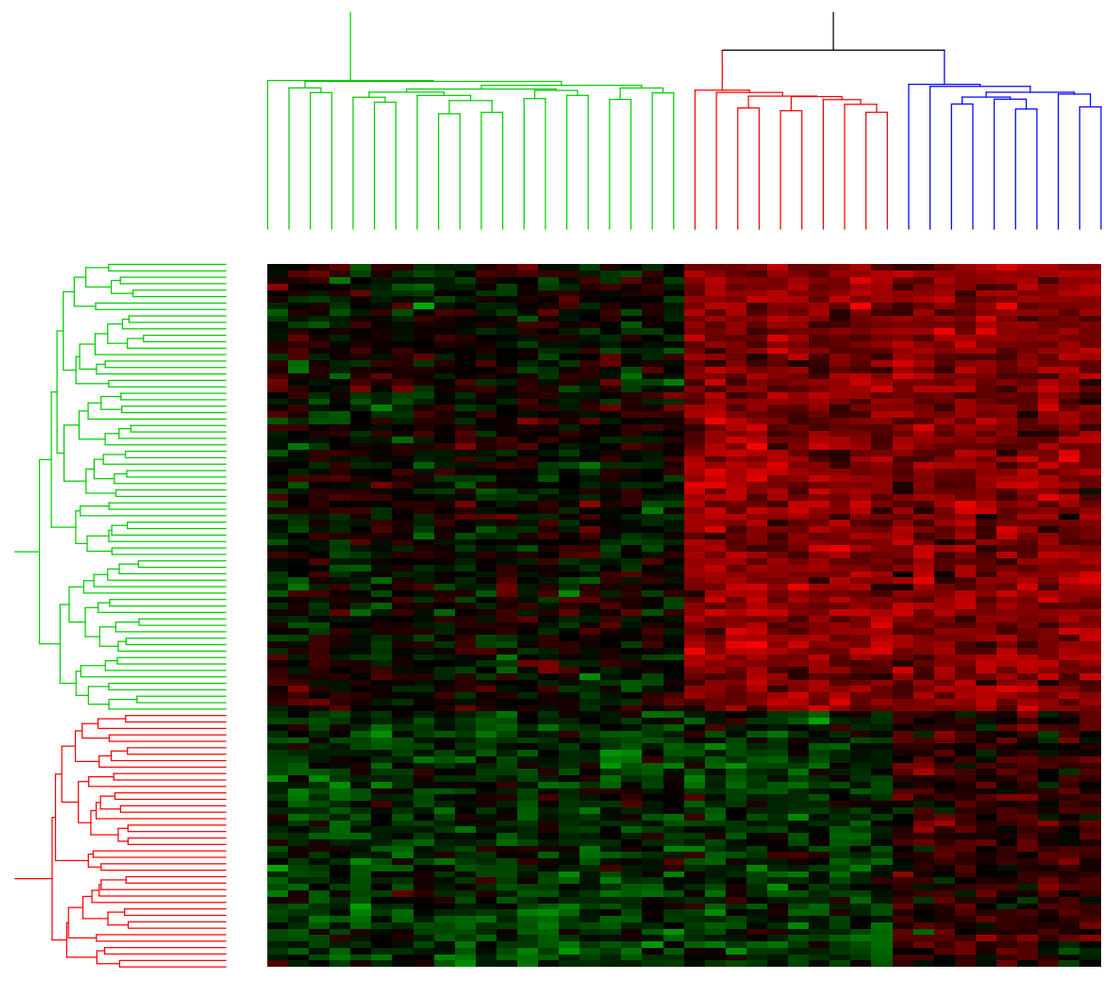
  - CH index looks at a ratio of between to within;

  - Gap statistic is based on the difference between within-class variation for our data and what we expect to see from the data with no clusters;

  - Silhouette measures how well a data point fits in its own cluster versus next closest cluster

Clustering as dimension reduction

- Artificial gene expression data
  - row: sample (observation)
  - column: gene (feature)



- In many situations, we can actually cluster the observations or the features or both.
- If we cluster the features, then we could replace the features by cluster centers or cluster assignments. This would reduce the dimension of our feature space.

Dimension reduction

- Dimension reduction: the task of transforming our data set to one with less features. A new feature can be one of the old features, or it can be a some linear or nonlinear combination of old features. We want this transformation to preserve the main structure that is present in the feature space

- This is a broader goal than that of clustering. It is often the first step in an analysis, to be followed by, e.g., visualization, clustering, regression, classification

- We're going to start with linear dimension reduction. This means: looking for straight lines in the feature space along which the data exhibit an interesting trend

- Specifically, we're going to interpret interesting to mean high variance

Revisit: singular value decomposition (SVD)

- Let $X \in R^{Nxp}$ be a matrix of rank r (r<p). Then, there exists orthogonal matrices $U=(U_1,U_0) \in R^{NxN}$, $V=(V_1,V_0) \in R^{pxp}$, and a diagonal matrix $D_1=diag(d_1,d_2,\ldots,d_r) \in R^{rxr}$ such that

$$X = UDV' = U\begin{pmatrix} D_1 & 0 \\ 0 & 0 \end{pmatrix}V' = U_1 D_1 V_1'$$

- $d_1 \geq d_2 \geq \ldots \geq d_r \geq 0$, $U_1 \in R^{Nxr}, U_0 \in R^{Nx(p-r)}$ and $V_1 \in R^{pxr}, V_0 \in R^{px(p-r)}$

- $XX' = UD^2U'$ and $X'X = VD^2V'$.

- We can have a low-dimensional approximation of X by

- $X \approx \sum_{i=1}^{k} d_i \mathbf{u}_i v_i^T = \mathbf{U}_{1:k}\mathbf{D}_{1:k}\mathbf{V}_{1:k}^T$ where $\mathbf{u}_i$ and v_i are i*th* column of U and V.

Orignal X

k=5

k=10

k=50

- Suppose columns of X is column-centered (column means are zeros).

- It is easy to see that

$$v_1 = \underset{\|v\|=1}{\operatorname{argmax}} (\mathbf{X}v)^T(\mathbf{X}v) \qquad\qquad v_1 = \underset{\|v\|=1}{\operatorname{argmax}} \widehat{Var}(\mathbf{X}v)$$

$$v_2 = \underset{\|v\|=1, v\perp v_1}{\operatorname{argmax}} (\mathbf{X}v)^T(\mathbf{X}v) \qquad\qquad v_2 = \underset{\|v\|=1, v\perp v_1}{\operatorname{argmax}} \widehat{Var}(\mathbf{X}v)$$

$$\vdots \qquad\qquad \vdots$$

$$v_p = \underset{\|v\|=1, v\perp (v_1, v_2, \ldots, v_{p-1})}{\operatorname{argmax}} (\mathbf{X}v)^T(\mathbf{X}v) \qquad\qquad v_p = \underset{\|v\|=1, v\perp (v_1, v_2, \ldots, v_{p-1})}{\operatorname{argmax}} \widehat{Var}(\mathbf{X}v)$$

- Note that X′X is a sample covariance matrix S, a sample variance of Xv is (Xv)′(Xv)/N= v′Sv.

- $(Xv_k)'(Xv_k)/N = v_k'V'D^2Vv_k/N = d_k^2/N$ is the amount of variance explained by v_k

Principal component analysis (PCA)

- Columns of V ($v_1,\ldots,v_p \in R^p$) are the principal component directions (PC loadings).

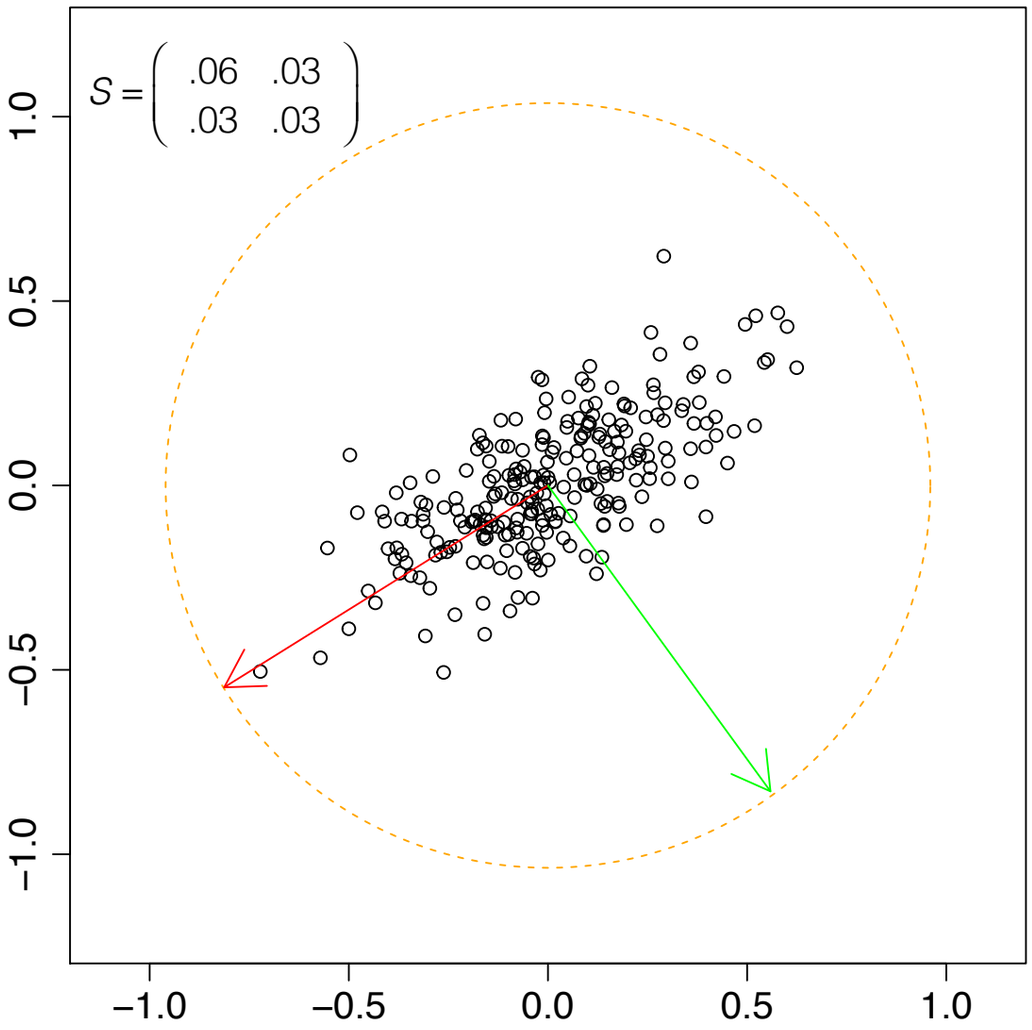
- Columns of U ($u_1,\ldots,u_p \in R^n$) are the normalized principal component (PC) scores.

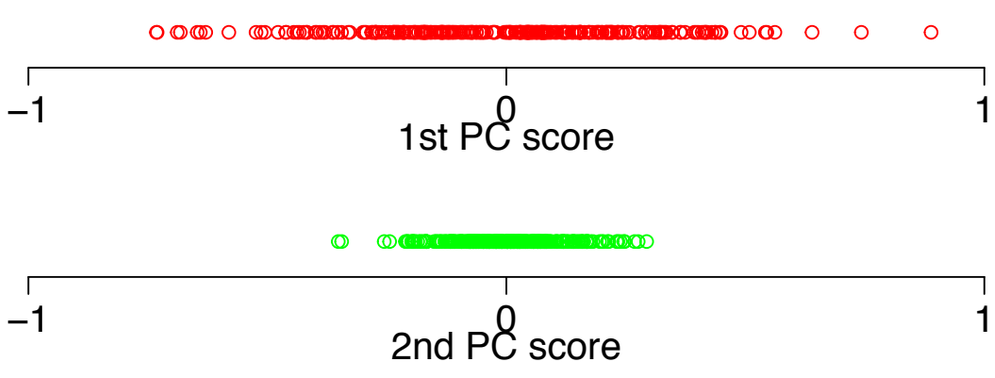
- Columns of XV=DU ($d_1u_1,\ldots,d_pu_p \in R^n$) are the principal component (PC) scores.

- The entries of $Xv_k=d_ku_k$ are the PC scores from projecting X onto v_k and the rows of $Xv_kv_k'=d_ku_kv_k'$ are the projected vectors .

- $u_k \perp u_l$ for $k \neq l$, so PC scores are uncorrelated.

X

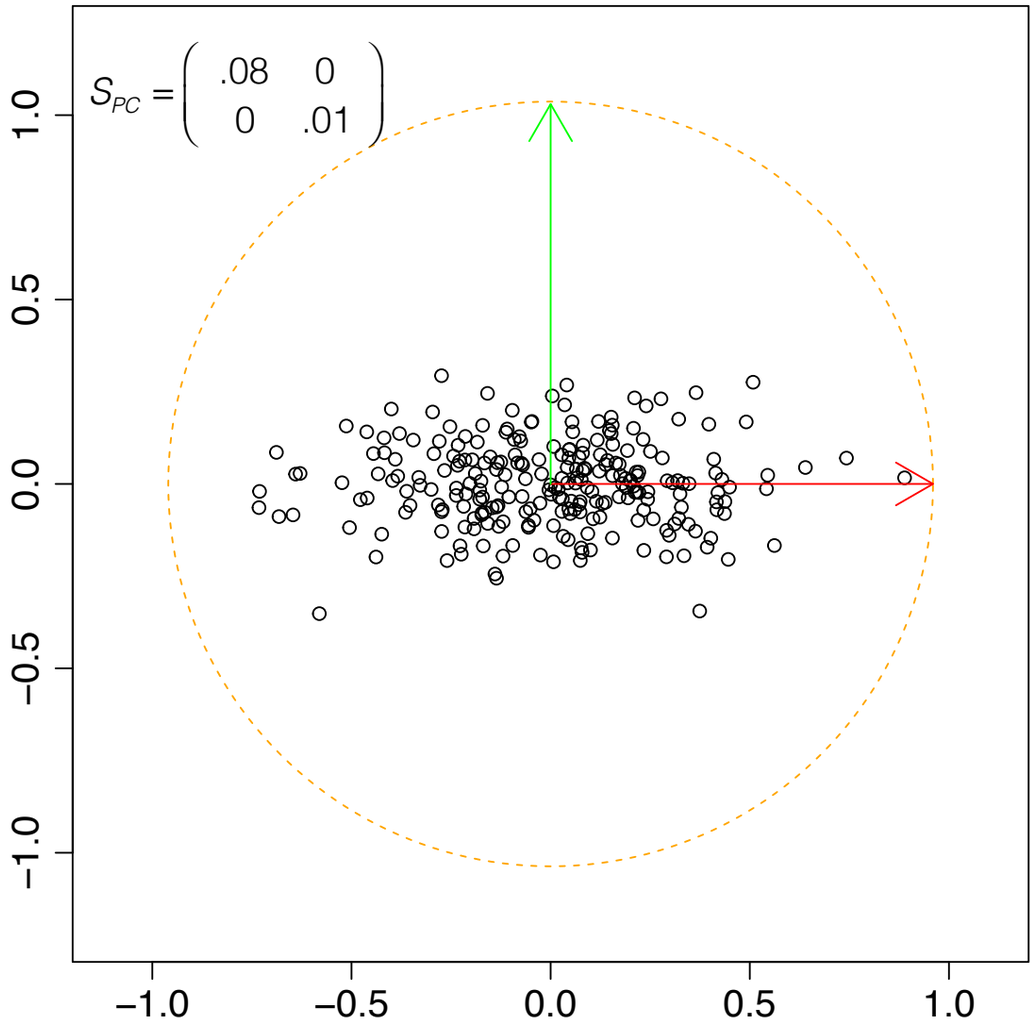




d_1u_1

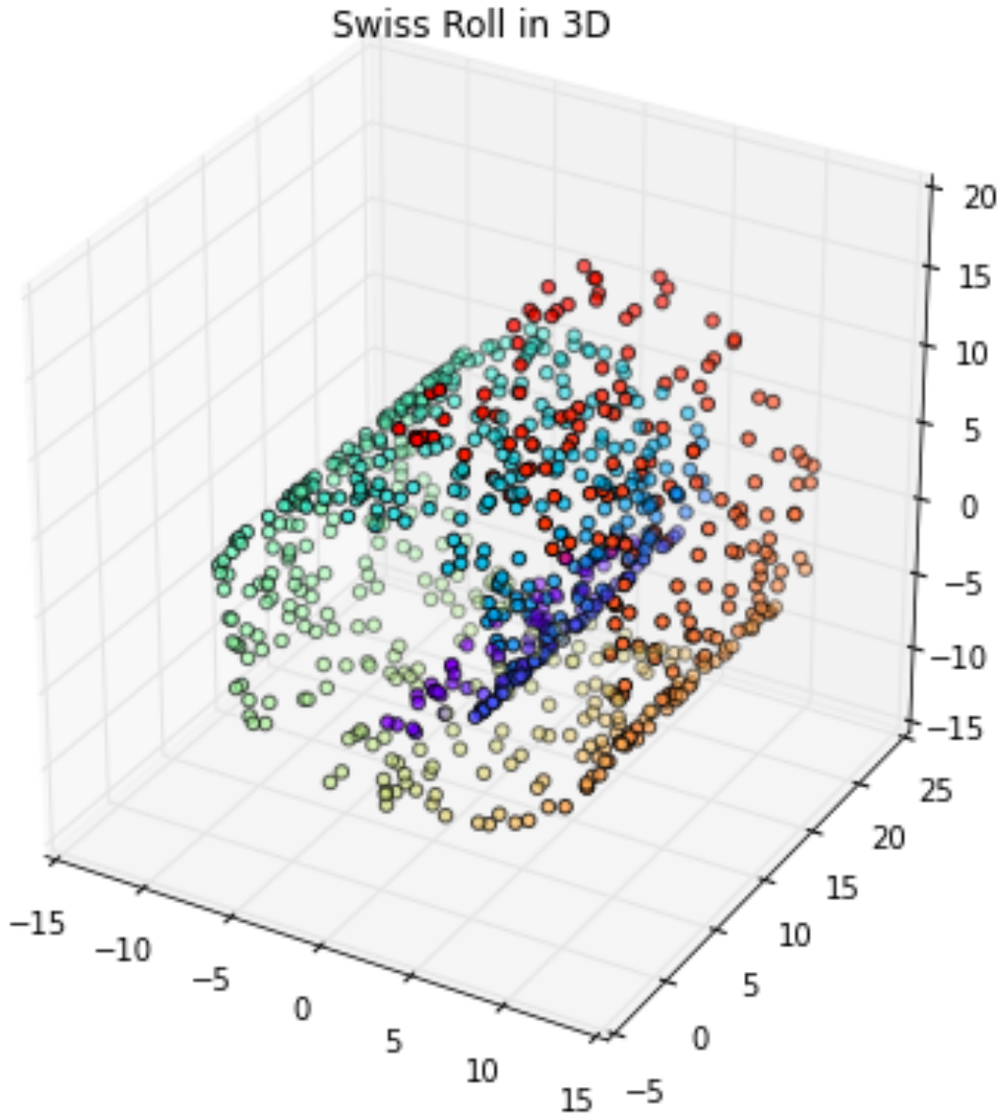
d_2u_2

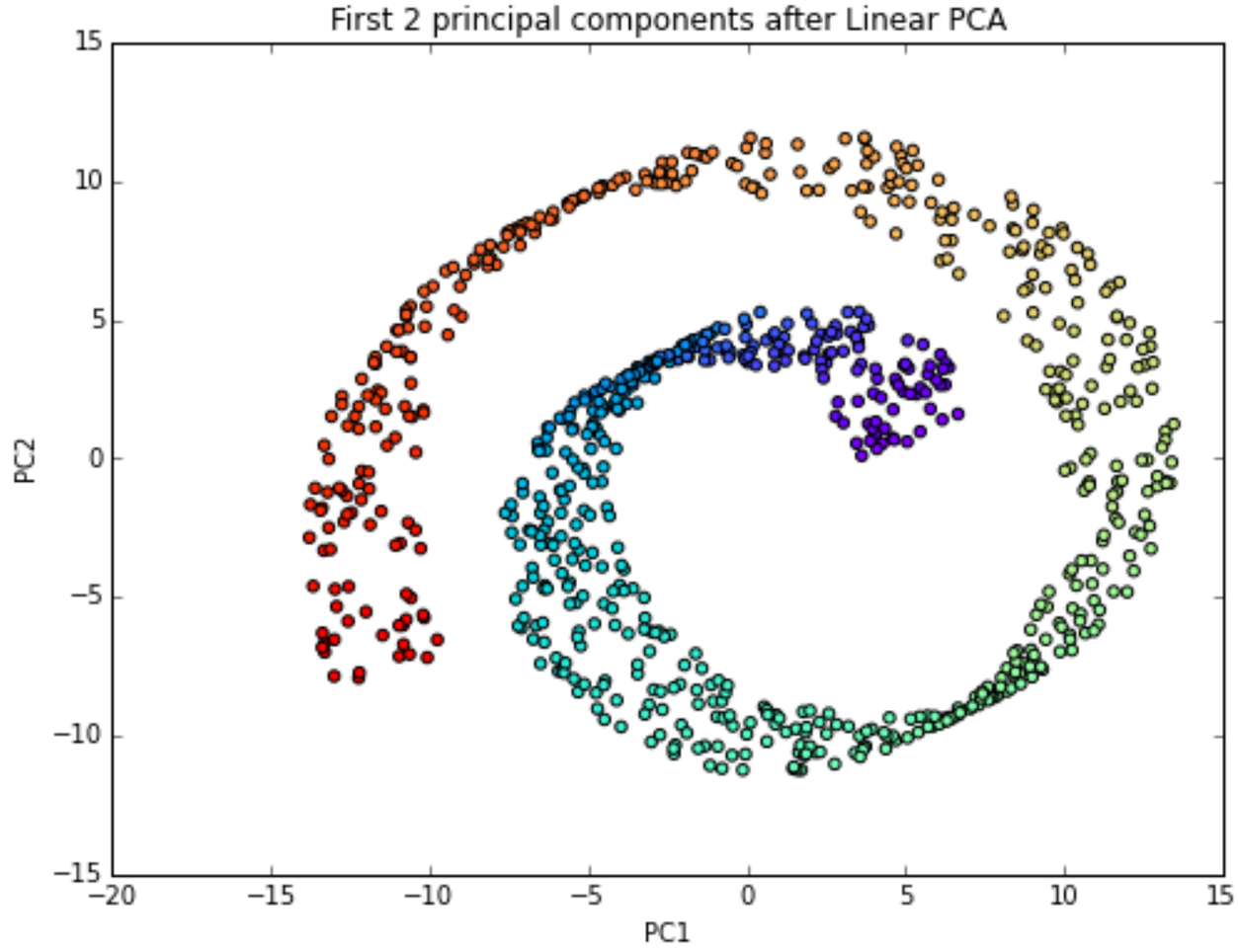
XV=UD



- V is an rotation operator which makes columns of XV (PCs) uncorrelated.
- $d_1^2/N = 0.08$, $d_2^2/N = 0.01$.

PCA in 3D (three features)





Credit: http://sebastianraschka.com/Articles/2014_kernel_pca.html

Approximation by PCA

- Think about approximating X by the projection of X onto the first k principal component directions

$$X \approx \mathbf{U}_{1:k}\mathbf{D}_{1:k}\mathbf{V}_{1:k}^T = \mathbf{X}\mathbf{V}_{1:k}\mathbf{V}_{1:k}^T$$

- An important alternate characterization of the principal component directions:

$$\mathbf{X}\mathbf{V}_{1:k}\mathbf{V}_{1:k}^T = \underset{rank(A)=k}{\operatorname{argmin}} \| X - A \|_F^2$$

- In other words, $XV1_{1:k}V_{1:k}^T$ is the best rank k approximation to X.

Orignal X

$XV1_{1:k}V_{1:k}^T$, k=10

How to choose k?

- In principal component regression, we used the cross validation to compare MSEs for different values of k.

- This is unsupervised learning. In general, we do not have a teacher who tells us about performance.

- The proportion of variance explained by the first k principal component directions $v_1,\ldots,v_k$ is

$$\frac{\sum_{j=1}^{k} d_j^2}{\sum_{j=1}^{p} d_j^2}$$

N·(total variance of $XV1_{1:k}V_{1:k}^T$) / N·(total variance of X)

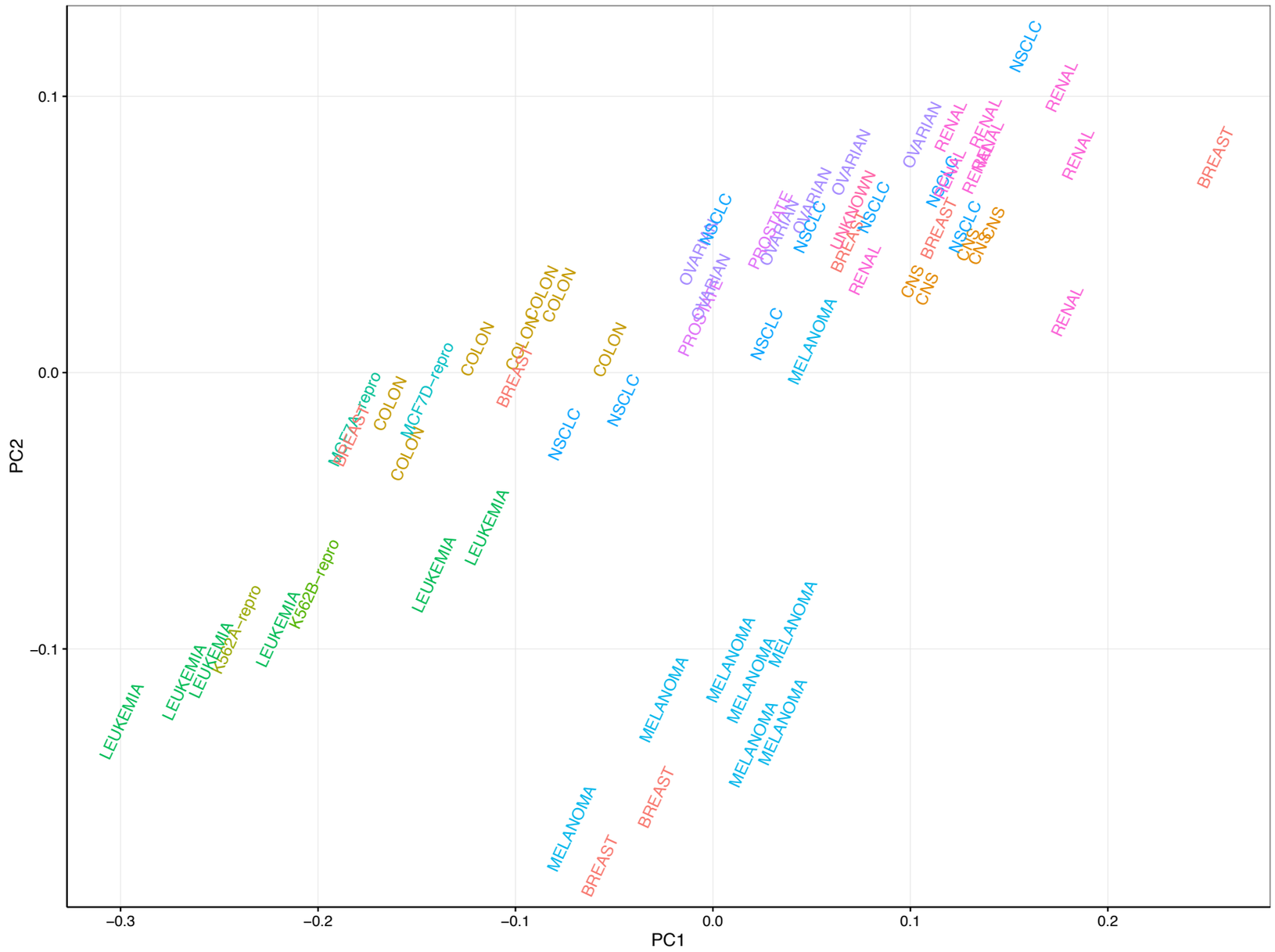
- If this is high for a small value of k, then it means that the main structure in X can be explained by a small number of directions

Scaling the features

- The results of PCA depend on the scales at which the variables are measured. Variables with the highest variances will tend to be emphasized in the first few principal components.

- If the variables either have different units of measurement (i.e., pounds, feet, gallons, etc), or if we wish each variable to receive equal weight in the analysis, then the variables should be standardized (to have unit variances) before a principal components analysis is carried out.

- But sometimes scaling is not appropriate (e.g., when you know the variables are all on the same scale to begin with)



- NCI microarray data (6830 genes and 64 samples).
- The first two PCs are used for the visualization.
- This technique is closely related to multidimensional scaling (non-linear dimensional reduction).

Spectral clustering

- Traditional clustering methods (e.g. K-means) use a spherical or elliptical metric to group data points, and they may not work well when clusters are non-convex.

- A convex set is the region such that, for every pair of points within the region, the straight line that connects the pair is also within the region.

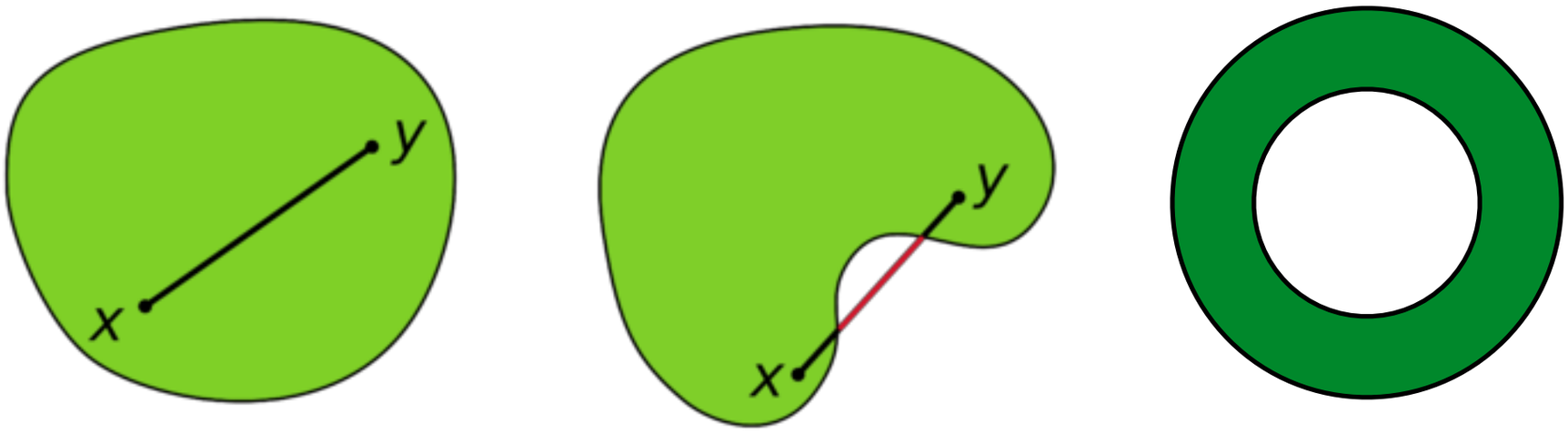
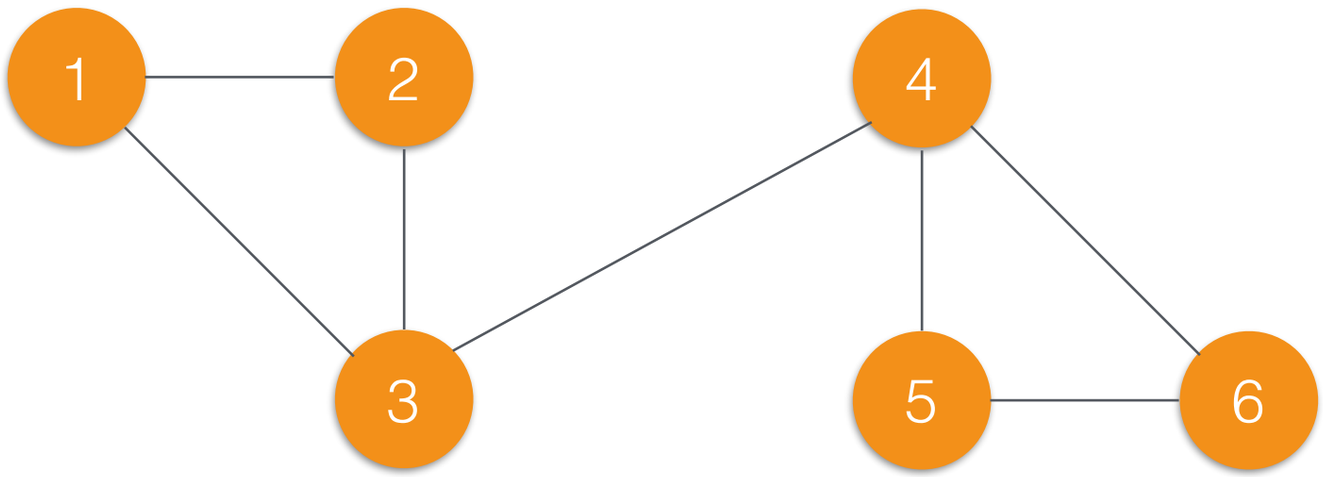


Image credit: https://en.wikipedia.org/wiki/Convex_set

Similarity graph

- A (undirected) similarity graph G = <V,E,W> consists of a vertex set V, an edge set E, and edge weights W (similarities).
- N vertices represent observations and paris of vertices are connected by an edge if they are similar.



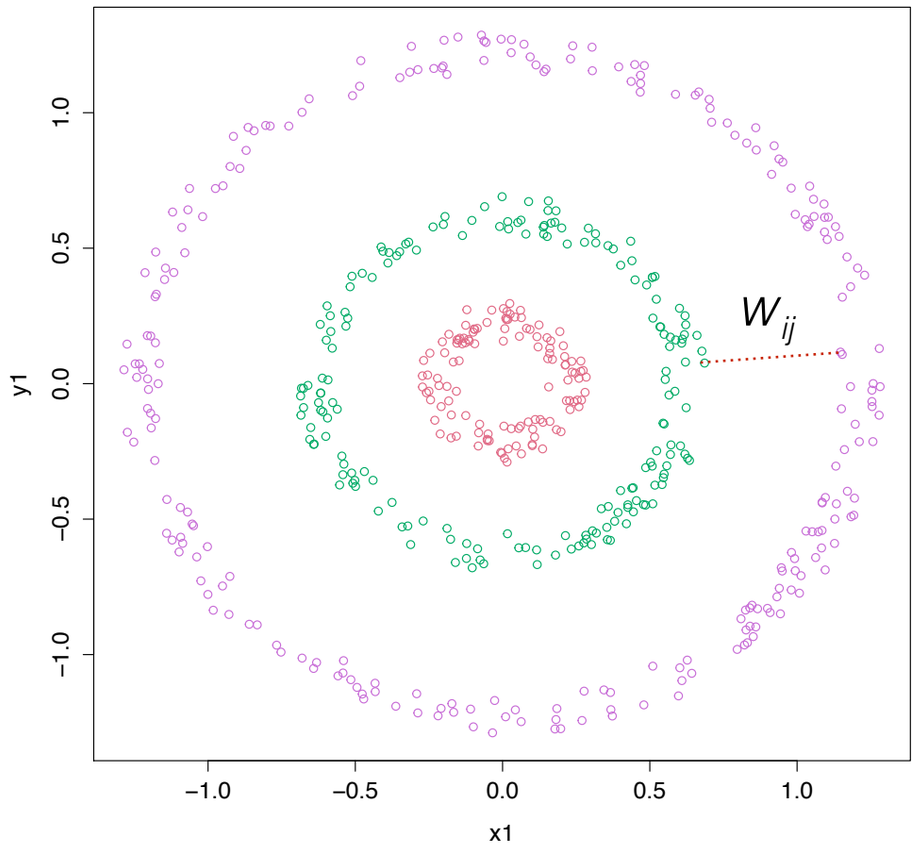
- w_{ij}>0 if i and j are connected; 0 otherwise

0	0.8	0.6	0	0	0
0.8	0	0.8	0	0	0
0.6	0.8	0	0.2	0	0
0	0	0.2	0	0.7	0.8
0	0	0	0.7	0	0.7
0	0	0	0.8	0.7	0

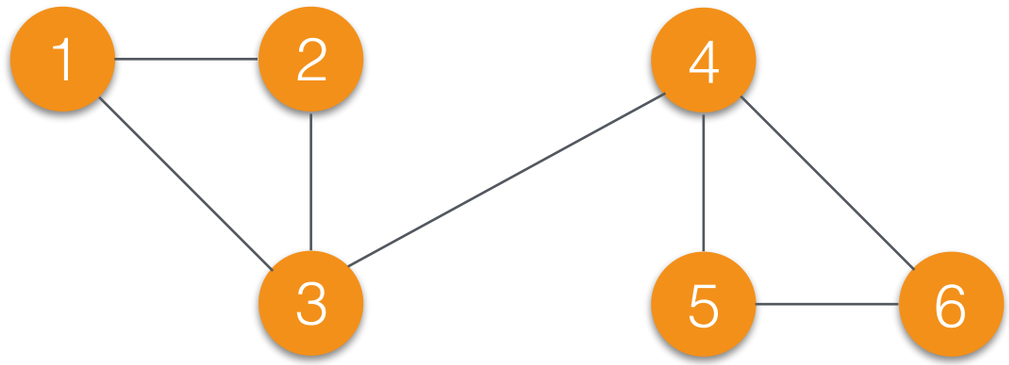
Adjacency matrix W

How to construct graphs?

- Similarity graphs: model local neighborhood relations between data points



$$w_{ij} = \exp(-\gamma \| x_i - x_j \|^2)$$



Graph Laplacian

1.4	-0.8	-0.6	0	0	0
-0.8	1.6	-0.8	0	0	0
-0.6	-0.8	1.6	-0.2	0	0
0	0	-0.2	1.7	-0.7	-0.8
0	0	0	-0.7	1.4	-0.7
0	0	0	-0.8	-0.7	1.5

Laplacian matrix L
(graph Laplacian)

=

1.4	0	0	0	0	0
0	1.6	0	0	0	0
0	0	1.6	0	0	0
0	0	0	1.7	0	0
0	0	0	0	1.4	0
0	0	0	0	0	1.5

Degree matrix G

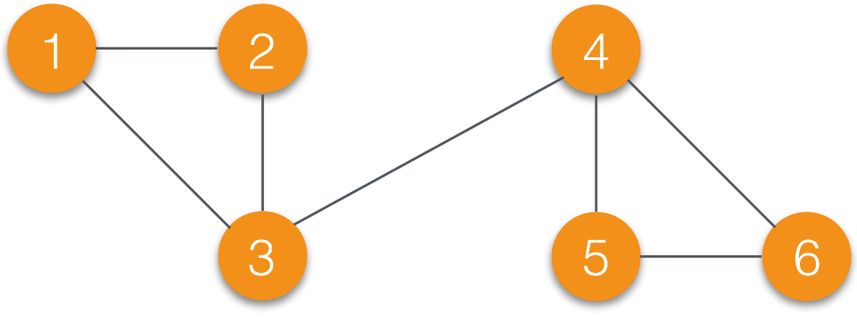
—

0	0.8	0.6	0	0	0
0.8	0	0.8	0	0	0
0.6	0.8	0	0.2	0	0
0	0	0.2	0	0.7	0.8
0	0	0	0.7	0	0.7
0	0	0	0.8	0.7	0

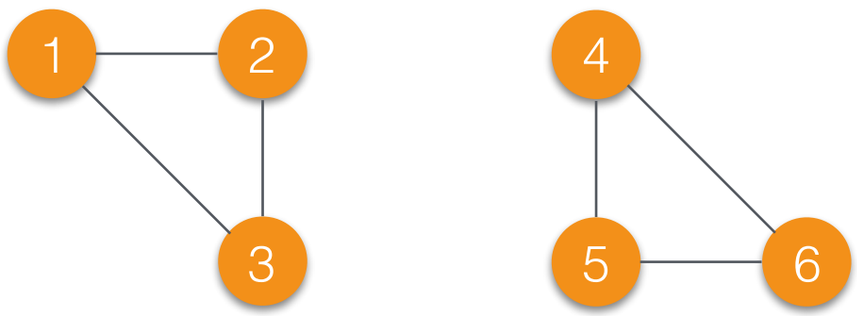
Adjacency matrix W

- L is positive semi-definite. The smallest eigenvalue of L is 0 corresponding to constant eigenvector ($0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$).

Eigenvalue and eigenvector



1.4	-0.8	-0.6	0	0	0	1	
-0.8	1.6	-0.8	0	0	0	1	
-0.6	-0.8	1.6	-0.2	0	0	1	
0	0	-0.2	1.7	-0.7	-0.8	1	= ?
0	0	0	-0.7	1.4	-0.7	1	
0	0	0	-0.8	-0.7	1.5	1	

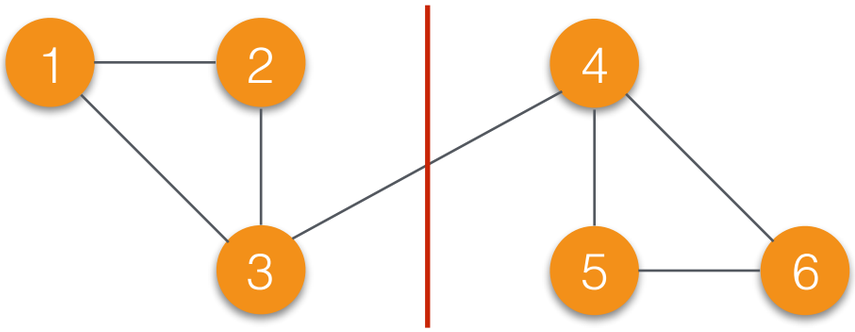


Graph Laplacian L

1.4	-0.8	-0.6	0	0	0	1	
-0.8	1.6	-0.8	0	0	0	1	
-0.6	-0.8	1.4	0	0	0	1	
0	0	0	1.5	-0.7	-0.8	-1	= ?
0	0	0	-0.7	1.4	-0.7	-1	
0	0	0	-0.8	-0.7	1.5	-1	

- For k connected graphs, L is a block diagonal matrix of k blocks, and k eigenvalues are zeros.

Partitioning a graph into two clusters



graph cut

1.4	-0.8	-0.6	0	0	0
-0.8	1.6	-0.8	0	0	0
-0.8	-0.8	1.6	-0.2	0	0
0	0	-0.2	1.7	-0.7	-0.8
0	0	0	-0.7	1.4	-0.7
0	0	0	-0.8	-0.7	1.5

Graph Laplacian L

0.43
0.43
0.36
-0.36
-0.43
-0.42

2nd eigenvector

- If clusters are connected loosely (small off-block diagonal entries), then the second eigenvectors gets first cut.

- Number of loosely connected graphs is number of eigenvalues close to 0 ($\lambda_2 = 0.1179$ in this example).

Spectral clustering algorithm

- Build Z by finding m eigenvectors corresponding to m smallest eigenvalues of L (ignoring zero eigenvalues).

$$Z = \begin{array}{|c|c|c|} \hline z_{11} & z_{12} & \dots \\ \hline z_{21} & z_{22} & \dots \\ \hline z_{31} & z_{32} & \dots \\ \hline z_{41} & z_{42} & \dots \\ \hline z_{51} & z_{52} & \dots \\ \hline z_{61} & z_{62} & \dots \\ \hline \end{array}$$

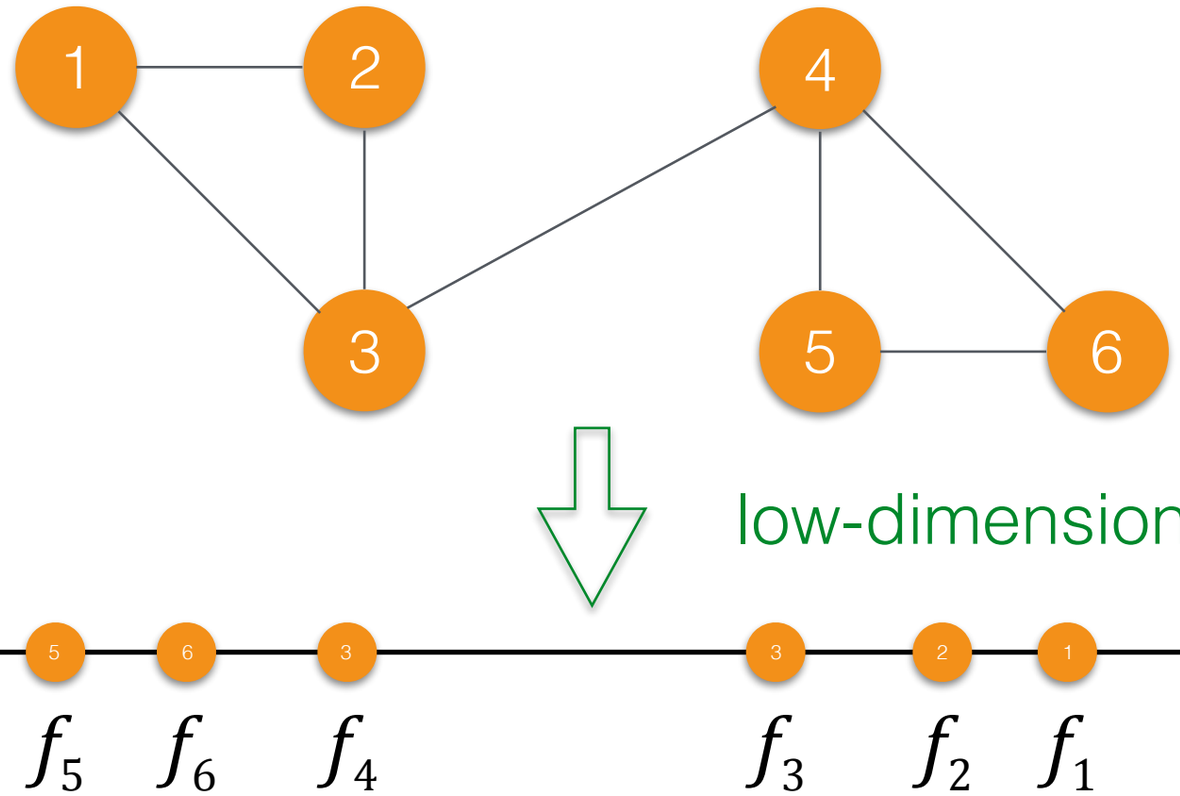
⇨ new coordinates

Dimensionality Reduction
N x N → N x m

- Data are projected into a lower-dimensional space (the spectral/ eigenvector domain) where they are easily separable.

- Apply a standard method (e.g. K-means, hierarchical clustering, etc.) on rows of Z

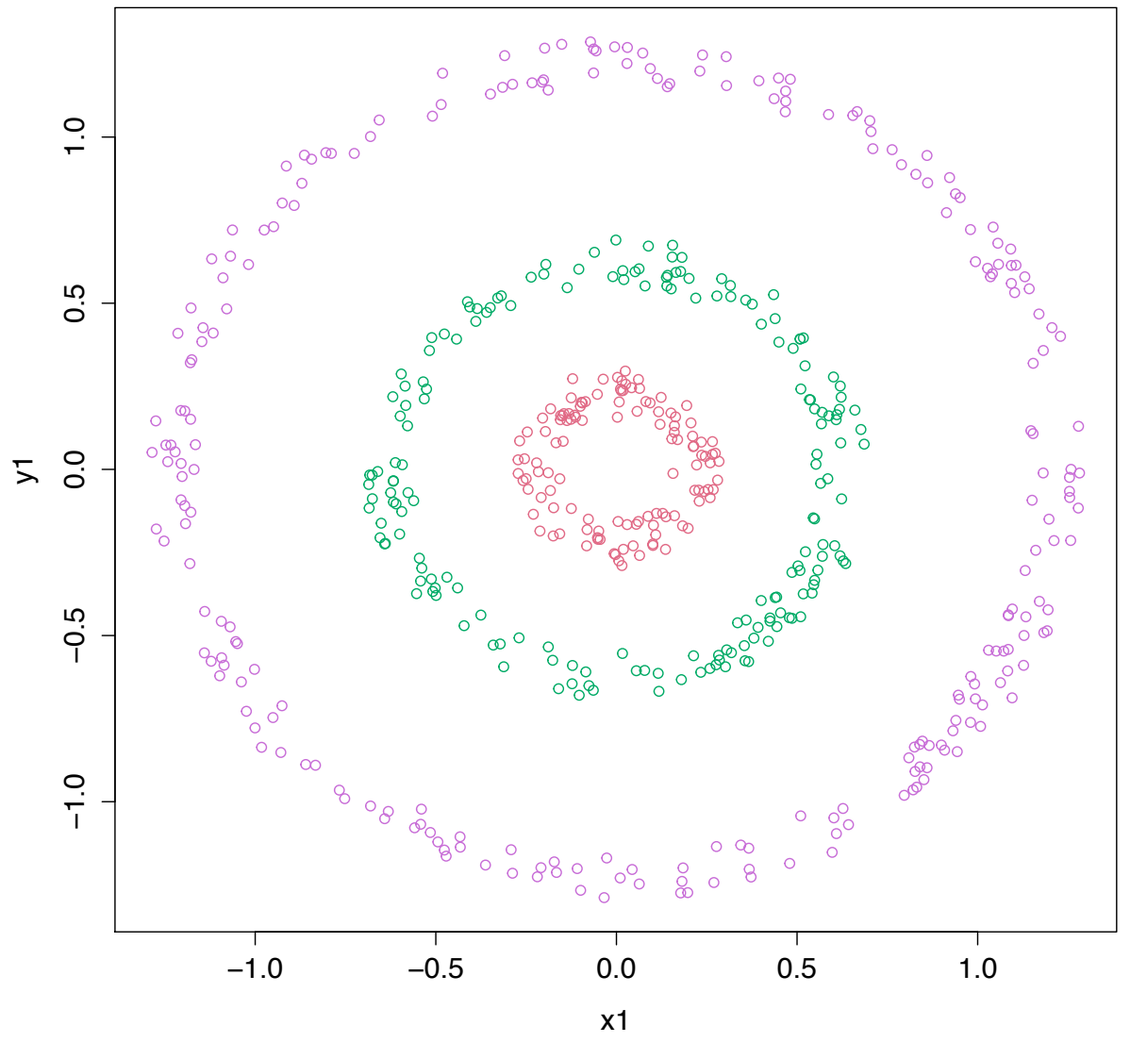
Why it works?



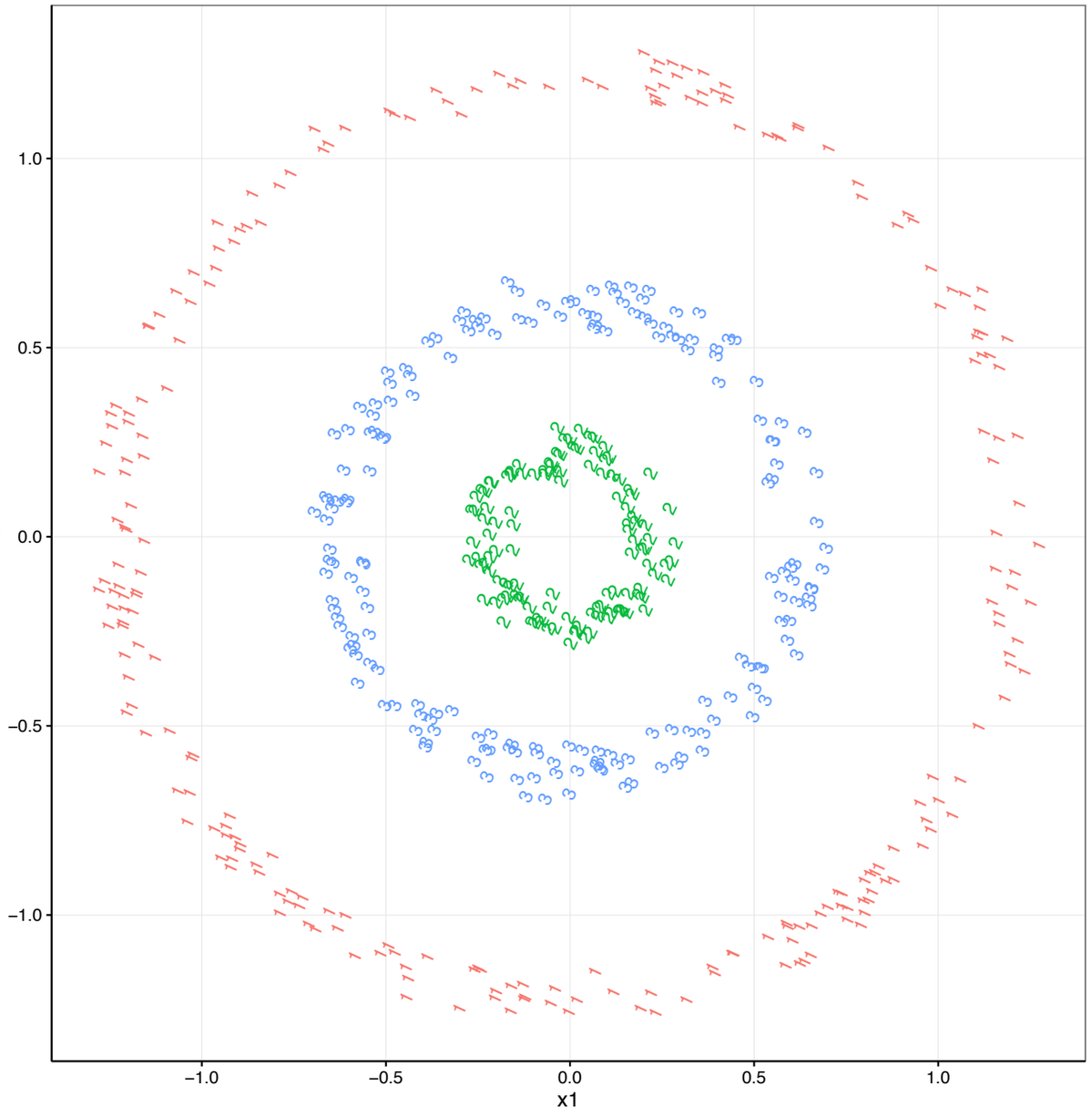
$$\begin{aligned}\mathbf{f}^T\mathbf{L}\mathbf{f} &= \sum_{i=1}^{N} g_i f_i^2 - \sum_{i=1}^{N}\sum_{i'=1}^{N} f_i f_{i'} w_{ii'} \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} w_{ii'}(f_i - f_{i'})^2.\end{aligned}$$

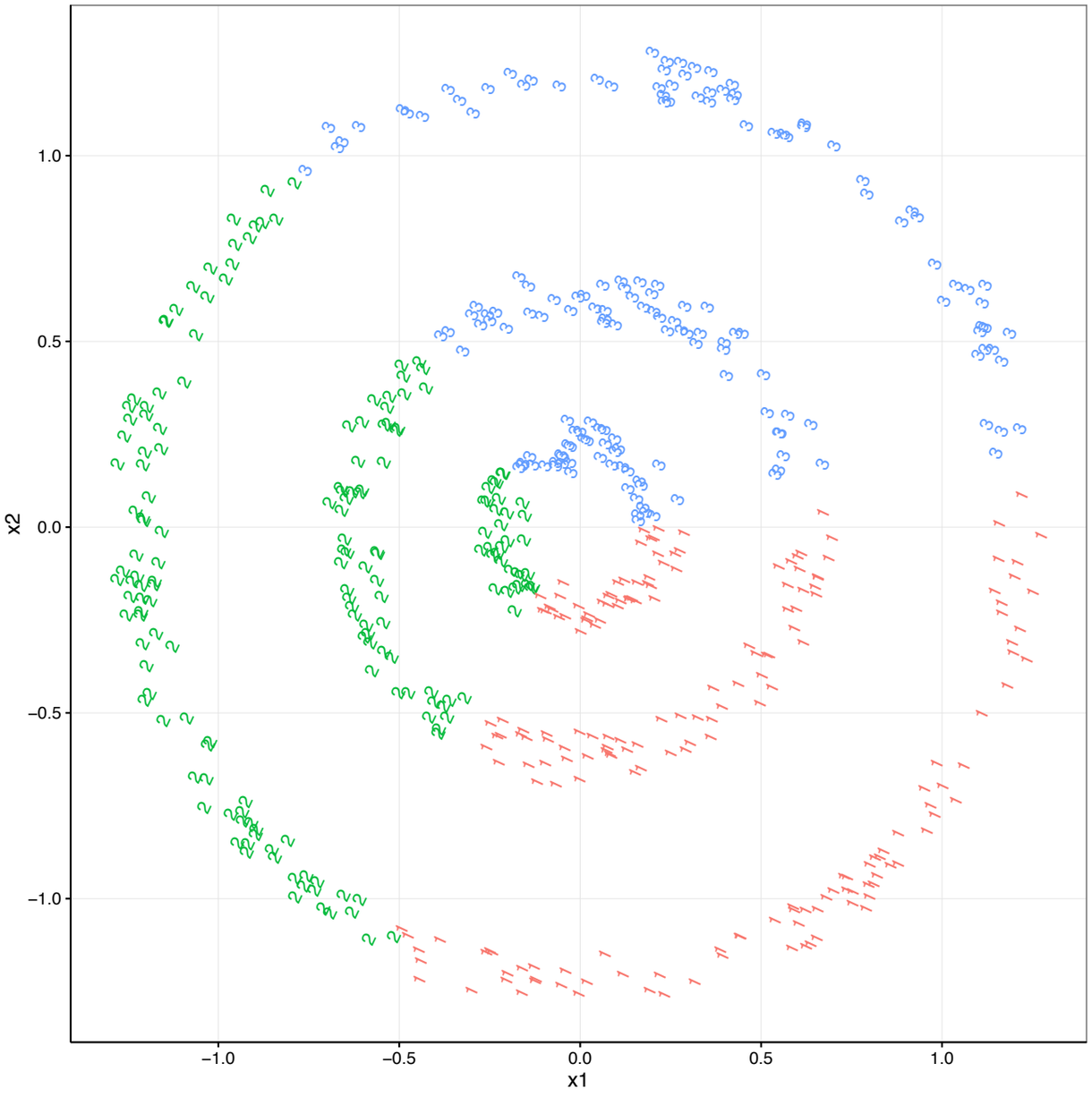
A small value of f^TLf will be achieved if pairs of points with large adjacencies have coordinates f_i and $f_{i'}$ close together.

Example: three concentric circles



- Three clusters are non-convex sets.





- Left: Spectral clustering for three concentric circles
  - Gaussian kernel is used with inverse kernel width = 150
- Right: K-means is applied on raw data

Summary: spectral clustering

- Most stable clustering is usually given by the value of k that maximizes the eigen-gap (difference between consecutive eigenvalues)

- Algorithm obtains data representation in the low-dimensional space (using eigenvectors of matrices derived from the data) that can be easily clustered

- It is useful in hard non-convex clustering problems and empirically very successful