SC4/SM4 Data Mining and Machine Learning
Clustering

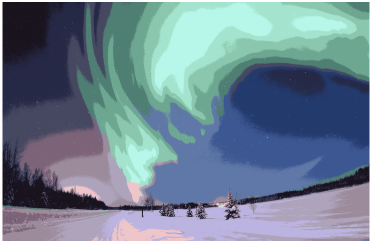
Dino Sejdinovic
Department of Statistics
Oxford

Slides and other materials available at:
http://www.stats.ox.ac.uk/~sejdinov/dmml

Clustering

Clustering

- Many datasets consist of multiple heterogeneous subsets.
- **Cluster analysis**: Given an unlabelled data, want algorithms that automatically group the datapoints into coherent subsets/clusters. Examples:
  - market segmentation of shoppers based on browsing and purchase histories
  - different types of breast cancer based on the gene expression measurements
  - discovering communities in social networks
  - image segmentation



Types of Clustering

- **Model-free** clustering:
  - Defined by **similarity/dissimilarity** among instances within clusters.
- **Model-based** clustering:
  - Each cluster is described using a probability model.

Model-free clustering

- notion of similarity/dissimilarity between data items is central: many ways to define and the choice will depend on the dataset being analyzed and dictated by domain specific knowledge
- most common approach is **partition-based** clustering: one divides n data items into K clusters $C_1, \dots, C_K$ where for all $k, k' \in \{1, \dots, K\}$,

$$C_k \subset \{1, \dots, n\}, \qquad C_k \cap C_{k'} = \emptyset \;\; \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \dots, n\}.$$

- Intuitively, clustering aims to group similar items together and to place separate dissimilar items into different groups
- two objectives can contradict each other (similarity is not a transitive relation, while being in the same cluster is an equivalence relation)

Axiomatic approach

Clustering method is a map $\mathcal{F} : (\mathcal{D} = \{x_i\}_{i=1}^n, \rho) \mapsto \{C_1, \dots, C_K\}$ which takes as an input dataset $\mathcal{D}$ and a dissimilarity function ρ and returns a partition of $\mathcal{D}$. Three basic properties required

- **Scale invariance.** For any $\alpha > 0$, $\mathcal{F}(\mathcal{D}, \alpha\rho) = \mathcal{F}(\mathcal{D}, \rho)$.
- **Richness.** For any partition $C = \{C_1, \dots, C_K\}$ of $\mathcal{D}$, there exists dissimilarity ρ, such that $\mathcal{F}(\mathcal{D}, \rho) = C$.
- **Consistency.** If ρ and ρ' are two dissimilarities such that for all $x_i, x_j \in \mathcal{D}$ the following holds:

$$\begin{aligned} x_i, x_j \text{ belong to the same cluster in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \le \rho(x_i, x_j) \\ x_i, x_j \text{ belong to different clusters in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \ge \rho(x_i, x_j), \end{aligned}$$

then $\mathcal{F}(\mathcal{D}, \rho') = \mathcal{F}(\mathcal{D}, \rho)$.

Kleinberg (2003) proves that there exists no clustering method that satisfies all three properties!

Examples of Model-free Clustering

- **K-means clustering**: a partition-based method into K clusters. Finds groups such that variation within each group is small. The number of clusters K is usually fixed beforehand or various values of K are investigated as a part of the analysis.
- **Spectral clustering**: Similarity/dissimilarity between data items defines a graph. Find a partition of vertices which does not “cut" many edges. Can be interpreted as nonlinear dimensionality reduction followed by K-means.
- **Hierarchical clustering**: nearby data items are joined into clusters, then clusters into super-clusters forming a hierarchy. Typically, the hierarchy forms a binary tree (a **dendrogram**) where each cluster has two “children” clusters. Dendrogram allows to view the clusterings for each possible number of clusters, from 1 to n (number of data items).

K-means

Goal: divide data items into a **pre-assigned number** K **of clusters** $C_1, \dots, C_K$ where for all $k, k' \in \{1, \dots, K\}$,

$$C_k \subset \{1, \dots, n\}, \qquad C_k \cap C_{k'} = \emptyset \;\; \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \dots, n\}.$$

Each cluster is represented using a **prototype** or **cluster centroid** μ_k.
We can measure the quality of a cluster with its **within-cluster deviance**

$$W(C_k, \mu_k) = \sum_{i \in C_k} \|x_i - \mu_k\|_2^2.$$

The overall quality of the clustering is given by the total within-cluster deviance:

$$W = \sum_{k=1}^{K} W(C_k, \mu_k).$$

W is the overall objective function used to select both the cluster centroids and the assignment of points to clusters.

K-means

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|_2^2$$

where $c_i = k$ if and only if $i \in C_k$.

- Given partition $\{C_k\}$, we can find the optimal prototypes easily by differentiating W with respect to μ_k:

$$\frac{\partial W}{\partial \mu_k} = 2 \sum_{i \in C_k} (x_i - \mu_k) = 0 \qquad \Rightarrow \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- Given prototypes, we can easily find the optimal partition by assigning each data point to the closest cluster prototype:

$$c_i = \underset{k}{\operatorname{argmin}} \|x_i - \mu_k\|_2^2$$

But joint minimization over both is computationally difficult.

K-means

The K-means algorithm is a widely used method that returns a **local optimum** of the objective function W, using iterative and alternating minimization.

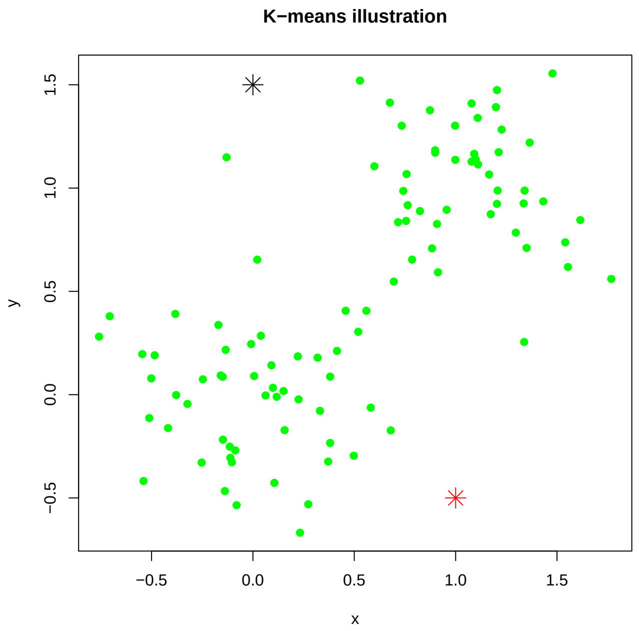
1. Randomly initialize K cluster centroids $\mu_1, \ldots, \mu_K$.
2. **Cluster assignment:** For each $i = 1, \ldots, n$, assign each x_i to the cluster with the nearest centroid,

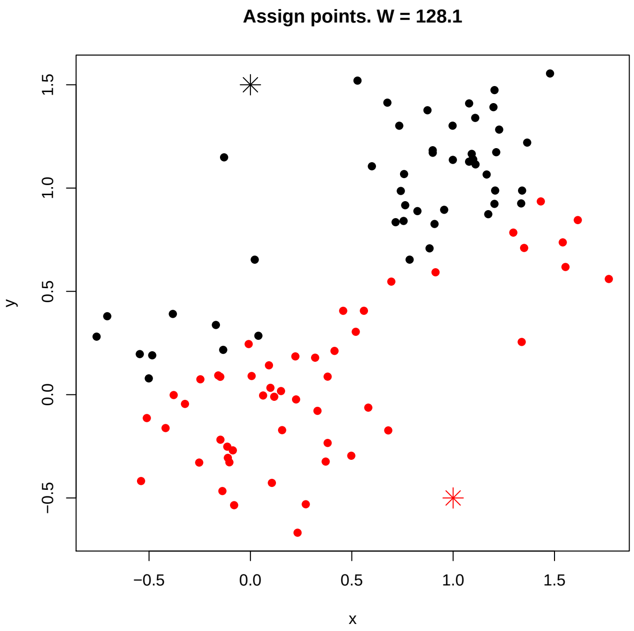
$$c_i := \underset{k}{\operatorname{argmin}} \|x_i - \mu_k\|_2^2$$

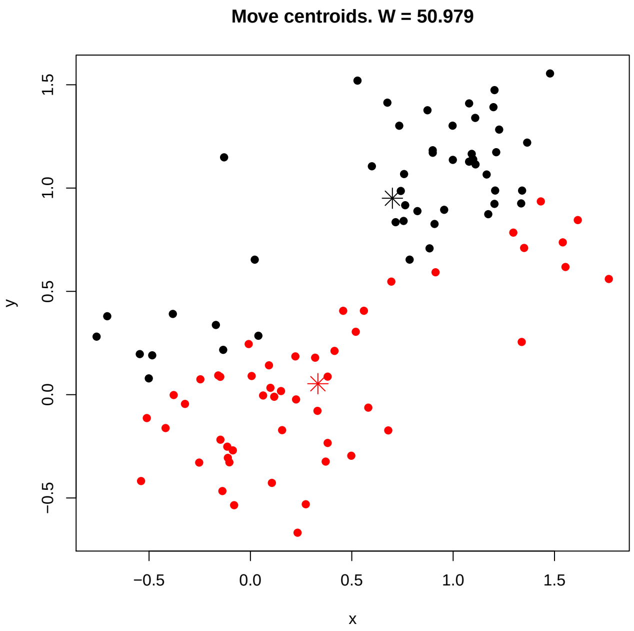
   Set $C_k := \{i : c_i = k\}$ for each k.
3. **Move centroids:** Set $\mu_1, \ldots, \mu_K$ to the averages of the new clusters:

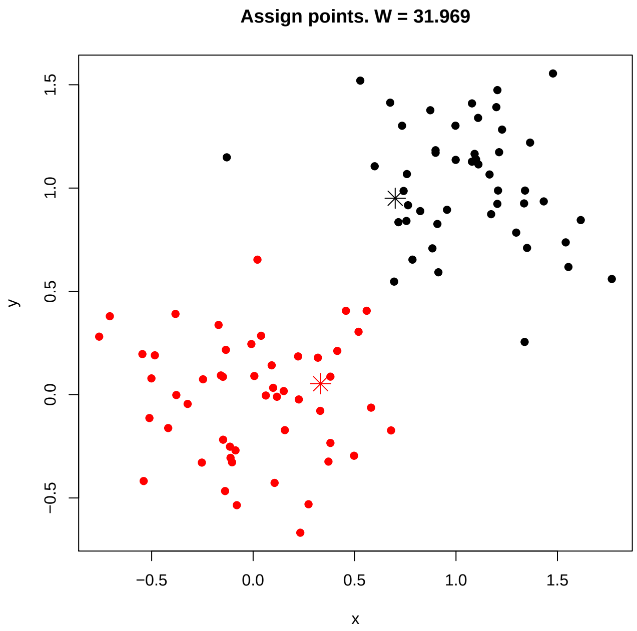
$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

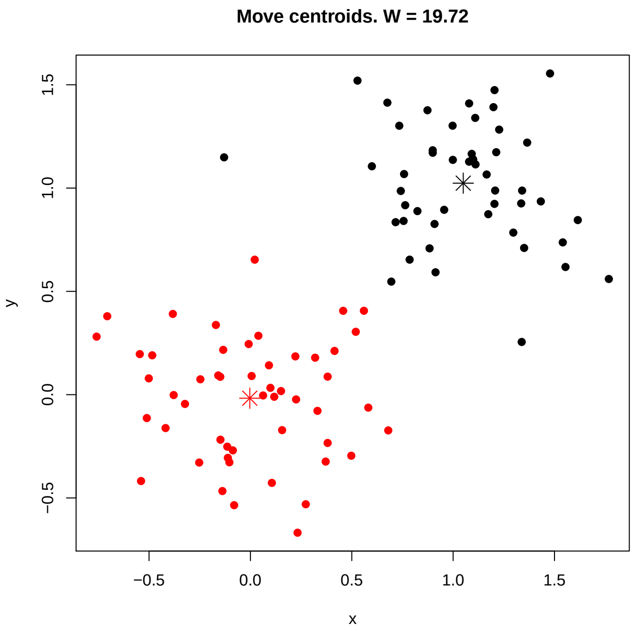
4. Repeat steps 2-3 until convergence.
5. Return the partition $\{C_1, \ldots, C_K\}$ and means $\mu_1, \ldots, \mu_K$.


K−means illustration
y
x
−0.5
0.0
0.5
1.0
1.5

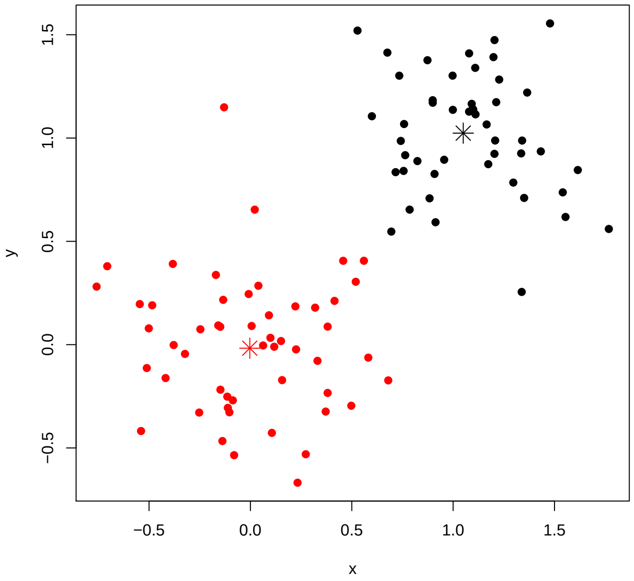

Assign points. W = 128.1
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5

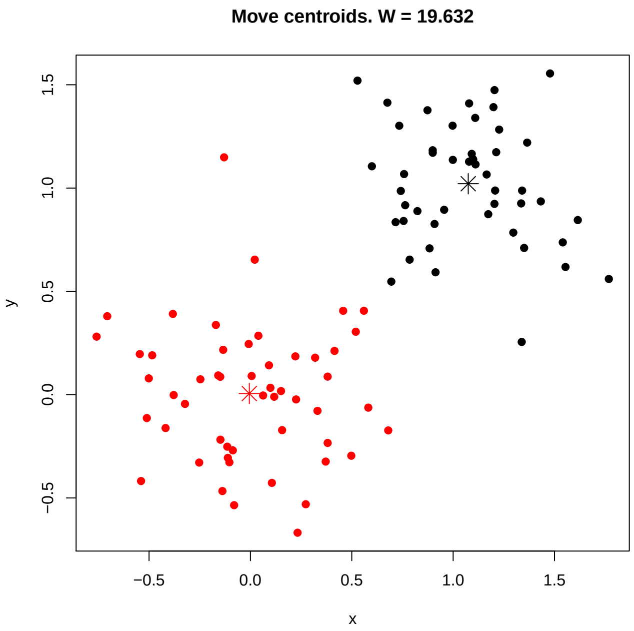

Move centroids. W = 50.979
y
x
−0.5
0.0
0.5
1.0
1.5
−0.5
0.0
0.5
1.0
1.5


Assign points. W = 31.969
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Move centroids. W = 19.72
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5

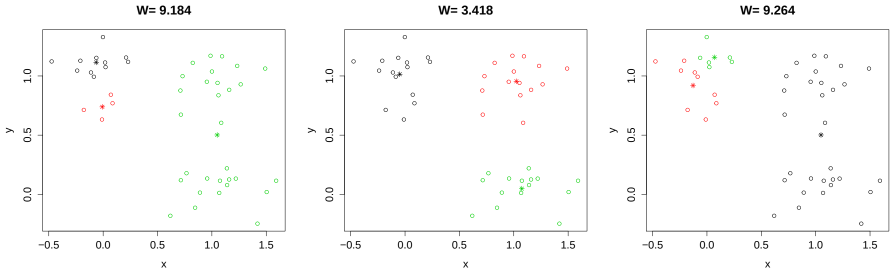
Assign points. W = 19.688




Move centroids. W = 19.632
1.5
1.0
0.5
0.0
−0.5
y
−0.5
0.0
0.5
1.0
1.5
x

K-means

- **The algorithm stops in a finite number of iterations.** Between steps 2 and 3, W either stays constant or it decreases, this implies that we never revisit the same partition. As there are only finitely many partitions, the number of iterations cannot exceed this.
- **The K-means algorithm need not converge to global optimum.** K-means is a heuristic search algorithm so it can get stuck at suboptimal configurations. The result depends on the starting configuration. Typically perform a number of runs from different configurations, and pick the end result with minimum W.



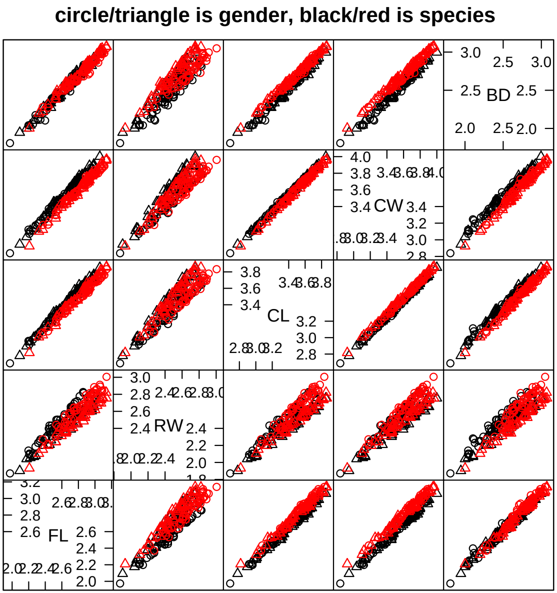
K-means on Crabs

Looking at the Crabs data again.

```
library(MASS)
library(lattice)
data(crabs)

splom(~log(crabs[,4:8]),
    pch=as.numeric(crabs[,2]),
    col=as.numeric(crabs[,1]),
    main="circle/triangle is gender, black/red is species")
```

K-means on Crabs



K-means on Crabs

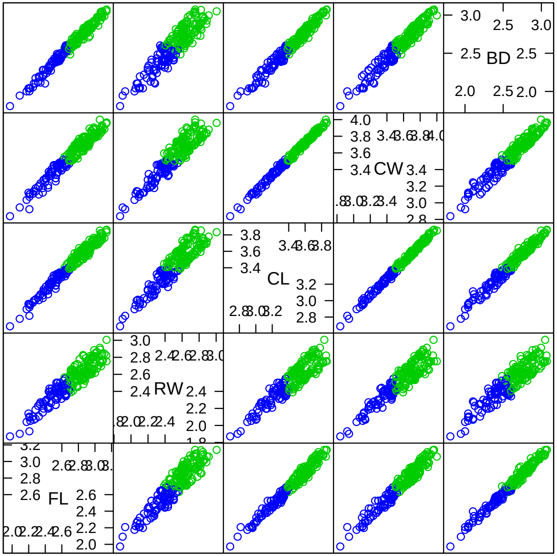
Apply K-means with 2 clusters and plot results.

```
Crabs.kmeans <- kmeans( log(crabs[,4:8]), 2, nstart=1, iter.max=10)

splom(~log(crabs[,4:8]),
      col=Crabs.kmeans$cluster+2,
      main="blue/green is cluster; finds big/small")
```

K-means on Crabs

blue/green is cluster finds big/small



K-means on Crabs

'Whiten' or 'sphere'[1] the data using PCA.

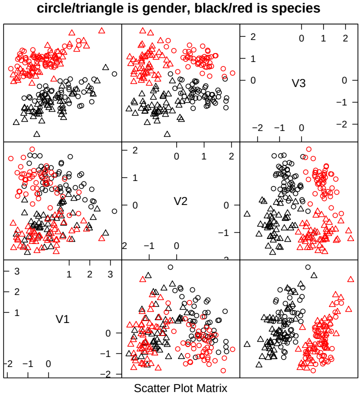
```
pcp <- princomp( log(crabs[,4:8]) )
Crabs.sphered <- pcp$scores %*% diag(1/pcp$sdev)
splom( ~Crabs.sphered[,1:3],
    col=as.numeric(crabs[,1]),
    pch=as.numeric(crabs[,2]),
    main="circle/triangle is gender, black/red is species")
```

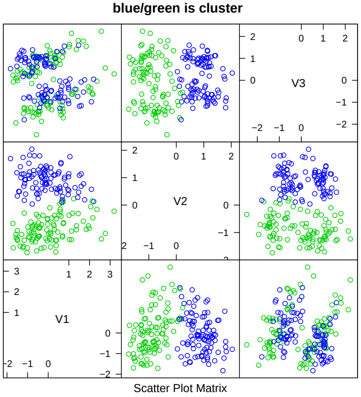
And apply K-means again.

```
Crabs.kmeans <- kmeans(Crabs.sphered, 2, nstart=1, iter.max=20)
splom( ~Crabs.sphered[,1:3],
      col=Crabs.kmeans$cluster+2, main="blue/green is cluster")
```

[1] Apply a linear transformation so that the covariance matrix is identity.

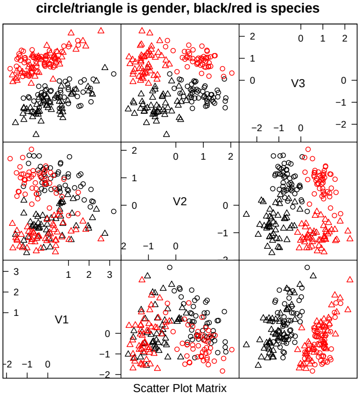
K-means on Crabs

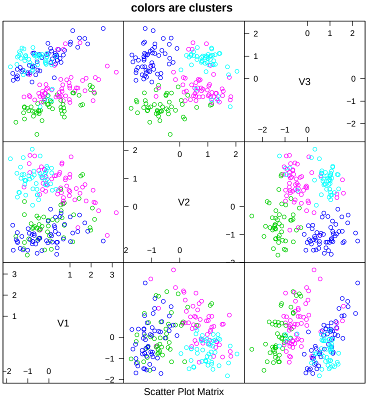




Discovers gender difference...
But the result depends crucially on sphering the data first!

K-means on Crabs with $K = 4$





```
> table(Crabs.kmeans$cluster,Crabs.class)
   Crabs.class
    BF BM OF OM
  1  3  0 41  0
  2 39  8  6  0
  3  8 42  0  0
  4  0  0  3 50
```

K-means Additional Comments

- **Good practice initialization.** Randomly pick K training examples (without replacement) and set $\mu_1, \mu_2, \dots, \mu_K$ equal to those examples
- **Sensitivity to distance measure.** Euclidean distance can be greatly affected by measurement unit and by strong correlations. Can use Mahalanobis distance instead:

  $$\|x - y\|_M = \sqrt{(x-y)^\top M^{-1}(x-y)}$$

  where M is positive semi-definite matrix, e.g. sample covariance.
- **Determination of K.** The K-means objective will always improve with larger number of clusters K. Determination of K requires an additional **regularization** criterion. E.g., in DP-means[2], use

  $$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 + \lambda K$$

[2]DP-means paper.

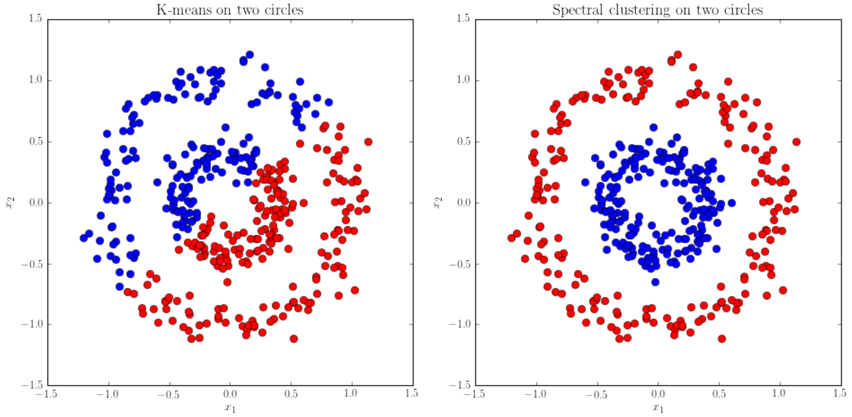
Other partition based methods

Other partition-based methods with related ideas:

- **K-medoids**[3]: requires cluster centroids μ_i to be an observation x_j
- **K-medians**: cluster centroids represented by a median in each dimension
- **K-modes**: cluster centroids represented by a mode estimated from a cluster

[3] See also Affinity propagation.

Nonlinear cluster structures



K-means algorithm will often fail when applied to data with elongated or non-convex cluster structures.

Clustering and Graph Cuts

- Construct a weighted undirected **similarity** graph $G = (\{1, \ldots, n\}, \mathbf{W})$, where vertices correspond to data items and $\mathbf{W}$ is the matrix of edge weights corresponding to pairwise item similarities.
- Partition the graph vertices into $C_1, C_2, \ldots, C_K$ to minimize the **graph cut**.
- The unnormalized **graph cut** across clusters is given by

$$\text{cut}\,(C_1, \ldots, C_K) = \sum_{k=1}^{K} \text{cut}(C_k, \bar{C}_k),$$

  where $\bar{C}_k$ is the complement of C_k and $\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$ is the sum of the weights separating vertex subset A from the vertex subset B, where A and B are disjoint.
- Typically results with singleton clusters, so one needs to balance the cuts by the cluster sizes in the partition. One approach is to consider the notion of “ratio cut"

$$\text{ratio-cut}\,(C_1, \ldots, C_K) = \sum_{k=1}^{K} \frac{\text{cut}(C_k, \bar{C}_k)}{|C_k|}.$$

Graph Laplacian

The **(unnormalized) Laplacian** of a graph $G = (\{1,\dots,n\}, \mathbf{W})$ is an $n \times n$ matrix given by

$$\mathbf{L} = \mathbf{D} - \mathbf{W},$$

where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \text{deg}(i)$, and $\text{deg}(i)$ denotes the **degree** of vertex i defined as

$$\text{deg}(i) = \sum_{j=1}^{n} w_{ij}.$$

- Laplacian always has the column vector $\mathbf{1}$ as an eigenvector with eigenvalue 0 (since all rows sum to zero)
- (**exercise**) Laplacian is a positive semi-definite matrix so all the eigenvalues are non-negative.

Laplacian and Ratio Cuts

Lemma

For a given partition $C_1, C_2, \dots, C_K$ *define the column vectors* $h_k \in \mathbb{R}^n$ *as*

$$h_{k,i} = \frac{1}{\sqrt{|C_k|}} \mathbf{1}_{\{i \in C_k\}}.$$

Then

$$\textit{ratio-cut}(C_1, \dots, C_K) = \sum_{k=1}^{K} h_k^\top \mathbf{L} h_k. \tag{1}$$

To minimize the ratio cut, search for orthonormal vectors h_k with entries either 0 or $1/\sqrt{|C_k|}$ which minimize the RHS in (1).
Equivalent to integer programming so computationally hard.

Laplacian and Ratio Cuts

Lemma

For a given partition $C_1, C_2, \ldots, C_K$ *define the column vectors* $h_k \in \mathbb{R}^n$ *as*

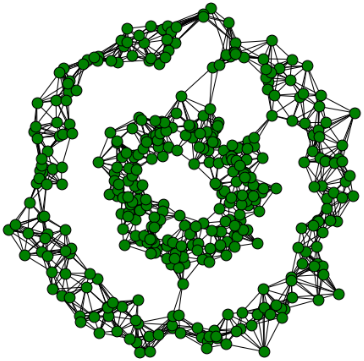
$$h_{k,i} = \frac{1}{\sqrt{|C_k|}} \mathbf{1}_{\{i \in C_k\}}.$$

Then

$$\textit{ratio-cut}\,(C_1, \ldots, C_K) = \sum_{k=1}^{K} h_k^\top \mathbf{L} h_k. \tag{1}$$

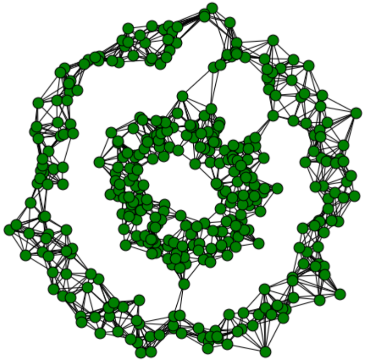
Relaxation: Search for **any collection of orthonormal vectors** h_k in $\mathbb{R}^n$ that minimize RHS in (1) – which corresponds to the eigendecomposition of the Laplacian.

Laplacian and Connected Components



If the original graph is disconnected, in addition to $\mathbf{1}$, there would be other 0-eigenvectors of $\mathbf{L}$, corresponding to the indicators of the connected components of the graph (**Murphy** – Theorem 25.4.1).

Laplacian and Connected Components



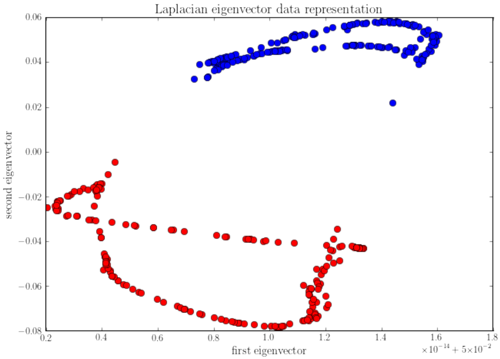
Spectral clustering treats the constructed graph as a “small perturbation" of a disconnected graph.

Eigenvectors as dimensionality reduction

Spectral Clustering. Eigendecompose $\mathbf{L}$ and take the K eigenvectors corresponding to the K smallest eigenvalues – this gives a new "data matrix"

$$\mathbf{Z} = [u_1, \ldots, u_K] \in \mathbb{R}^{n \times K}$$

on which we can apply a more conventional clustering algorithm, such as K-means.



Hierarchical Clustering

- Hierarchically structured data is ubiquitous (genus, species, subspecies, individuals...)
- There are two general strategies for generating hierarchical clusters. Both proceed by seeking to **minimize some measure of overall dissimilarity**.
  - Agglomerative / Bottom-Up / Merging
  - Divisive / Top-Down / Splitting
- Higher level clusters are created by merging clusters at lower levels. This process can easily be viewed by a tree/dendrogram.
- Avoids specifying how many clusters are appropriate.

`hclust, agnes{cluster}`

EU Indicators Data

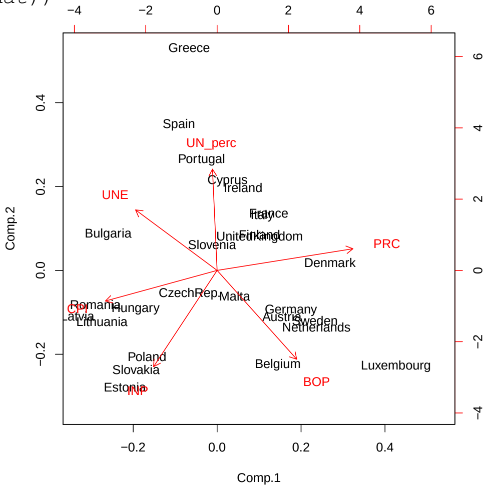
6 Economic indicators for EU countries in 2011.

```
> eu<-read.csv(
    'http://www.stats.ox.ac.uk/~sejdinov/sdmml/data/eu_indicators.csv',sep=' ')
> eu[1:15,]
    Countries abbr    CPI   UNE    INP     BOP     PRC UN_perc
1     Belgium   BE 116.03  4.77 125.59   908.6  6716.5    -1.6
2    Bulgaria   BG 141.20  7.31 102.39    27.8  1094.7     3.5
3   CzechRep.   CZ 116.20  4.88 119.01  -277.9  2616.4    -0.6
4     Denmark   DK 114.20  6.03  88.20  1156.4  7992.4     0.5
5     Germany   DE 111.60  4.63 111.30   499.4  6774.6    -1.3
6     Estonia   EE 135.08  9.71 111.50   153.4  2194.1    -7.7
7     Ireland   IE 106.80 10.20 111.20  -166.5  6525.1     2.0
8      Greece   EL 122.83 11.30  78.22  -764.1  5620.1     6.4
9       Spain   ES 116.97 15.79  83.44  -280.8  4955.8     0.7
10     France   FR 111.55  6.77  92.60  -337.1  6828.5    -0.9
11      Italy   IT 115.00  5.05  87.80  -366.2  5996.6    -0.5
12     Cyprus   CY 116.44  5.14  86.91 -1090.6  5310.3    -0.4
13     Latvia   LV 144.47 12.11 110.39    42.3  1968.3    -3.6
14  Lithuania   LT 135.08 11.47 114.50   -77.4  2130.6    -4.3
15 Luxembourg   LU 118.19  3.14  85.51  2016.5 10051.6    -3.0
```

Data from Greenacre (2012)

EU Indicators Data

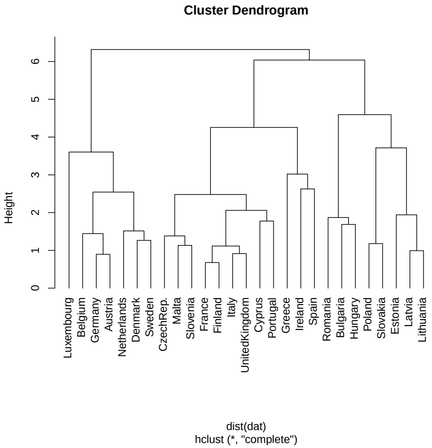
```
dat<-scale(eu[,3:8])
rownames(dat)<-eu$Countries
biplot(princomp(dat))
```

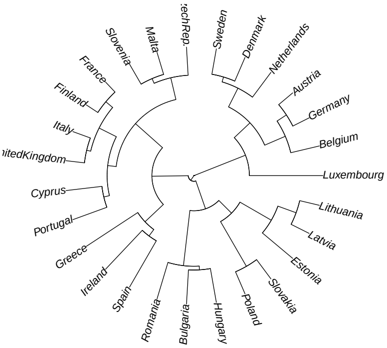


Visualising Hierarchical Clustering

```
> hc<-hclust(dist(dat))
> plot(hc,hang=-1)
```

```
> library(ape)
> plot(as.phylo(hc), type = "fan")
```





Visualising Hierarchical Clustering

Levels in the dendrogram represent a dissimilarity between examples.

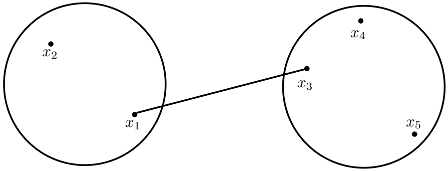
- Tree dissimilarity d_{ij}^T = minimum height in the tree at which examples i and j belong to the same cluster.
- **ultrametric** (stronger than triangle) inequality:

$$d_{ij}^T \leq \max\{d_{ik}^T, d_{kj}^T\}.$$

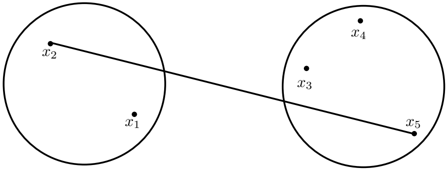
- Hierarchical clustering can be interpreted as an approximation of a given dissimilarity d_{ij} by an ultrametric dissimilarity.

Measuring Dissimilarity Between Clusters

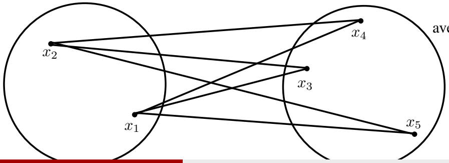
To join clusters C_i and C_j into super-clusters, we need a way to measure the dissimilarity $D(C_i, C_j)$ between them.



single linkage: $d(x_1, x_3)$



complete linkage: $d(x_2, x_5)$



average linkage: $\frac{1}{6}\sum_{i=1}^{2}\sum_{j=3}^{5} d(x_i, x_j)$

Measuring Dissimilarity Between Clusters

To join clusters C_i and C_j into super-clusters, we need a way to measure the dissimilarity $D(C_i, C_j)$ between them.

(a) **Single Linkage**: elongated, loosely connected clusters

$$D(C_i, C_j) = \min_{x,y} \left(d(x, y) | x \in C_i, y \in C_j \right)$$

(b) **Complete Linkage**: compact clusters, relatively similar objects can remain separated at high levels

$$D(C_i, C_j) = \max_{x,y} \left(d(x, y) | x \in C_i, y \in C_j \right)$$

(c) **Average Linkage**: tries to balance the two above, but affected by the scale of dissimilarities

$$D(C_i, C_j) = \text{avg}_{x,y} \left(d(x, y) | x \in C_i, y \in C_j \right)$$

Using Dendrograms

- Different ways of measuring dissimilarity result in different trees.
- Dendrograms are useful for getting a feel for the structure of high-dimensional data though they don't represent distances between observations well.
- Dendrograms depict cluster assignments with respect to increasing values of dissimilarity threshold. Cutting a dendrogram horizontally at a particular height partitions the data into disjoint clusters which are represented by the vertical lines it intersects.
- Despite the simplicity of this idea and the above drawbacks, hierarchical clustering methods provide users with interpretable dendrograms that allow clusters in high-dimensional data to be better understood.

Further reading

- Hastie et al, 14.3
- Murphy, 25
- Shalev-Shwartz and Ben-David, 22
- von Luxburg: Tutorial on Spectral Clustering
- Clustering on scikit-learn