# ICT for Health
# Hierarchical trees

Monica Visintin

Politecnico di Torino

2016/17

## Table of Contents

## Table of Contents

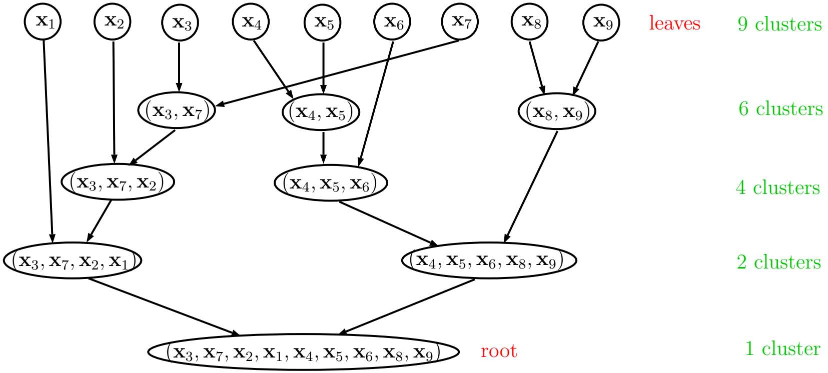
## Types of hierarchical clustering

- Agglomerative (the only case considered in these slides)
- Divisive

## Agglomerative clustering [1]

- We start with $N$ observations, each made of $F$ features
- We evaluate the **distance** between any two observations (in the overall we need to evaluate $N \times (N-1)/2$ distances)
- We generate a cluster by agglomerating the two observations at minimum distance; we have $N-2$ original observations and 1 cluster, in the overal we have $N-1$ objects (an object can be an observation or a cluster)
- We evaluate the **distance** between any two objects (in the overall we need $(N-1) \times (N-2)/2$ distances), we find the minimum distance, we agglomerate the two objects at minimum distance
- We repeat until we have just one object

## Agglomerative clustering [2]


$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$\mathbf{x}_5$
$\mathbf{x}_6$
$\mathbf{x}_7$
$\mathbf{x}_8$
$\mathbf{x}_9$
leaves
9 clusters
$(\mathbf{x}_3, \mathbf{x}_7)$
$(\mathbf{x}_4, \mathbf{x}_5)$
$(\mathbf{x}_8, \mathbf{x}_9)$
6 clusters
$(\mathbf{x}_3, \mathbf{x}_7, \mathbf{x}_2)$
$(\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6)$
4 clusters
$(\mathbf{x}_3, \mathbf{x}_7, \mathbf{x}_2, \mathbf{x}_1)$
$(\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9)$
2 clusters
$(\mathbf{x}_3, \mathbf{x}_7, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9)$
root
1 cluster

## Agglomerative clustering [3]

Advantages:

- We do not need to select the number of clusters $K$
- We get a **binary tree**, which is called **dendrogram**, and we can cut it at the depth we prefer to change the number of clusters

Problems:

- We need to define the distance not only between two observations but also between two objects, which can be two clusters or a cluster and an observation. However, an observation can be thought of as a cluster with just one object

## Agglomerative clustering [4]

- Distances $d$ between observations $\mathbf{x}$ and $\mathbf{y}$ used in the literature:
  1. Euclidean distance (it comes from th assumption of Gaussian pdf):

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{F}(x_k - y_k)^2}$$

  2. Manhattan distance (heuristic):

$$d(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{F}|x_k - y_k|$$

  3. Maximum distance (heuristic):

$$d(\mathbf{x},\mathbf{y}) = \max_{k=1,\ldots,F}|x_k - y_k|$$

## Agglomerative clustering [5]

4. Minkowski (heuristic, $p = 1$ corresponds to Manhattan, $p = 2$ corresponds to Euclidean, $p = \infty$ corresponds to maximum component distance, $p = -\infty$ corresponds to minimum component distance):

$$d(\mathbf{x}, \mathbf{y}) = \left[ \sum_{k=1}^{F} |x_k - y_k|^p \right]^{1/p}$$

## Agglomerative clustering [6]

- Distances $D$ between clusters $A$ and $B$ used in the literature ($d(\mathbf{x}, \mathbf{y})$ as previously defined):
  1. Maximum linkage clustering:
  $$D(A,B) = \max\{d(\mathbf{x},\mathbf{y}), \mathbf{x} \in A, \mathbf{y} \in B\}$$
  2. Minimum linkage clustering:
  $$D(A,B) = \min\{d(\mathbf{x},\mathbf{y}), \mathbf{x} \in A, \mathbf{y} \in B\}$$
  3. Mean linkage clustering ($|A|$ is the number of observations in $A$):
  $$D(A,B) = \frac{1}{|A| \times |B|} \sum_{\mathbf{x}\in A} \sum_{\mathbf{y}\in B} d(\mathbf{x},\mathbf{y})$$

## Agglomerative clustering [7]

4. Centroid linkage clustering:

$$D(A, B) = d(\mathbf{c}_A, \mathbf{c}_B)$$

where $\mathbf{c}_A$ and $\mathbf{c}_B$ are the centroids of $A$ and $B$:

$$\mathbf{c}_A = \frac{1}{|A|} \sum_{\mathbf{x} \in A} \mathbf{x}$$

## Note

A similar algorithm (**Huffman algorithm**) is used to perform **compression of files**: the two symbols (for example in the set of capital letters A, B, C, . . . Z) with the smallest probabilities are clustered together in a super-symbol having probability equal to the sum of probabilities. The obtained binary tree is read from the root to the leaf to get the codeword for the symbol: a left branch corresponds to a bit “0”, a right branch to a bit “1”. In the compressed file, each symbol is substituted with its codeword. By construction, symbols with smaller probabilities have longer codewords, frequent symbols have shorter codewords and this provides the desired compression.

## Hierarchical clustering in Matlab [1]

Given the matrix `X` (with dimension $N$ rows and $F$ columns, $N$ patients, $F$ features) Matlab allows to evaluate the distance between the rows using the command line:

```
≫ D = pdist(X,distance)
```

where `distance` takes one of the following values (others exist):

- `'euclidean'` (Euclidean distance, default)
- `'cityblock'` (Manhattan distance)
- `'minkowski'` ($\left[\sum_{k=1}^{F} |x_k - y_k|^p\right]^{1/p}$, you have to specify the positive integer $p$)
- `'cosine'` ($1 - \mathbf{x}\cdot\mathbf{y}/(\|\mathbf{x}\|\|\mathbf{y}\|)$)
- `'hamming'` (percentage of components that are different in $\mathbf{x}$ and $\mathbf{y}$; this is useful when non-numerical values like “yellow”, “green”, “red” are changed into triplets “100”, “010”, “001”)

## Hierarchical clustering in Matlab [2]

The output of `pdist` is the row vector `D` with $N(N-1)/2$ distances, appropriately ordered. It is possible to get the matrix ($N$ rows $N$ columns) with the distance $d(i,j)$ between thye $i$-th row and the $j$-th row of matrix `X` using the command

```
≫ Dmatr=squareform(D);
```

## Hierarchical clustering in Matlab [3]

It is then necessary to generate the tree, by specifying how to measure the distance between clusters:
$\gg$ `Z = linkage(D,method)`
where `method` takes one of the following values (others exist):

- `'single'` (minimum linkage clustering, default)
- `'average'` (mean linkage clustering)
- `'centroid'` (distance between the centroids of the clusters)
- `'complete'` (maximum distance)
- `'median'` (if $d(k)$ $k = 1, \ldots, K$ are the <u>ordered</u> distances between two vectors in the clusters, then the median is $d(K/2)$ if $K$ is even, or $d((K-1)/2+1)$ if $K$ is odd)

## Hierarchical clustering in Matlab [4]

The output of `linkage` is a matrix with $N$ rows and 3 columns, which describes the binary tree. To view the tree, use:

$\gg$ `dendrogram(Z)`

The assignment of the cluster for each of the rows is provided with the line:

$\gg$ `T = cluster(Z,'maxclust',K)`

which outputs a column vector `T` with $N$ elements, taking values in the range $1, 2, \dots, K$ ($K$ clusters). If `T(n)=2`, then this means that patient `n` belongs to cluster 2.

It is also possible to use the unique command

$\gg$ `T = clusterdata(X,K)`

which corresponds to the sequence of commands:

$\gg$ `D = pdist(X,distance)`

$\gg$ `Z = linkage(D,method)`

$\gg$ `T = cluster(Z,'maxclust',K)`

## Comments

Also hierarchical clustering fails. For example it fails in the case of the arrhythmia data with two clusters (depending on the distance you use, the second cluster has really few elements, from 1 to 10, whereas the first cluster has more than 430 elements).

## Matlab example [1]

Generate 40 vectors, each having 4 elements:

```
x1=randn(1,4);% 1st centroid
x2=randn(1,4);% 2nd centroid
x3=randn(1,4);% 3rd centroid
x4=randn(1,4);% 4th centroid
n=randn(10,4)*0.1;% noise
% the data:
X=[ones(10,1)*x1+n;ones(10,1)*x2+n;ones(10,1)*x3+n;ones(10,1)*x4+n];
[N,F]=size(X);
```

Hierarchical clustering:

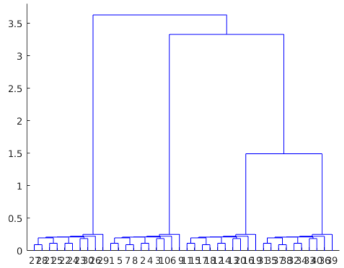
```
d = pdist(X);
s = squareform(d);
Tree = linkage(d);
c = cluster(Tree,'maxclust',4);
```

## Matlab example [2]

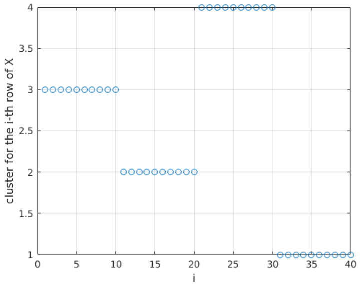
Figures:

```
figure
p=0;% p=0 means that all the leaves must be included in the plotted tree
dendrogram(Tree,p);
% figure
% leafOrder = optimalleaforder(Tree,d);
% dendrogram(Tree,'Reorder',leafOrder)
figure
nn=[1:N];
plot(nn,c,'o'),grid on
xlabel('i')
ylabel('cluster for the i-th row of X')
```

## Matlab example [3]



## Matlab example [4]



## Table of Contents

## How to compare two different clusterings

We've seen three different clustering techniques: hard $K$-means, soft $K$-means, hirarchical clustering. The three techniques in general provide three different clusterings. Which of them is “better”? How can we measure the **performance** of the obtained clustering?
A performance measure is the sum of squared error (SSE):

$$\text{SSE} = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{C}_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

where $\mathbf{x}_i$ is the $i$-th vector belonging to cluster $\mathcal{C}_k$ and $\boldsymbol{\mu}_k$ is the centroid of cluster $\mathcal{C}_k$:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in \mathcal{C}_k} \mathbf{x}_i$$

If the clusters are compact and well separated, then SSE is small. The clustering with the smallest SSE is then the best one.

## Table of Contents

## How to treat non numerical features [1]

If all the features are numerical but **binary**, i.e. they take the only two values 0 and 1, the Hamming distance is typically used:

$$d(\mathbf{x}_i, \mathbf{x}_j) = F - m$$

where $m$ is again the number of matches (0 in both vectors or 1 in both vectors) and $F$ is the number of features. For example the distance bewteen the two vectors

$$\mathbf{x}_1 = [00110]$$

$$\mathbf{x}_2 = [01011]$$

is $d(\mathbf{x}_1, \mathbf{x}_2) = (5-2) = 2$ (the first component is 0 in both vectors, the 4-th component is 1 in both vectors, so 2 matches out of 5 features). You can normalize the Hamming distance and divide it by the number of features:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{F-m}{F}$$

## How to treat non numerical features [2]

If all the features/attributes are **non-numerical** (or nominal), i.e. they give a quality like the color, the opinion, etc. it is impossible to use the distances previously defined (Euclidean, Minkowski, etc). Typically the distance between two nominal vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{F - m}{F}$$

where $F$ is the number of features and $m$ is the number of matches (i.e. equal “values” in the two vectors). For example the distance bewteen the two vectors

$$\mathbf{x}_1 = [\text{yellow}, \text{green}, \text{tall}, \text{nice}]$$

$$\mathbf{x}_2 = [\text{brown}, \text{green}, \text{short}, \text{horrible}]$$

is $d(\mathbf{x}_1, \mathbf{x}_2) = (4 - 1)/4$ (we have 1 match in 4 features).

## How to treat non numerical features [3]

Otherwise, if the feature takes only 4 nominal values (for example A C T G in the DNA strand), then we can map one value with vector $[1, 0, 0, 0]$, another value with vector $[0, 1, 0, 0]$, another value with vector $[0, 0, 1, 0]$ and the last value with vector $[0, 0, 0, 1]$. Then one nominal feature is mapped into 4 binary features; notice that the Hamming distance between each couple of binary vectors is 2, like the distance between two different nominal values.

## Mixed features

What happens if the features are partially numerical, partially binary, partially nominal?
One possibility is to simply add the distances evaluated using for example the Euclidean distance for the features that are numerical, the Hamming distance (normalized or unnormalized) for the binary features, the nominal distance for the nominal features. This is of course a reasonable, but heuristic, way to solve the problem of mixed features, there is no proof that it works better than other methods.

## Table of Contents

## Classification trees in Matlab [1]

Trees can also be build to classify data. If `X` is the matrix ($N$ rows, $F$ columns/features) and vector $c$ ($N$ rows) stores the classes of the rows in `X`, then command

```
≫  tc = fitctree(X,c);
```

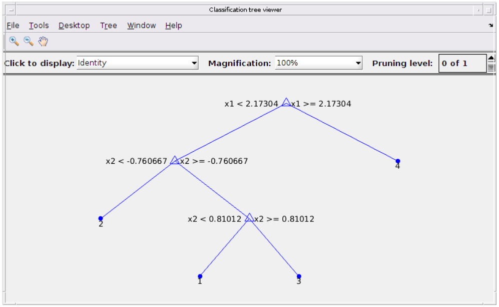
generates the classification tree, and command

```
≫ view(tc,'Mode','graph')
```

shows the decision tree.

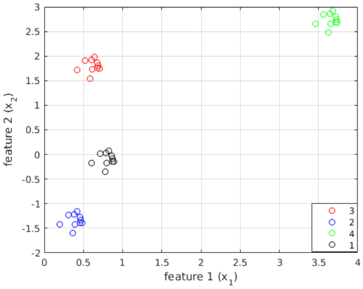
## Classification trees in Matlab [2]

The classification tree for the Matlab example in the previous pages is:



## Classification trees in Matlab [3]

Note that only the first two features are sufficient to classify the 40 vectors (each with 4 features), and the algorithm finds this; below is the plot of the generated 40 vectors considering only the first two features



## Classification trees in Matlab [4]

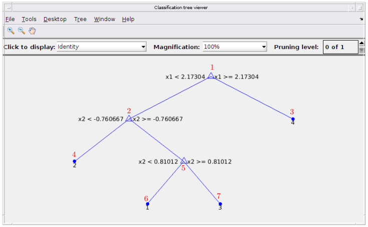
The Matlab command

$\gg$ `view(tc)`

gives the following result:

```
Decision tree for classification
1 if x1<2.17304 then node 2 elseif x1>=2.17304 then node 3 else 1
2 if x2<-0.760667 then node 4 elseif x2>=-0.760667 then node 5 else 1
3 class = 4
4 class = 2
5 if x2<0.81012 then node 6 elseif x2>=0.81012 then node 7 else 1
6 class = 1

7 class = 3
```



## Classification trees [1]

Classification trees work as follow:

- the most “meaningful” feature is found and the initial part of the tree is generated by splitting this feature into two or more ranges (two subranges correspond to a binary tree)
- each subtree is analyzed and again the most “meaningful” feature is found and split into subranges that generate a new part of the tree
- the operation is repeated until all the features are analyzed

How to find the most “meaningful” feature will be explained in a future lecture, when algorithm C4.5 will be described in detail. From the point of view of the decision regions, the hierarchical classification gives rise to decision regions which are hyper-rectangles of the type $\mathcal{R}_k = \{x_1^i(k) \leq x_1 \leq x_1^s(k), x_2^i(k) \leq x_2 \leq x_2^s(k), \ldots, x_F^i(k) \leq x_F \leq x_F^s(k), \}$. It is thus convenient to first apply the Karhunen-Loeve method (PCA) to get uncorrelated features.

## Classification trees in Matlab

In Matlab, being `X` the matrix with $N$ rows and $F$ features (with removed mean value for all the features), you first estimate the covariance matrix and diagonalize it to get the matrix with the eigenvectors

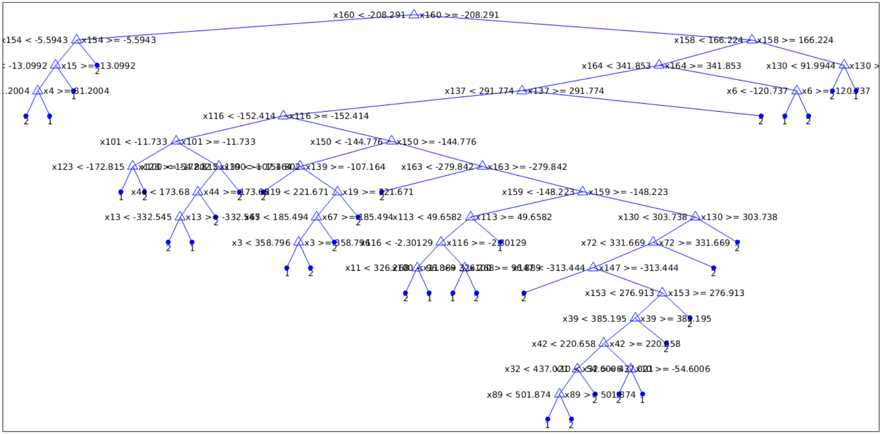
```
≫ X=X-ones(N,1)*mean(X);
≫ R=X'*X/N;
≫ [U,D]=eig(R);
≫ Y=X*U*sqrt(inv(D));
```

then you apply the Matlab function `fitctree` to get the decision tree

```
≫  tc = fitctree(Y,c);
≫ view(tc,'Mode','graph')
```

## Classification for arrhythmia

### Only 164 new features retained, out of the original 257


x160 < -208.291 x160 >= -208.291
x154 < -5.5943 x154 >= -5.5943
x158 < 166.224 x158 >= 166.224
-13.0992 x15 >= 13.0992
x164 < 341.853 x164 >= 341.853
x130 < 91.9944 x130 >
.2004 x4 >= 81.2004
x137 < 291.774 x137 >= 291.774
x6 < -120.737 x6 >= 120.737
x116 < -152.414 x116 >= -152.414
x101 < -11.733 x101 >= -11.733
x150 < -144.776 x150 >= -144.776
x123 < -172.815
x139 >= -107.164
x163 < -279.842 x163 >= -279.842
x44 < 173.68 x44 >= 173.68
x19 < 221.671 x19 >= 221.671
x159 < -148.223 x159 >= -148.223
x13 < -332.545 x13 >= -332.545
x67 < 185.494 x67 >= 185.494
x113 < 49.6582 x113 >= 49.6582
x130 < 303.738 x130 >= 303.738
x3 < 358.796 x3 >= 358.796
x116 < -2.30129 x116 >= -2.30129
x72 < 331.669 x72 >= 331.669
x11 < 326.268
x147 >= -313.444
x153 < 276.913 x153 >= 276.913
x39 < 385.195 x39 >= 385.195
x42 < 220.658 x42 >= 220.658
x32 < 437.021
x21 >= -54.6006
x89 < 501.874 x89 >= 501.874

## Table of Contents

## Laboratory #5 [1]

- In this lab, we use clustering/classification trees on data that are better suited for this analysis than those on arrhythmic patients: chronic kidney disease
- Download from `https://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease` the file `chronic_kidney_disease.arff` (with the data) and file `chronic_kidney_disease.names` with the meanings of the collected data.
- Import `chronic_kidney_disease.arff` as `cell array` using the Matlab import data function (don't consider the first 27 rows). Note that some of the features take nominal values, which are however binary (either normal or abnormal, either yes or no, etc). Other features are numerical, but Matlab import them as strings in the cell structure, and therefore they must also be converted into numbers, to be later processed by Matlab. The last column

## Laboratory #5 [2]

stores the classification of the doctors as “ckd” (chronic kidney disease) or “notckd” (not chronic kidney disease).

- Apparently, it is not possible to work on columns of cell arrays (at least Matlab 2015 does not allow to do it), and therefore each element of the cell array must be converted to a numerical value within two nested for instructions (one for the rows, and one for the columns). We suggest that you create the list of keywords that must be converted and the vector of their corresponding numerical values:

```
keylist={'normal','abnormal','present','notpresent','yes','no',
'good','poor','ckd','notckd','?',''};
keymap=[0,1,0,1,0,1,0,1,2,1,NaN,NaN];
```

  If `a` is the cell array with the nominal values, and `b` is the numerical mattrix with the same size, then you can perform the conversion through the following lines of code:

```
c=strtrim(akr,kc);% removes blanks
check=strcmp(c,keylist);% check(i)=1 if c==keylist(i)
```

## Laboratory #5 [3]

```
if sum(check)==0
 b(kr,kc)=str2num(akr,kc);% from text to numeric
else
 ii=find(check==1);
 b(kr,kc)=keymap(ii);% use the lists
end;
```

- Perform clustering (two clusters) using the Matlab functions `pdist`, `linkage`, `cluster` and plot the obtained tree. Of course don't consider the column with the classes “ckd” and “notckd” for the hierarchical tree generation. Note: patients with NaN attributes are not considered by Matlab for the generation of the tree and they are therefore set aside in the picture with the tree. Compare the clusters automatically generated by Matlab and the classification given by the medical doctors; if possible evaluate the error probability

## Laboratory #5 [4]

- Perform classification (this time you must use the column with the classes “ckd” and “notckd”), using the Matlab function `fitctree` and plot the decision tree using the Matlab function `view`. Then implement the decision as specified by the decision tree and evaluate the error probability. In your report, comment the results, specifically checking the physical meaning of the features involved in the decision tree.
- Write the report.