Advanced Algorithmics

Clustering

Jaak Vilo

2009 Spring
Topics

• What is clustering
• Hierarchical clustering
• K-means
  + K-medoids
• SOM
• Fuzzy
• EM
• ...

Unsupervised vs. Supervised

Find groups inherent to data (clustering)

Find a “classifier” for known classes
Clustering

- An old problem
- Many methods
- No single best “suites all needs” method
# Vehicle Example

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Top speed km/h</th>
<th>Colour</th>
<th>Air resistance</th>
<th>Weight Kg</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>220</td>
<td>red</td>
<td>0.30</td>
<td>1300</td>
</tr>
<tr>
<td>V2</td>
<td>230</td>
<td>black</td>
<td>0.32</td>
<td>1400</td>
</tr>
<tr>
<td>V3</td>
<td>260</td>
<td>red</td>
<td>0.29</td>
<td>1500</td>
</tr>
<tr>
<td>V4</td>
<td>140</td>
<td>gray</td>
<td>0.35</td>
<td>800</td>
</tr>
<tr>
<td>V5</td>
<td>155</td>
<td>blue</td>
<td>0.33</td>
<td>950</td>
</tr>
<tr>
<td>V6</td>
<td>130</td>
<td>white</td>
<td>0.40</td>
<td>600</td>
</tr>
<tr>
<td>V7</td>
<td>100</td>
<td>black</td>
<td>0.50</td>
<td>3000</td>
</tr>
<tr>
<td>V8</td>
<td>105</td>
<td>red</td>
<td>0.60</td>
<td>2500</td>
</tr>
<tr>
<td>V9</td>
<td>110</td>
<td>gray</td>
<td>0.55</td>
<td>3500</td>
</tr>
</tbody>
</table>
Vehicle Clusters

- Lorries
- Sports cars
- Medium market cars

Weight [kg] vs. Top speed [km/h]
Terminology

- Object or data point
- Feature space
- Label
- Cluster

Graph:
- Weight [kg]
- Top speed [km/h]
- Sports cars
- Medium market cars
- Lorries
- Feature
Motivation: Why Clustering?

**Problem:** Identify (a small number of) groups of similar objects in a given (large) set of object.

**Goals:**
- Find representatives for homogeneous groups → **Data Compression**
- Find “natural” clusters and describe their properties → ”natural” **Data Types**
- Find suitable and useful grouping → ”useful” **Data Classes**
- Find unusual data object → **Outlier Detection**
Clustering – it’s “easy” (for humans)
- **Edge-Detection** (advantage to smooth contours)

- **Texture clustering**
Clustering cont...
Distance measures: which two profiles are similar to each other?

Euclidean, Manhattan etc.  

Correlation, angle, etc.  

Rank correlation  

Time warping
Distance measures

• How to formally describe which objects are “close” to each other, and which are not
• More than one way to define distances.
• Distance is a metric, if
  – $d(X,X) = 0$
  – $d(X,Y) = d(Y,X) \geq 0$
  – $d(A,B) \leq d(A,C) + d(C,B)$
Some standard distance measures

Euclidean distance

\[ d(f, g) = \sqrt{\sum_{i=1}^{c} (f_i - g_i)^2} \]

Euclidean squared

\[ d(f, g) = \sum_{i=1}^{c} (f_i - g_i)^2 \]

Manhattan (city-block)

\[ d(f, g) = \sum_{i=1}^{c} |f_i - g_i| \]

Average distance

\[ d(f, g) = \frac{1}{c} \sum_{i=1}^{c} (f_i - g_i)^2 \]
Pearson correlation

\[ d(f, g) = 1 - \frac{\sum_{i=1}^{c} (f_i - \bar{f})(g_i - \bar{g})}{\sqrt{\sum_{i=1}^{c} (f_i - \bar{f})^2 \sum_{i=1}^{c} (g_i - \bar{g})^2}} \]

If means of each column are 0, then it becomes:

\[ d(f, g) = 1 - \frac{\sum_{i=1}^{c} f_i g_i}{\sqrt{\sum_{i=1}^{c} f_i^2 \sum_{i=1}^{c} g_i^2}} = 1 - \cos \Theta \]
Chord distance

\[ d(f, g) = \sqrt{2 \left(1 - \frac{\sum_{i=1}^{c} f_i g_i}{\sqrt{\sum_{i=1}^{c} f_i^2 \sum_{i=1}^{c} g_i^2}}\right)} \]

\[ d(f, g) = \sqrt{2(1 - \cos \Theta)} \]

Euclidean distance between two vectors whose length has been normalized to 1

Legendre & Legendre: Numerical Ecology 2nd ed.
Rank correlation

\[ d(f, g) = 1 - \frac{6 \sum_{i=1}^{c} (\text{rank}_{f_i} - \text{rank}_{g_i})}{c(c^2 - 1)} \]

Rank - smallest has rank 1, next 2, etc.
Equal values have rank that is average of the ranks

\[
\begin{array}{cccccc}
f &=& 3 & 17 & 12 & 12 & 8 \\
\text{rank} &=& 1 & 5 & 3.5 & 3.5 & 2 \\
\end{array}
\]
Hierarchical clustering

Performance: $O(dn^2)$

1. All against all distance matrix
2. Linkage strategy – identify closest clusters and merge
Hierarchical clustering

Cluster matrices:
- Minimum distance => Single linkage
- Maximum distance => Complete linkage
- Average distance => Average linkage (UPGMA, WPGMA)

Cluster sequences:
Hierarchical clustering

• Calculate all pairwise distances
  – and assign each object into a singleton cluster

• While more than 1 cluster
  – select smallest distance
  – merge the two clusters
  – update the changed distances after merger
Update distances

• Merge Ca,Cb into C

• Re-calculate all distances D(Ci, C)

• D(Ci, C) = min{ D(Ci, Ca) , D(Ci, Cb) }
Merge Ca, Cb into C

- $D(C_i, C) = \min\{ D(C_i, Ca) , D(C_i, Cb) \}$
  - Single link; Minimal distance

- $D(C_i, C) = \max\{ D(C_i, Ca) , D(C_i, Cb) \}$
  - Complete link; Maximum distance

- $D(C_i, C) = \text{average}\{ D(C_i, Ca) , D(C_i, Cb) \}$
  - $n_a/( n_a+n_b) * D(C_i, C_a) + n_b/( n_a+n_b) * D(C_i, C_b)$
  - UPGMA – Unweighted Pair Group Method Average
Running time for hierarchical clustering

![Graph showing running time for hierarchical clustering.](image-url)
Time

• $O( n^2 )$ distances

• n-1 times merge
  – select smallest distance
  – update all distances to new cluster
Hierarchical clustering output

GENOMES: Yeast

"Cut"

"Zoom"
Design any heat-map coloring scheme

1. Click on the colour squares in the colour map on the left in the order of the colours in the desired palette, from left to right. To create a familiar Green-Black-Red palette, click on a green square, followed by a black and a red square. You'll see the palette being formed below the colour cube map.

2. Pick a discretisation type (linear/exponential/harmonic/histogram), and enter the number of slots into which you'd like to discretise the palette.

3. Click on the "CREATE DISCRETISATION" link. You'll see the palette broken up into chunks of colour. Linear discretisation will contain equal size chunks, and others will have proportionately varied colour stretches.

4. Click on the "Save" button to save this palette in the main colour options screen.
Heat map color schema design
Limits of standard clustering

• Hierarchical clustering is (very) good for visualization (first impression) and browsing
• Speed for modern data sets remains relatively slow (minutes or even hours)
• ArrayExpress database needs some faster analytical tools
• Hard to predict number of clusters (=>Unsupervised)
6200 genes, 80 exp.

Monitor size 1600x1200 pixels

Laptop: 800x600
6200 genes, 80 exp.

Monitor size 1600x1200

Laptop: 800x600

“COLLAPSE”

75 subtrees

Developed and implemented in Expression Profiler in October 2000 by
Figure 1: A biological case study with VisHiC. (A) gene expression matrix and annotated dendrogram with significant clusters; (B) mitochondrion cluster (ID:30189) (C) muscle cluster (ID:25531) (D) annotation box of the mitochondrion cluster, appears when moving the mouse over the dendrogram (E) detailed view of the muscle cluster with heatmap, dendrogram and lineplot (F) table with functional enrichments, including clusters 30189 and 25531. The data presented in the figure comprises microarray measurements of the heart tissue of cardiovascular patients with left ventricular assist device. VisHiC reveals clusters with expected relevant annotations, e.g. mitochondrion, muscle tissue and extracellular matrix (see Results section).
Fast Approximate Hierarchical Clustering using Similarity Heuristics

Hierarchical clustering is applied in gene expression data analysis, number of genes can be 20000+

Hierarchical clustering:
Hierarchy is built by iteratively joining two most similar clusters into a larger one. Each subtree is a cluster.
Fast Hierarchical Clustering

Avoid calculating all $O(n^2)$ distances:

- Estimate distances
- Use pivots
- Find close objects
- Cluster with partial information

Input data

<table>
<thead>
<tr>
<th>$i$</th>
<th>$x_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3.5, 0.9)</td>
</tr>
<tr>
<td>2</td>
<td>(1.5, 3.0)</td>
</tr>
<tr>
<td>3</td>
<td>(1.7, 5.5)</td>
</tr>
<tr>
<td>4</td>
<td>(2.1, 8.0)</td>
</tr>
<tr>
<td>5</td>
<td>(3.0, 8.4)</td>
</tr>
<tr>
<td>6</td>
<td>(5.1, 5.2)</td>
</tr>
<tr>
<td>7</td>
<td>(8.4, 6.5)</td>
</tr>
<tr>
<td>8</td>
<td>(8.3, 2.2)</td>
</tr>
<tr>
<td>9</td>
<td>(8.3, 1.4)</td>
</tr>
</tbody>
</table>

Input data visualized
### Euclidean distances

<table>
<thead>
<tr>
<th>(d_{L^2}(\cdot,\cdot))</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
<th>(x_5)</th>
<th>(x_6)</th>
<th>(x_7)</th>
<th>(x_8)</th>
<th>(x_9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>0.0</td>
<td>2.9</td>
<td>4.9</td>
<td>7.2</td>
<td>7.5</td>
<td>4.6</td>
<td>7.4</td>
<td>5.0</td>
<td>4.8</td>
</tr>
<tr>
<td>(x_2)</td>
<td>2.9</td>
<td>0.0</td>
<td>2.5</td>
<td>5.0</td>
<td>5.6</td>
<td>4.2</td>
<td>7.7</td>
<td>6.8</td>
<td>7.0</td>
</tr>
<tr>
<td>(x_3)</td>
<td>4.9</td>
<td>2.5</td>
<td>0.0</td>
<td>2.5</td>
<td>3.2</td>
<td>3.4</td>
<td>6.8</td>
<td>7.4</td>
<td>7.8</td>
</tr>
<tr>
<td>(x_4)</td>
<td>7.2</td>
<td>5.0</td>
<td>2.5</td>
<td>0.0</td>
<td>1.0</td>
<td>4.1</td>
<td>6.5</td>
<td>8.5</td>
<td>9.1</td>
</tr>
<tr>
<td>(x_5)</td>
<td>7.5</td>
<td>5.6</td>
<td>3.2</td>
<td>1.0</td>
<td>0.0</td>
<td>3.8</td>
<td>5.7</td>
<td>8.2</td>
<td>8.8</td>
</tr>
<tr>
<td>(x_6)</td>
<td>4.6</td>
<td>4.2</td>
<td>3.4</td>
<td>4.1</td>
<td>3.8</td>
<td>0.0</td>
<td>3.5</td>
<td>4.4</td>
<td>5.0</td>
</tr>
<tr>
<td>(x_7)</td>
<td>7.4</td>
<td>7.7</td>
<td>6.8</td>
<td>6.5</td>
<td>5.7</td>
<td>3.5</td>
<td>0.0</td>
<td>4.3</td>
<td>5.1</td>
</tr>
<tr>
<td>(x_8)</td>
<td>5.0</td>
<td>6.8</td>
<td>7.4</td>
<td>8.5</td>
<td>8.2</td>
<td>4.4</td>
<td>4.3</td>
<td>0.0</td>
<td>0.8</td>
</tr>
<tr>
<td>(x_9)</td>
<td>4.8</td>
<td>7.0</td>
<td>7.8</td>
<td>9.1</td>
<td>8.8</td>
<td>5.0</td>
<td>5.1</td>
<td>0.8</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### Average linkage hierarchical clustering
Distances from one pivot

Distances from two pivots

Distances from two pivots

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d(x_i, x_1)$</td>
<td>0.0</td>
<td>2.9</td>
<td>4.9</td>
<td>7.2</td>
<td>7.5</td>
<td>4.6</td>
<td>7.4</td>
<td>5.0</td>
<td>4.8</td>
</tr>
<tr>
<td>$d(x_i, x_6)$</td>
<td>4.6</td>
<td>4.2</td>
<td>3.4</td>
<td>4.1</td>
<td>3.8</td>
<td>0.0</td>
<td>3.5</td>
<td>4.4</td>
<td>5.0</td>
</tr>
</tbody>
</table>
Distances from two pivots

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d(x_i, x_1)$</td>
<td>0.0</td>
<td>2.9</td>
<td>4.9</td>
<td>7.2</td>
<td>7.5</td>
<td>4.6</td>
<td>7.4</td>
<td>5.0</td>
<td>4.8</td>
</tr>
<tr>
<td>$d(x_i, x_6)$</td>
<td>4.6</td>
<td>4.2</td>
<td>3.4</td>
<td>4.1</td>
<td>3.8</td>
<td>0.0</td>
<td>3.5</td>
<td>4.4</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Here we use Chebyshev distance (maximum of differences).

By triangle inequality we get:
Euclidean distance in original plot cannot be smaller than Chebyshev distance here.
Epsilon Grid Order (EGO)

1) Datapoints sorted according to EGO ord
2) Each point is compared with the later points until one hypercube away
Epsilon Grid Order (EGO)

1) Datapoints sorted according to EGO order
2) Each point is compared with the later points until one hypercube away
e.g. \( x_2 \) is compared with the points in the marked hypercubes
Major Clustering Approaches

- **Partitioning algorithms/Representative-based/Prototype-based Clustering**
  
  **Algorithm**: Construct various partitions and then evaluate them by some criterion or fitness function → Kmeans

- **Hierarchical algorithms**: Create a hierarchical decomposition of the set of data (or objects) using some criterion

- **Density-based**: based on connectivity and density functions → DBSCAN, DENCLUE,…

- **Grid-based**: based on a multiple-level granularity structure

- **Model-based**: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other → EM
Representative-Based Clustering

- Aims at finding a set of objects among all objects (called *representatives*) in the data set that best represent the objects in the data set. Each representative corresponds to a cluster.

- The remaining objects in the data set are then clustered around these *representatives* by assigning objects to the cluster of the closest representative.

Remarks:

1. The popular *k-medoid algorithm*, also called PAM, is a representative-based clustering algorithm; K-means also shares the characteristics of representative-based clustering, except that the representatives used by k-means not necessarily have to belong to the data set.

2. If the representative do not need to belong to the dataset we call the algorithms prototype-based clustering. K-means is a *prototype-based* clustering algorithm
K-means, K-medoids, ...

- Partition the data points into K groups

- Each group is centered around its mean or medoid

- Mean is an “abstract point”
- Medoid: most central object
K-means

1. Guess K centres

2. Assign obj to clusters

3. Move C to gravity centres
Representative-Based Clustering ...

(Continued)
Representative-Based Supervised Clustering ...
(continued)

**Objective of RSC**: Find a subset $O_R$ of $O$ such that the clustering $X$ obtained by using the objects in $O_R$ as representatives minimizes $q(X)$; $q$ is an objective/fitness function.
The *K-Means* Clustering Method

- **Given** *k*, the *k-means* algorithm is implemented in 4 steps:
  1. Partition objects into *k* nonempty subsets
  2. Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
  3. Assign each object to the cluster with the nearest seed point.
  4. Go back to Step 2, stop when no more new assignment.
The *K-Means* Clustering Method

- Example
Comments on *K-Means*

**Strength**

- *Relatively efficient:* \(O(t*k*n*d)\), where \(n\) is the number of objects, \(k\) is the number of clusters, and \(t\) is the number of iterations, \(d\) is the number of dimensions. Usually, \(d, k, t \ll n\); in this case, K-Mean’s runtime is \(O(n)\).
- Storage only \(O(n)\)—in contrast to other representative-based algorithms, only computes distances between centroids and objects in the dataset, and not between objects in the dataset; therefore, the distance matrix does not need to be stored.
- Easy to use; well studied; we know what to expect
- Finds *local optimum of the SSE fitness function*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Implicitly uses a fitness function (finds a local minimum for SSE see later) --- does not waste time computing fitness values

**Weakness**

- Applicable only when *mean* is defined --- what about categorical data?
- Need to specify \(k\), the *number* of clusters, in advance
- Sensitive to *outliers*
- Not suitable to discover clusters with non-convex shapes
- Sensitive to initialization; bad initialization might lead to bad results.
We assume that the k-means initialization assigns the green, blue, and brown points to a single cluster; after centroids are computed and objects are reassigned, it can easily be seen that that the brown cluster becomes empty.
Convex Shape Cluster

• Convex Shape: if we take two points belonging to a cluster then all the points on a direct line connecting these two points must also in the cluster.
• Shape of K-means/K-mediods clusters are convex polygons $\subseteq$ Convex Shape.
• Shapes of clusters of a representative-based clustering algorithm can be computed as a Voronoi diagram for the set of cluster representatives.
• Voronoi cells are always convex, but there are convex shapes that are different from those of Voronoi cells.
Voronoi Diagram for a Representative-based Clustering

Each cell contains one representative, and every location within the cell is closer to that sample than to any other sample.

A Voronoi diagram divides the space into such cells.

Voronoi cells define cluster boundary!
K-means clustering

- New centers: center of gravity for a cluster
- Cluster - objects closest to a center
- Start clustering by choosing K initial centers randomly
- Iterate clustering step until no cluster changes
- Deterministic, might get "stuck" in local minimum

Out of 15 requested clusters there were 15 non-empty clusters

Only the graph-lines:

Seeds: Final centers
K-means clustering output

Cluster nr. 3 (size=13) contents:

Cluster nr. 4 (size=37) contents:
K-means

• Finds local optimum
  – vary many times with random start
  – make an “educated guess” to start with
    • e.g. sample the data, perform hierarchical clustering, select K “centers”.
K-medoids

- Choose the cluster center to be one of the existing objects.
- **Why?**
- If more complex data or distance measure the “Real” center could not be found easily
- What is the mean of categorical data?
  - yellow, red, pink?
- Instead of trying to “invent” – use one of the existing objects, whatever the distance measure
Self Organising Maps (SOM)

MxN matrix of neurons, each representing “a cluster”
Object X is put to Wi, to which it is most similar.
Wi and its near surrounding is changed to resemble X more
Train, train, train…

Problem - there is no clear objective function to map D-dimesnional data to 2 dime
The problem is how to find out semantics relationship among lots of information without manual labor.

- How do I know, where to put my new data in, if I know nothing about information's topology?
- When I have a topic, how can I get all the information about it, if I don‘t know the place to search them?
Motivation: The Idea

- Computer know automatically information classification and put them together

Input Pattern 1

Input Pattern 2

Input Pattern 3
Text objects must be automatically produced with semantics relationships.
Self-Organizing Maps: Origins

Self-Organizing Maps

- Ideas first introduced by C. von der Malsburg (1973), developed and refined by T. Kohonen (1982)
- Neural network algorithm using unsupervised competitive learning
- Primarily used for organization and visualization of complex data
- Biological basis: ‘brain maps’

Teuvo Kohonen
Self-Organizing Maps

SOM - Architecture

- Lattice of neurons (‘nodes’) accepts and responds to set of input signals
- Responses compared; ‘winning’ neuron selected from lattice
- Selected neuron activated together with ‘neighbourhood’ neurons
- Adaptive process changes weights to more closely resemble inputs

2d array of neurons

Weighted synapses

Set of input signals
(connected to all neurons in lattice)
Self-Organizing Maps

SOM – Result Example
Classifying World Poverty

Helsinki University of Technology

‘Poverty map’ based on 39 indicators from World Bank statistics (1992)
Initialisation

(i) Randomly initialise the weight vectors $w_j$ for all nodes $j$
(ii) Choose an input vector \( \mathbf{x} \) from the training set

In computer texts are shown as a frequency distribution of one word.

**A Text Example:**

Self-organizing maps (SOMs) are a data visualization technique invented by Professor Teuvo Kohonen which reduce the dimensions of data through the use of self-organizing neural networks. The problem that data visualization attempts to solve is that humans simply cannot visualize high dimensional data as is so technique are created to help us understand this high dimensional data.
(iii) Find the best-matching neuron \( \omega(x) \), usually the neuron whose weight vector has **smallest Euclidean distance** from the input vector \( x \).

The winning node is that which is in some sense ‘closest’ to the input vector.

‘Euclidean distance’ is the straight line distance between the data points, if they were plotted on a (multi-dimensional) graph.

Euclidean distance between two vectors \( \mathbf{a} \) and \( \mathbf{b} \), \( \mathbf{a} = (a_1, a_2, \ldots, a_n) \), \( \mathbf{b} = (b_1, b_2, \ldots, b_n) \), is calculated as:

\[
d_{\mathbf{a}, \mathbf{b}} = \sqrt{\sum_{i} (a_i - b_i)^2}
\]
Weight Update

SOM Weight Update Equation

\[ w_j(t + 1) = w_j(t) + \mu(t) \lambda_{\omega(x)}(j,t) [x - w_j(t)] \]

“The weights of every node are updated at each cycle by adding Current learning rate \( \times \) Degree of neighbourhood with respect to winner \( \times \) Difference between current weights and input vector to the current weights”

Example of \( \mu(t) \)

Example of \( \lambda_{\omega(x)}(j,t) \)

- x-axis shows distance from winning node
- y-axis shows ‘degree of neighbourhood’ (max. 1)
**Example: Self-Organizing Maps**

Animal names and their attributes

<table>
<thead>
<tr>
<th></th>
<th>Dove</th>
<th>Hen</th>
<th>Duck</th>
<th>Goose</th>
<th>Owl</th>
<th>Hawk</th>
<th>Eagle</th>
<th>Fox</th>
<th>Dog</th>
<th>Wolf</th>
<th>Cat</th>
<th>Tiger</th>
<th>Lion</th>
<th>Horse</th>
<th>Zebra</th>
<th>Cow</th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Small</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Medium</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Big</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2 legs</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4 legs</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hair</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hooves</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mane</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Manes</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>likes</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hunt</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Run</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Fly</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Swim</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

A grouping according to similarity has emerged:

- **peaceful**
- **birds**
- **hunters**

[Teuvo Kohonen 2001] Self-Organizing Maps; Springer;
Clustering etc. algorithms

- Hierarchical clustering methods + visualisation
- K-means, Self Organising Maps (SOM)
- SOTA trees (Self Organising Maps + Tree)
- Fuzzy, EM (object can belong to several clusters)
- Graph theory (cliques, strongly connected components)
- Similarity search: X \rightarrow Y \text{ s.t. } d(X,Y)< 0.3
- Model based (rediscover distributions)
- Planar embeddings, Multidimensional scaling
- Principal Component Analysis
- Correspondence analysis
- Independent Component Analysis
Similarity searches
Similarity searches

Query: “cyc1” (cyc1, activator for cyc1, repressor for cyc1)
=> 3 genes + 10 most similar ones for each
= 3 “clusters”
Similarity searches

Expand a tight cluster by other most similar genes:
EM — Expectation Maximization

• EM — A popular iterative refinement algorithm
• An extension to k-means
  – Assign each object to a cluster according to a weight (prob. distribution)
  – New means/covariances are computed based on weighted measures
• General idea
  – Starts with an initial estimate of the parameter vector
  – Iteratively rescores the patterns against the mixture density produced by the parameter vector
  – The rescored patterns are used to update the parameter updates
  – Patterns belonging to the same cluster, if they are placed by their scores in a particular component
• Algorithm converges fast but may not be in global optima

April 14, 2009
The EM (Expectation Maximization) Algorithm

• Initially, randomly assign $k$ cluster centers
• Iteratively refine the clusters based on two steps
  – Expectation step: assign each data point $X_i$ to cluster $C_i$ with the following:
    \[ P(X_i \in C_k) = p(C_k|X_i) = \frac{p(C_k)p(X_i|C_k)}{p(X_i)}, \]

  – Maximization step:
    • Estimation of model parameters
    \[ m_k = \frac{1}{N} \sum_{i=1}^{N} \frac{X_i P(X_i \in C_k)}{\sum_j P(X_i \in C_j)}. \]
Other Clustering Methods

• PCA (Principal Component Analysis)
  – Also called SVD (Singular Value Decomposition)
  – Reduces dimensionality of gene expression space
  – Finds best view that helps separate data into groups

• Supervised Methods
  – SVM (Support Vector Machine)
  – Previous knowledge of which genes expected to cluster is used for training
  – Binary classifier uses ‘feature space’ and ‘kernel function’ to define a optimal ‘hyperplane’
  – Also used for classification of samples- ‘expression fingerprinting’ for disease classification