Revista de Sistemas de Informação da FSMA

n. 4 (2009) pp. 18-36

SISTEMAS DE INFORMAÇÃO

http://www.fsma.edu.br/si/sistemas.html

TUTORIAL

Clustering Techniques

Ricardo Linden

Abstract

This article describes some clustering techniques for feature extraction are described. These methods work well for functional group separation and can be used for further studies.

Keywords *Clustering; Heuristics; K-Means; Neural Nets; Graphs*

1. Introduction

In this section we will describe the main concepts associated with the clustering problem. The reader will realize how difficult the process is and how it can be used. Besides, we will present the basic mathematical concepts needed to understand the algorithms introduced in section 2.2.

1.1 Clustering: Basic Concepts

Clustering analysis is the name given to the group of computational techniques whose goal is to separate objects into groups based on the characteristics displayed by those objects. Its fundamental goal is to place into the same group objects that are similar according to predefined criteria.

Criteria are usually based on a dissimilarity function that receives two objects and calculates the distance between them. The main functions used will be discussed in section 2.1.2. All groups should exhibit high internal homogeneity and high separation (external heterogeneity). This means that elements in a certain group should be similar between themselves and, hopefully, different from elements from different groups.

The clustered objects are also called examples, tuples and/or records. Each object represents a data entry that may be composed of an attribute vector that may include numerical and categorical data (categorical data being a type of field that can assume one value among several predefined values). Examples of numerical data include age (integer), temperature (real), wages (real), while examples of categorical data include DNA bases (A, C, T or G), military rank (soldier, corporal, sergeant, lieutenant, etc) or the fact whether a person is sick or not (a boolean value that may assume the true or false values) (GORDON, 1981).

Clustering analysis is a useful tool for data analysis in several situations. This technique can be used to reduce data set dimension, reducing a huge set of objects into the information of the group center. Given that clustering is a non supervised learning technique[1], it can also be used to extract hidden characteristics from the data and develop hypothesis on its true nature.

Unfortunately, the grouping problem has an exponential complexity, which means that brute force methods such as exhaustive group enumeration, are not viable. For instance, if we intend to separate 100 elements into 5 groups, there are $5^{100} \approx 10^{70}$ possible groupings. Even if we used a computer able to test 10^9 different configurations per second, it would take more than 10^{53} years to finish this task (10^{40} bigger that the Earth's age). Obviously, we need an efficient heuristic that allows us to solve this problem before the Sun cools down and all life on Earth becomes extinct.

1.2 Dissimilarity Metrics

All clustering methods described in the next sections assume that all rlevant relationship between two objects can be described in a matrix that contains in each position a dissimilarity or proximity metric between a pair of objects.

Each entry p_{ij} in the matrix consists on a numeric value that shows how close objects i and j are. Some metrics calculate similarity, while others calculate dissimilarity, but the concepts they describe are essentially the same.

All dissimilarity coefficients are functions $d : \Gamma X \Gamma \Rightarrow \Re$, where Γ is the set of objects which we are working with. Basically, these functions allow us to transform the data matrix, given by:

$$\Gamma = \begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix} \quad (1)$$

into a distance matrix, given by:

$$d = \begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ : & : & : & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix} \quad (2)$$

where entry d(i,j) represents exactly the distance between elements i and j.

All dissimilarity functions must obey the following criteria:

- $d_{ij} \geq 0, \forall i,j \in \Gamma$
- $d_{ij} = d_{ji}, \forall i,j \in \Gamma$. This rule states that the distance between elements does not vary, no matter how we measure it. Hence, the matrix in (2) is shown as inferior triangular. Given that it is symmetric, all values above the diagonal are implicitly defined.

[1] When the process is supervised, it is called classification.

- $d_{ij} + d_{jk} \geq d_{ik}$, $\forall i,j,k \in \Gamma$. This is known as the triangular inequality and basically specified that the smaller distance between two points is a straight line.

If the metric, besides complying with the above stated properties, also complies with the property d(ax,ay)=|a|d(x,y), then it is called a norm. We will see now all the main dissimilarity metrics used in clustering algorithms.

The first metric we mention is called the Minkowski metric, and is given by the following formula:

$$d(i,j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + ... + |x_{ip} - x_{jp}|^q)} \quad (3)$$

As can be derived by the formula, all Minkowski metrics are norms. In order to verify that, multiply all elements by *a*. Inside the root, the value a^q can be factored out and taken out of the root, assuming the value of *a*, as required by the definition.

The most widely known example of this metric is when *q=2*. In this case, we have the famous euclidian distance between two points, given by:

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + ... + |x_{ip} - x_{jp}|^2)} \quad (4)$$

There are two versions of the Minkowski metric that are also popular. The first one is given by *q=1*, when we have the so called Manhattan distance, and the second one is when $q \to \infty$, when we have the so called Chebyshev distance. In the latter, we can come to the following formula for the distance:

$$d(i,j) = \max_k |x_{ik} - x_{jk}| \quad (5)$$

The choice of *q* depends solely on the emphasis desired on the bigger distances. The higher the value of *q*, the higher the metric sensitivity to the bigger distances. For example, let us try to calculate the distance between the points (0,2,1) and (1,18,2). Using the different values of q, we have the following value foe the distances:

- q=1 → d=[|1-0| + |18-2| + |2-1|] = 18
- q=2 → d=[(1-0)^2 + (18-2)^2 + (2-1)^2] $^{1/2}$ = 258$^{1/2}$ = 16,06
- q=3 → d=[(1-0)^3 + (18-2)^3 + (2-1)^3] $^{1/3}$ = 4098$^{1/3}$ = 16,002
- q=∞ → d=[(1-0)$^\infty$ + (18-2)$^\infty$ + (2-1)$^\infty$]$^{1/\infty}$ = 16

As we can see in the example, the higher the value of *q*, the closer the distance is to the bigger distance between the points. If one of the dimensions exceeds the others greatly, it will dominate the final value calculated as *q* goes higher. Whether this is a desirable effect or not depends solely on the nature of your application.

Figure 1 shows us the different interpretation between the different versions of Minkowski metric when the objects are two dimensional. As we can see, the Minkovski metric for *q=2* is exactly the value we were taught in high school algebra, when we were struggling with the Pithagoric theorem.

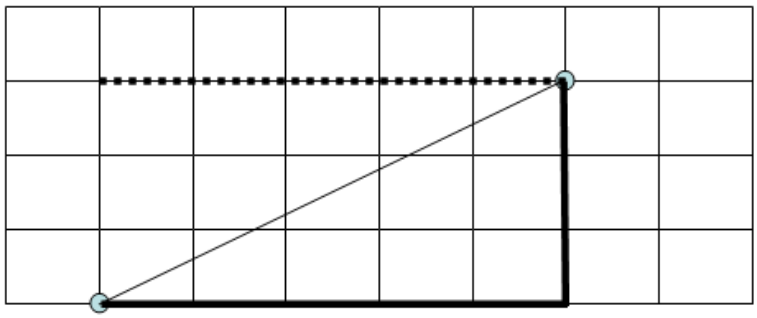

Figure 1. Example of a distance as calculated by the various versions of the Minkowski metric. Euclidian distance (fine line) is calculated as a straight line between the points. Manhattan distance is calculated one block at a time (if we don't go straight, the resulting value will be the same). Chebyschev distance (dotted line) is given by the bigger dimension of the two used to calculate the distance.

Another distance metric easy to calculate is the Camberra distance, which adds the fractionary differences among the coordinates of the pair of objects. Its formula is:

$$d(\vec{x},\vec{y}) = \sum_{i=1}^{p} \frac{|x_i - y_i|}{|x_i| + |y_i|} \quad (6)$$

If one of the coordinates is zero, the term is equal to 1. In all other cases, the term belongs to the interval [0,1]. The total sum always results in a number belonging to [0,p], where p is the vectors dimension.

If both vectors have a coordinate *i* equal to zero, the term *i* is defined arbitrarily as zero (actually it is equal to 0/0). One of the problems with this metric is that the distance is very sensitive to small variations when both coordinates are very close to zero.

Another problem associated with the Canberra distance is that coordinated whose distance is the same, but whose modulus is quite different, will add different contributions to the sum. For instance, assume that coordinate *i* presents the values 1 and 4. Therefore, the contribution of term *i* will be $|1-4| \big/ |1|+|4| = 0{,}6$. Meanwhile, if coordinate *j* presents the values 30 and 33, its contributin will be $|30-33| \big/ |30|+|33| \approx 0{,}05$. That is, the contributions have different magnitude orders, in spite of the fact of their being identical.

Mahalanobis distance is a metric that is different from Euclidian distance because it takes into consideration the correlation between the data sets. The formula for the Mahalanobis distance between two vectors of the same distribution that have a covariance matrix Σ is given by:

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T \Sigma^{-1} (\vec{x}-\vec{y})} \quad (7)$$

If the covariance matrix is the identity matrix, this formula becomes equivalent to the Euclidian distance. In case the covariance matrix is diagonal, Mahalanobis distance becomes a normalized Euclidian distance, whose formula is given by:

$$d(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{P} \frac{(x_i - y_i)^2}{\sigma_i^2}} \quad (8)$$

In this formula, p is the vector dimension and σ_i is the standard deviation of x_i in the sample space.

It can be shown that the surfaces where the Mahalanobis distance is a constant are ellipsoids centered on the average of the sample space. In the special case where all characteristics are not correlated, those surfaces are spheroids, as in the case of Euclidian distance.

Using Mahalanobis distance corrects some of the limitations of Euclidian distance, because the former automatically takes into account the axis scales and the correlations between characteristics. Nevertheless, since there is no free lunch, there is a high price to pay for these advantages. Covariance matrixes can be hard to calculate and both memory usage and computation time increase quadratically with the number of characteristics.

We can also use Mahalanobis distance to measure the distance between an element *x* and a cluster whose average is given by $\mu = \{\mu_1, \mu_2, ..., \mu_n\}$ and whose covariance matrix is given by Σ. In this case, the distance is given by the formula $d = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$. Conceptually, it is as if we were evaluating element pertinence to the cluster not only based on its distance to the center but also on the cluster distribution, calculating the distance of element *x* based on a comparison to the cluster standard deviation. The bigger the calculated value, the higher the number of standard deviations the element will be away from the center of the cluster and the smaller its chance to belong to the cluster.

Outlier detection is one of the most common usages of Mahalanobis distance, because a high value will show that an element is several standard deviations away from the center (given all the axial differences) and, therefore, it is probably an outlier. If we compare to several existing groups, whose distributions are known, this metric can be used as a way to determine to which group this element will probably belong.

Another interesting metric is correlation. It measures the strength of the relationship between two variables. Therefore, it is a similarity measure. The correlation coefficient is represented by the letter "r" and does not have units. Its range of values is [-1,1] where the correlation magnitude represents its strength and its sign represents the type of relationship: positive signs indicate direct association (when one variable increases, so does the other) and negative signs indicate inverse association (when one variable increases, the other decreases). Correlation is calculated using the following formula:

$$r = \frac{1}{n-1} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right) \qquad (9)$$

In this formula, $\bar{x}$ and S_x represent the average and the standard deviations for variable *x* and $\bar{y}$ and S_y the same for variable *y*, while *n* stands for the number of examples in the sample.

1.3 Cluster representation

There are several distinct forms of representing groups created by the clusterign algorithms we describe in this paper. The most common forms are shown in figure 2. Some formats are associated with specific methods, such as dendograms (figure 2.d) who are associated with hierarchical methods, while others are more generic, such as partitions and Venn diagrams (fig. 2a and b, respectively). Tables (figure 2.c) are more useful in cases of methods that use fuzzy logic, but can be used in other case, since their representation power is similar to Venn diagrams. Each format will be further explained when we discuss clustering heuristics in section 2.

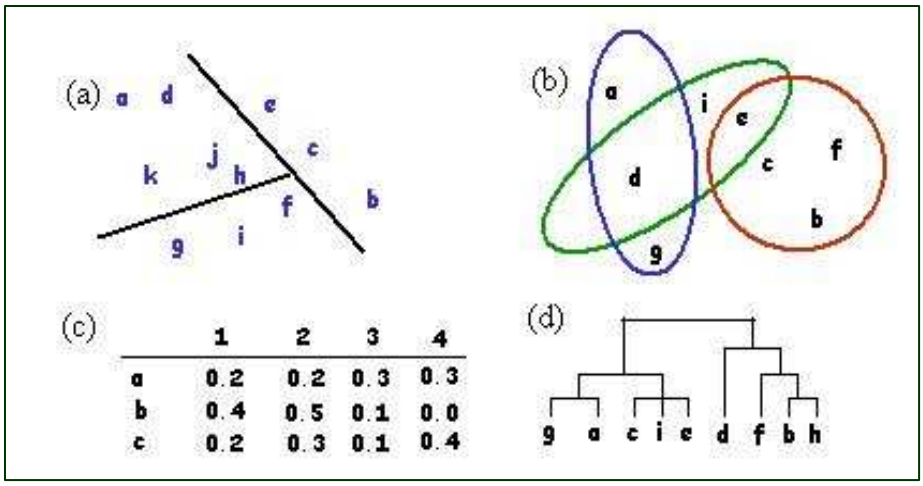

Figure 2. Examples of several ways of representing groups.

1.4 Clustering techniques applications

Clustering techniques have been used in several different areas, including the following:

- *Marketing:* They can help professional of this are to discover groups in their client base in order to create focused marketing strategies to provide better service and enhance their sales (CHIANG, 2003).
- *Land use*: Identification of the possibility to define the allocation of land for agricultural of urban usage based on satellite observation data (LEVIA JR, 2000).
- *Insurance:* identifying groups of people that have car insurance and present a high sinistrality rate (YEOH, 2001).
- *World Wide Web*: grouping documents according to their semantic similarity in order to improve the results offered by search engines (HAMMOUDA, 2002).
- *Earthquake studies*: Analysis of real and synthetic earthquake data in order to find characteristics that can forecast the occurrence of events that preclude seismic events (DZWINNEL, 2005).

2. Clustering Heuristics

In this section we will present some of the widely used clustering algorithms. Each algorithm will be fully described, with a complete discussion on its strengths and weaknesses.

2.1. K-Means

K-Means is a non-hierarchical clustering heuristic that intends to minimizes elements distance to a group of *k* centers given by $\chi=\{x_1,x_2,...,x_k\}$ using an iterative technique. The distance from a point p_i to a group of clusters, given by $d(p_i,\chi)$, is defined as the distance from the point to the cluster closer to it. Hence, the function to be minimized is given by:

$$d(P,\chi) = \frac{1}{n}\sum_{i=1}^{n} d(p_i,\chi)^2$$

The algorithm depends on a parameter k that represents the number of clusters and is defined *ad hoc* by the user. This is usually a problem, given that it is non know *a priori* how many clusters there are.

K-Means algorithm can be described in the following way:

1. Choose k distinct values for the group centers (possibly randomly);
2. Associate each point to the closer center;
3. Recalculate each group center;
4. Repeat steps 2-3 until no element changes group.

This algorithm is extremely fast, usually converging to a stable configuration (one where no element changes group). An example execution of K-Means can be seen in figure 3.

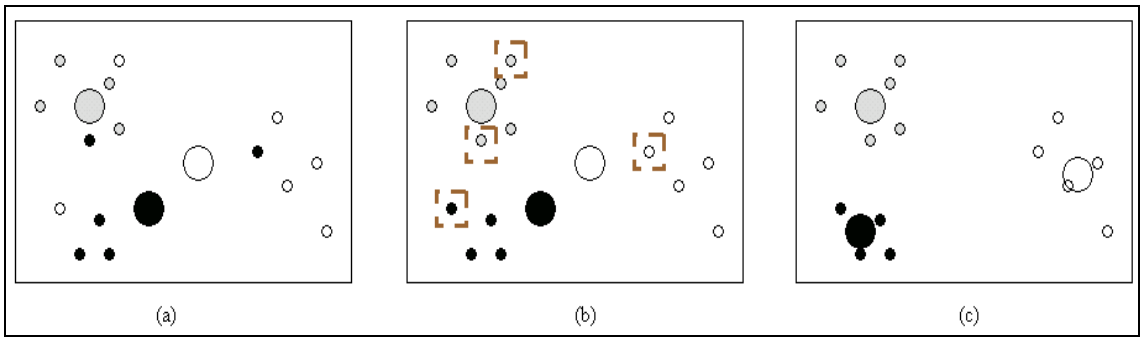

Figure 3. Example execution of the K-Means algorithm. (a) Each element was randomly assigned to one of three groups and the centroids (larger circles) for each group were calculated. (b) The elements were reassigned to the clusters that are closer to them – the elements surrounded by the dotted squares changed group (c) The centroids were recalculated and the groups have arrived to their final forma. If they were not, we would have repeated steps (b) and (c) until the centroids were stable.

This iterative algorithm tends to converge to a minimum of the energy function given above. An eventual problem is that this condition emphasizes homogeneity and ignores cluster separation. This can cause a bad cluster separation in case of bad center initialization. Remember that the algorithm used so far performed this initialization randomly. In figure 4 we can see the effect of bad initialization in a K-Means algorithm.

Another issue that can affect result quality is the user choice for the number of clusters. A number that is too small can make two natural clusters come together while a number that is too big can make a natural cluster to be artificially broken in two.

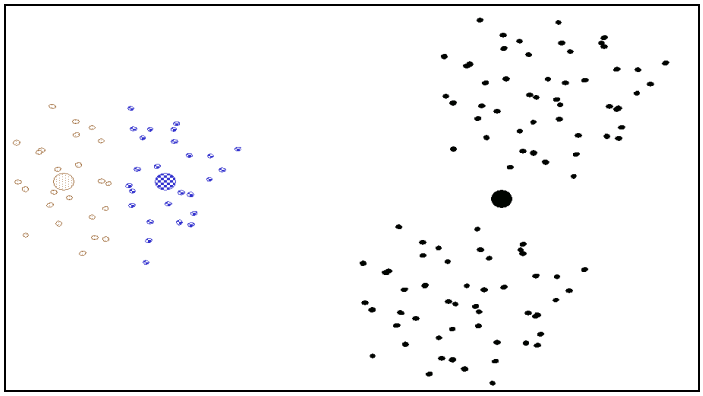

Figura 4. An example of the effect of bad initialization on the results of the K-Means algorithm. There are three natural clusters in this data set, two of which were clustered into the black group. The problem is that the third cluster is separated into two clusters because the initialization created two clusters on the "left" side of the data, where they continued through the iterations.

The bigger advantage of this heuristic is its simplicity. An experienced programmer can implement his version of K-Means in an hour work. Besides, there are several sample programs in many sites that can help a student understand the inherent characteristics of this clustering method. One example of those examples is the one located in the Internet site whose address is http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html (last access in august/2011).

2.2. Hierarchical Algorithms

Hierarchical algorithms create a relationship hierarchy among elements. They are very popular in bioinformatics, specially when applied to phylogenetics, and work very well, even though their theoretical justification based on statistics or information theory is flimsy at best.

There are two versions: the agglomerative that creates bigger sets based on smaller ones and the divisive, that starts with a huge set containing all elements and breaks it one element at a time until all elements are separated one from another.

The agglomerative version works according to the following algorithm:

1. Create a cluster for eachelement;
2. Find the pair of clusters that are most similar, according to a predefined distance measure;
3. Join them into a larger cluster and redo the calculations on distance from this cluster to all others;
4. Repeat steps 2 and 3 until there is a single cluster.

We can show a simple example to explain how hierarchical algorithms work. For example, let us try to cluster the group of points given by S={2; 4; 6,3;9;11,6}.

First we calculate the distances among the points using Euclidian distance (a metric we chose arbitrarily):

$$D = \begin{bmatrix} 0 & & & & \\ 2 & 0 & & & \\ 4{,}3 & 2{,}3 & 0 & & \\ 7 & 5 & 2{,}7 & 0 & \\ 9{,}6 & 7{,}6 & 5{,}3 & 2{,}6 & 0 \end{bmatrix},$$

As we can see, the closer elements are the first (2) and the second (4) ones. We group them as shown in figure 5 (b) and the new distances are recalculated, replacing both elements by their new group center (their average, which is 3). The new distances are:

$$D = \begin{bmatrix} 0 & & & \\ 3{,}3 & 0 & & \\ 6 & 2{,}7 & 0 & \\ 8{,}6 & 5{,}3 & 2{,}6 & 0 \end{bmatrix}$$

At this point, the closer elements are the third (9) and the fourth (11,6) ones and we cluster them, as shown in figure 5(c). The new distances are recalculated, and we find the following values:

$$D = \begin{bmatrix} 0 & & \\ 3{,}3 & 0 & \\ 7{,}3 & 4 & 0 \end{bmatrix}$$

We now group the first group (represented by its center, 3) and the second one (a single element, 6,3), and we find the groups shown in figure 5(d). At last, we joint the two remaining groups, achieving the final clustering, as shown in figure 5(e)

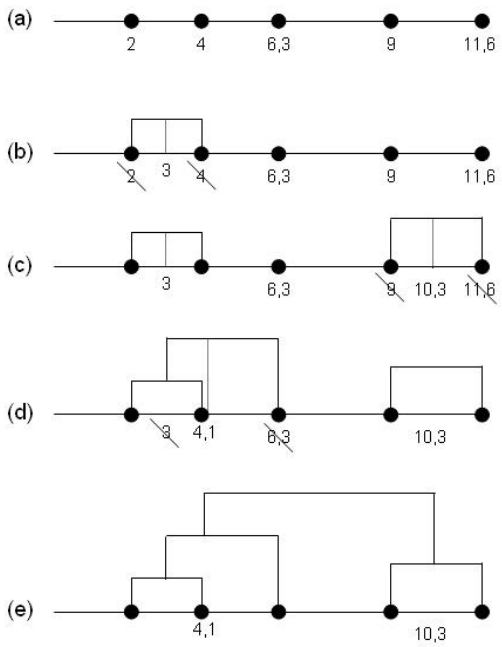

Figure 5. A step by step representation of our example execution of the hierarchical algorithm.

The hierarchical structure formed by the union of the groups was represented through a dendogram, which is the most common form of graphical representation for the results of hierarchical algorithms. Dendograms show intuitively the grouping order – the higher the line connecting two clusters, later was their grouping performed. Therefore, the height of the line connecting two clusters is somewhat proportional to their distance.

As we can also see in figure 6, there are three different ways to measure the distance between two clusters, which are the following:

- Single Link: The distance between two clusters is given by the distance between their closest points. It is also called neighbour clustering and is a greedy method that prioritizes elements that are closer, ignoring those further away.
- Average Link: The distance between cluster is given by the distance between their centroids. The only problem with this measurement is the necessity to recalculate centroids every iteraction.
- Complete Link: The distance between clusters is given by the distance between their points further apart.

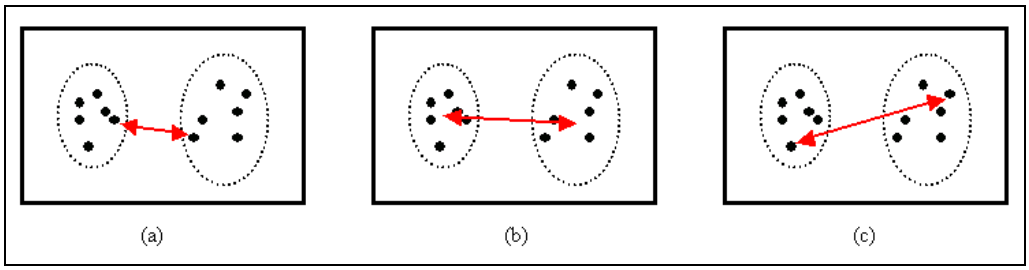

Figure 6. Distance measurements between two clusters that can be used by hierarchical algorithms. (a) single link, where the distance is given by the points closer together. (b) average link, where the distance is given by the distance between the centroids. (c) complete link, where the distance is given by the distance between the elements further apart.

These three methods are not completely equivalent. Different choices among them can generate different groupings, as we can see in figure 7.

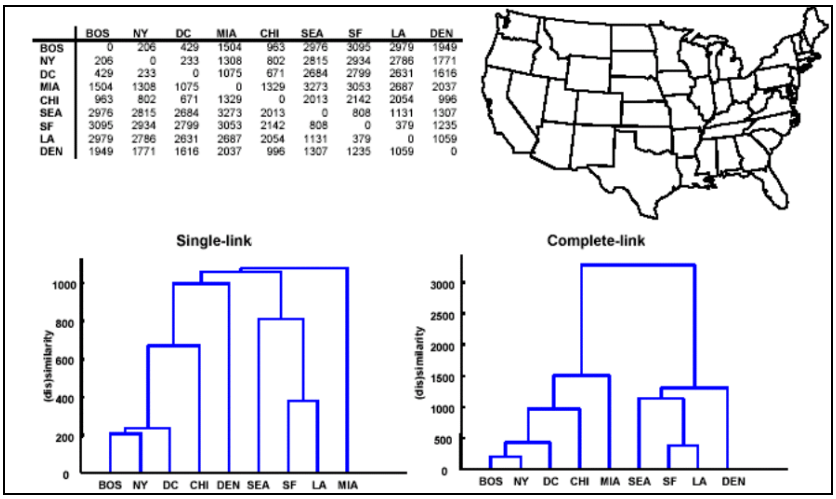

Figure 7. As example of the application of a hierarchical algorithm with two different ways of measuring distance. In the left, the grouping was calculated using single link and in the right, using complete link. Notice that the last city to be added in single link is Miami, who is quickly clustered in complete link (Example taken from the site http://www.analytictech.com/networks/hiclus.htm, last access in August/2011) .

The problem with this algorithm is that it results only in a hierarchy of relationships, not in specific groups. In order to obtain *k* groups, we need to cut the *k-1* highest edges of the dendogram. An example of this process can be seen in figure 8.

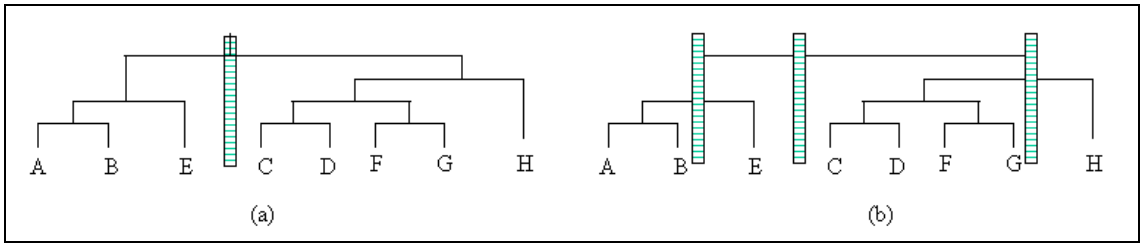

Figure 8. An example of dendogram cut in order to achieve the number of clusters needed. In (a) one cut was made in order to find two groups. In (b), three cuts were made and four groups were found: ({A,B},{E},{C,D,F,G},{H}).

There is a divisive version of the hierarchical clustering algorithm. This version starts with a single cluster constraining all elements. In each iteration all existing clusters are divided into two using a *flat* algorithm (such as K-Means) in order to simulate the breaking of an edge. The procedure iterates until we have sets composed of single elements or until a specified criterion (such as number of clusters) is matched.

As stated above, it is usual to break the set in two to make it look like it is the opposite of the agglomerative technique, but we are not forced to break any set into two halves. We can choose the number of sets using a metric that intend to optimize the cost of each cut, using two fundamental costs: intra-cluster similarity and extra-cluster similarity. The metric intends to maximize the relationship among those costs in order to generate the most cohesive clusters. Mathematical, we have the following:

Given

- $V = \{v_1, v_2, ..., v_n\}$, the set of all existing objects
- $\{C_1, C_2, ..., C_k\} \mid V = \bigcup_{i=1}^{k} C_i$, a partitioning of the objects;

We have that

- Intra-cluster similarity: $\text{int}ra_p = \sum_{v_i, v_j \in C_p} sim(v_i, v_j)$
- Extra-cluster similarity: $extra_p = \sum_{v_i \in C_p, v_j \notin C_p} sim(v_i, v_j)$

Then, the cost of partitioning in *k* sets is given by $\cos t_k = \sum_{p=1,2,...k} \frac{\text{int}ra_p}{extra_p}$, and we choose the partitioning that has the smallest cost among all possibilities.

This technique is extremely costly in computational terms because it demands a lot of partitioning in each iteraction, what may take a long time if the number of elements is huge. The cost may be limited if we limit the value o k, but the algorithm may still take an unacceptable long time.

The divisive version is a lot more complex than the agglomerative version, because it depends on a flat algorithm such as K-Means to work. Nevertheless, it has two main advantages:

1. It is not necessary to recalculate the distances between all clusters each step of the algorithm and we can stop the process before we get to the last level of the tree. Therefore, the algorithm is potentially faster than the agglomerative version.

2. We start by using the information on the whole data set instead of making decisions based only on local information as in the agglomerative algorithm. Hence, the clusters we achieve tend to be closer to the real information distribution in the data set.

An interesting fact is that this is one of the possible ways to determine how many clusters we will use for the K-Means technique. Instead of defining this number a priori, we can use the following algorithm:

```
Define a data set with all the elements and put it in a queue of groups to process;
While there are sets in the queue do:
        Take the first set from the queue;
        Calculate information loss related to this set;
        If information loss value is bigger that a predefined value
            Perform K-Means with k=2;
            Put two groups achieved in the queue;
        End If
End While
```

It may seem that we have replace an arbitrary value for another (after all, at first glance it may seem that we exchanged the value of k for the value of information loss). Nevertheless, we can see that instead of defining an arbitrary number of sets, we now define the quality of each set. This quality can be defined according to the problem we are trying to solve. That means that our problem can impose an inherent quality to our sets and we will search for it.

Another way to execute a divisive hierarchical algorithm is to start with the *minimal spanning tree*. This can be defined as the tree that connects all elements in the set in such a way that the sum of their distances is minimal. Once you have this tree (whose determination is performed with a fast algorithm), you can divide it in two set in one of the following two ways:

- Removing the biggest edge;
- Removing the edge of length l such that $l >> l'$, where l' represents the average size of all edges in the cluster. This method's advantages include generating denser clusters and early removal of outliers (because dense clusters must have all edges with a relatively small length).

Using one of those techniques we will find two clusters for the cluster under consideration. We repeat the process until the desired number of clusters is found.

In the past, hierarchical clustering algorithms were extremely popular, but with the development of flat techniques, the latter have become more accepted. Therefore, the decision to use a hierarchical algorithm is complex and must be made taking under consideration a few important issues.

The first issue is memory usage. In order to use a hierarchical algorithm we need to store $O(n^2)$ distances. Even if memory price keeps falling fast, if we have 10.000 elements, we will need close to 10^8 distances. This value is within the grasp of a personal computer but is quite large and demands an efficient management.

Non hierarchical algorithms usually depend on certain values that are predefined arbitrarily by the researcher, such as the number of sets or the seeding of the clusters. All choices can cause negative impact on the result quality. Yet, hierarchical agglomerative algorithms are not dependent on such factors, are completely deterministic and do not rely on human defined parameters.

The main advantage of hierarchical algorithms is that they offer not only the clusters, but the whole structure of the relationship between instances and allow for easy determination of subsets within that data. Besides, they allow us to visualize in the dendogram the way all instances are connected and the true resemblance of two different points, whether or not they belong to the same cluster.

2.3 Self Organizing Maps

A system is a group of parts that interact and work as a whole, having collective properties that are different from the properties of each of their parts. Systems are organized in order to have a specific function. Organization can be of two types:

- external, when imposed by external factors, such as machines, walls, etc.
- Self-organized, when the system evolves to an organized form by itself. For example: crystals, brain, economies, etc.

The brain is clearly an organized system that learns spontaneously without a professor. A few hunderd years ago there was the theory that there were little men inside the brain guiding the learning. Nowadays, the most accepted theory is the Hebb learning rule which states that when a neuron A repeatedly and persistently helps firing neuton B the efficiency of the association between those two cells increases (HEBB, 1949).

Those chances occur in the brain in three possible ways: increasing the number of transmitters emitted by A, increasing the strength of the synaptic link or by forming new synapses. This allows us to come to the conclusion that the brain is a self-organized system that is able to learn by modifying the connections between neurons.

The main result of the brain self-organization is the formation of characteristics maps of linear or planar topology in the brain. Two examples are the tonotopic map (sound frequencies are mapped spatially in the cortex in a progression from the lowest to the highest) and the retinotopic map (the visual field is mapped in the visual cortex with higher resolution to the center).

There is a type of neural network that implements the conccepts of cerebral self-organization. These networks are known as Self-Organizing Feature Maps (SOFMs). Other common names for them are competitive filter association memories or Kohonen networks, in homage to their creator, Dr. Eng. Teuvo Kohonen.

A Kohonen network has two layes (an input and a competitive one) as we can see in figure 9). Each neuron in the input layer represents a dimension in the input pattern and distributes its vector component to the competitive layer.

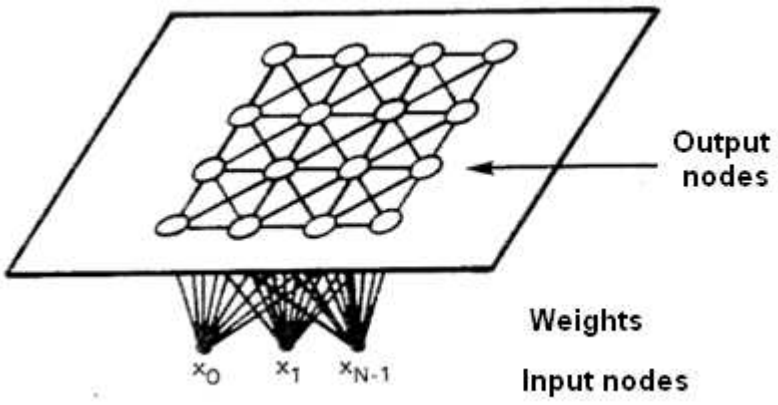

Figure 9. Example of the topology of a Kohonen network. The input layer neurons are connected to all output (competitive) layer neurons through weights. Some neurons of the output layers are connected to other neurons in the same layer, and those neuron to which it is connected form its neighborhood (BRAGA, 2000).

Each neuron in the competitive layer receives the weighted sum of the inputs and has a neighborhood of k neurons. This neighborhood can be arranged in one, two or n dimensions. When a neuron receives an input, some neurons will be excited enough to fire and each firing neuron may have an excitatory or inhibitory effect on its neighborhood.

After initializing the synaptic weights, three basic processes occur:

- Competition: The highest function value (the winner) is selected. The firing neuron is given by : $i(x) = \arg\min \|x-w_j\|$, $j = 1,2,3,..,l$. The winner determines the location of the center of the neighborhood to be trained.
- Cooperation: The neighbors of the winning neuron are selected and excited through a neighborhood function.
- Synaptic adaptatation: The excited neurons adjust their synaptic weights when a neuron ins a competition. The winner is not the only one to adapt: all nodes located in its neighborhood are also adjusted, according to a function shown in figure 10.

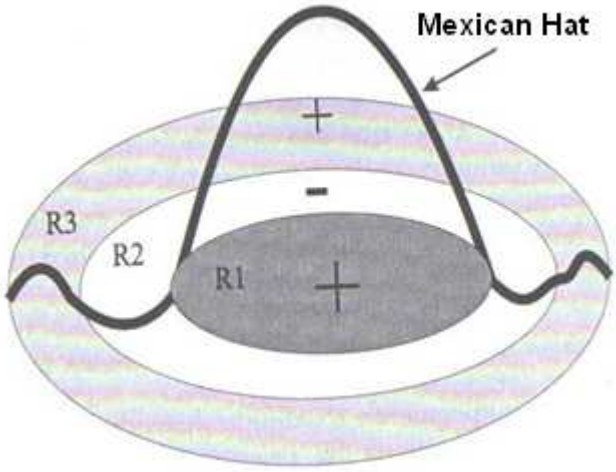

Figure 10. Output of the neighbors of the winning neuron, for the synaptic adaptation process. Notice that the elements that are closer to the winning neuron are excited while those farther away are inhibited. Elements that are very far away receive a little positive adjustment in order to avoid that outlier neurons are never adjusted (BRAGA, 2000).

The algorithm is given by the following steps:

Start with small random weights
Repeat until weights stabilize or until the number of iterations exceeds τ
Normalize weights
Apply an input I^u to the network
Find the winning neuron, i^*
Adjust the weights of the winner and its neighborhood according to the following formula:

$$\Delta w_j = \eta O(I_j^u - w_j),$$

O is the neuron's output which is calculated according to the following criteria:

- The output of i* is 1,
- The output of its neighborhood is given by the mexican hat function given in figure 10),

j is the coordinate index, j=1,2,...,p.

Kohonen networks are excellent for classification problems. Their problems is that the system behaves like a black box, without any assurance of convergence for higher dimension networks and the training process can be too long for large classification problems. On the other hand, their advantage is that they not only find the sets but also show a natural ordering between them, given the notion of neighborhood among neurons (OJA *et al*, 2002), and this notion can be very useful for several applications.

2.4 Graph based methods

A graph is a mathematical model that represents relationships among objects. A graph is a set given by *G=(V, E)*, where *V* is a finite set of points, usually called nodes or vertices and *E* is a relationship among nodes, that is a set of pairs *V*x*V*. In figure 11 we can see some example of graphs. In example (a) we have a graph with 5 nodes, having the set of nodes V={1,2,3,4,5}. These nodes are connected among themselves for some edges, defining the set E={ (1,1), (1,2), (3,4), (3,5), (4,5) }. A graph can be represented by an adjacencies matrix *A*, where A_{ij}=0 if the edge (i,j) does not exist and 1, in case it exists, as we can also see in figure 11.

The number of edges that leave or enter a certain node is called its degree or connectivity. For instance, in figure 11 (a), vertex 4 has connectivity 2, while in figure 11 (b), vertex 4 has a degree equal to 3.

Edges can receive a value or a label, and in this case the graph is called labeled. Labels can represent the connection strength between two elements. The graphs in figure 11(a) and 11(b) are non labeled, while the graph in 11(c) is label and, for instance, the edge between nodes 3 and 4 has a label equal to 8.

Edges in a graph can have a direction and in this case the graph is called directed. In order to define the direction, we draw the edges with an arrow point in the destination edge. In the case of a non directed graph, edges (e_1,e_2) are (e_2,e_1) identical, what does not happen in a directed graph.

In figure 11(b) we can see a directed graph. As we can see, there are two edges connecting nodes 1 and 4 – one for each direction. If besides being directed the graph is also labeled, then it is called a network.

A graph is called connected if there is a path between any pair of nodes in the graph, where a path is a sequence of edges { (e_1,e_2), (e_2,e_3), ..., (e_{k-1},e_k)}, where e_1 is the origin and e_k is the destiny. The example is figure 11(c) is connected, but the examples (a) and (b) are not. In the first case, it is not possible to find a path between nodes 1 and 4 and in the second there is no path leaving node 2 and arriving at node 1 (the existing edge is directed and leaves node 1 to arrive at node 2, and there is no way to make the inverse way).

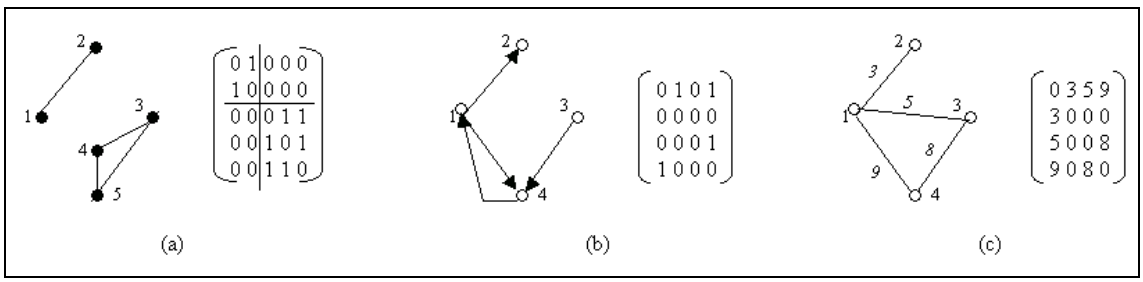

Figure 11. Graph examples. (a) A non directed, non connected and non labeled clique graph. (b) A directed, non connected non labeled graph. This graph is not a network because its edges are not labeled. (c) A non directed, connected labeled graph.

A graph is said to be complete if all pair of vertices are connected by an edge. A clique graph is a graph where each connected component is a complete graph. The example in figure 11(a) is a clique graph, because the graph can be split into two subgraphs G_1 e G_2, as we can see by the separation of the matrix into two submatrices (G_1 is represented by the upper left matrix and G_2 is representedn by the lower tight one). The other two matrices have all elements equal to zero, indicating that there is no connection between the subgraphs. The first subgraph contains nodes 1 and 2, while the second contains nodes 3, 4 and 5. Notice that each vertex in each subgraph is connected to all the other nodes in the same subgraph.

One way to group elements represented in a graph consists of assuming that the graph to be found will be a corrupted clique. That way, we just need to find the minimal number of edges to be added or removed in order to find a clique graph. The complete subgraphs found will represent the most adequate clusters for the data. An example of that technique can be seen in figure 12.

Unfortunately, this problem (called the corrupted clique problem) is NP-hard. This implies that there is not practical algorithm to solve it. Fortunately, there are heuristics to help us. We will discuss a few of them, but first let us understand how we can create the graph on which the heuristic will be applied.

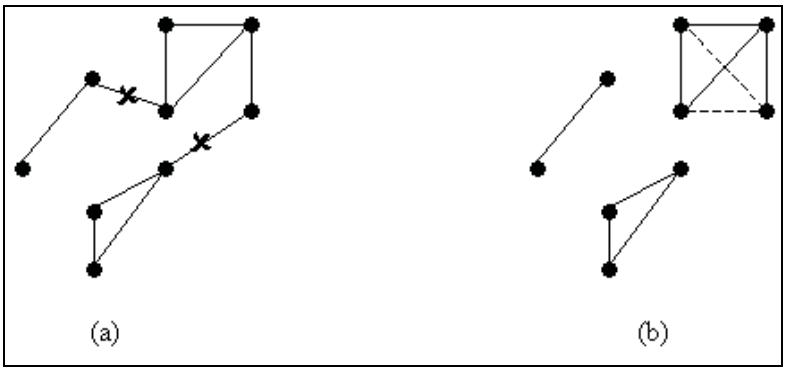

Figure 12. An example of how to find cliques minimizing graph change. (a) The original graph. The edges with a "X" will be erased. (b) The clique graph we found. The dotted edges were inserted. The minimal number of chances in order to find this clique graph was 4 (two inclusions and two deletions).

If the distances between two elements are calculated using one of the metrics defined in section 1.2, we can find a symmetric matrix D where each element D_{ij} represents the distance between elements i and j. We can then find an arbitrary limit θ so that we will define that there is an edge between nodes i and j if and only if the distance d_{ij} is smaller than θ (or bigger, if we are using correlation). An example of that technique can be seen in figure 13

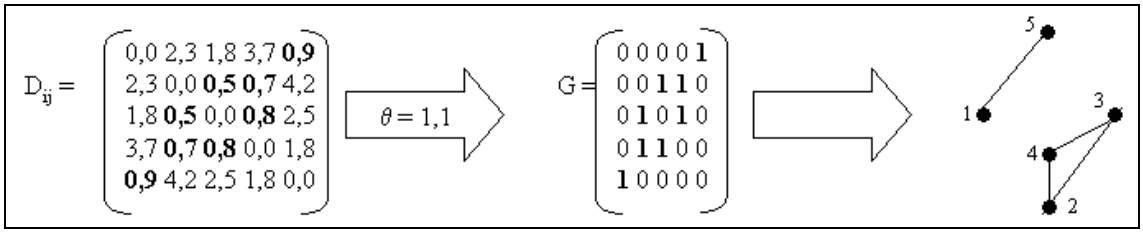

Figure 13. Example on the application of the technique to find a graph based on the calculated distances. To the existing distance matrix we applied a θ=1,1. All elements written in bold are below that threshold. We generate a graph representing matrix that contains entries equal to one in the positions written in bold. The represented graph can be seen in its graphic form in the right corner of the figure.

A practical and efficient algorithm to solve the corrupted clique problem is called CAST (Cluster Affinity Search Technique), defined in (BEN-DOR *et al*, 1999). The algorithm builds iteratively the partition P of a set S finding a cluster C so that no element $i \in C$ is distant from C and no element $i \notin C$ is close to C, given a graph formed by a distance threshold θ. The algorithm is the following:

```
S=set of vertices from graph G
P=∅
While S≠∅
        C={v}, where v = vertex with maximum degree in graph G
        Add to C all nodes connected to an element in C
        P=P+C
        S=S/C
        Remove all vertices from cluster C from graph G
End While
```

There are several other algorithms based on graphs that can be used instead of CAST. Among them , we can quote PCC, also defined in (BEN-DOR *et al*, 1999), which is based on a definition on the entries of a stochastic error model and CLICK (SHARAN *et al*, 2000), which is also based on the definition of a statistical model that assigns a probabilistic meaning to each graph edge. Both models have been widely used and are a lot more complex than CAST. Nevertheless, they must be considered by those wishing to expand their knowledge on graph based clustering techniques.

3. Conclusion

In this tutorial we have seen the main concepts on clustering and several heuristics applied to this problem that can be used in different application fields.

When there is no prior information, there is no "optimal" classification for the data set. In this situation, clustering methods serve as an important tool to explore data that, if used adequately, can partition the data without the application of any arbitrary notion or preconception. Given that this problem is NP-hard, it is necessary to find an heuristic to solve it.

The large number of existing clustering techniques may be confusing to the beginner. Sometimes, instead of choosing, it may be ideal to use all technique simultaneously. There are research works, such as (HU *et al*, 2004) that use several clustering techniques at one and combine their results in order to make a final decision.

It must be stressed that when dealing with data coming from a specific area of knowledge, one must not discard any prior knowledge of this are. All data and results found must be analyzed under the light of existing knowledge in order to validate them. No algorithm must be ever used in isolation, without thinking on the conclusions and ramifications of its application.

All these questions are important, not only to clustering techniques, but also to the algorithm you will apply to any application area of interest. Clustering techniques will often serve as input to other algorithms or as a preparation step for your experimental work.

4. References

BEN-DOR, A., SHAMIR, R., YAKHINI, Z. (1999), "*Clustering Gene Expression Patterns*", Journal of Computational Biology, v. 6, n. 3, pp. 281-297.

BRAGA, A. P, CARVALHO, A. C. P. L. F., LUDERMIR, T. B., 2000, "Redes Neurais Artificiais – Teoria e Aplicações", LTC Editora , Brasil.

CHIANG I.W-Y.; LIANG G-S.; YAHALOM S.Z (2003), "The fuzzy clustering method: Applications in the air transport market in Taiwan", The Journal of Database Marketing & Customer Strategy Management, v 11, n 2, pp. 149-158

DZWINEL, W., YUEN, D. A., BORYCZKO, K., *et al.* (2005), "Nonlinear multidimensional scaling and visualization of earthquake clusters over space, time and feature space", Nonlinear Processes in Geophysics n. 12 pp. 117–128

GORDON, A. D, 1981, Classification, Chapman and Hall Ed., 1981

HAIR, J F JR., ANDERSON, R. E. *et al*, Multivariate Data Analysis, Prentice Hall, 1995

HAMMOUDA, K. M. (2002), "Web Mining: Identifying Document Structure for Web Document Clustering", Master Dissertation, Department of Systems Design Engineering, University of Waterloo, Canada

HAYKIN, S., Redes Neurais: Princípios e Práticas, Ed. Bookman, 2001

HEBB, D. O. (1949). "The organization of behavior: A neuropsychological theory" New York: Wiley.

HU, X., ILLHOY, Y. (2004), **"**Cluster ensemble and its applications in gene expression analysis**",**ACM International Conference Proceeding Series; Vol. 55 Proceedings of the second conference on Asia-Pacific bioinformatics – v. 29, pp. 297-302 , New Zealand.

JACKSON, C. R., DUGAS, S. L. (2003), "Phylogenetic analysis of bacterial and archaeal *arsC* gene sequences suggests an ancient, common origin for arsenate reductase", *BMC Evolutionary Biology* 2003, v. 3 n. 18

JIANG, D., ZHANG, A. (2002) "Cluster Analysis for Gene Expression Data: A Survey", University of New York.

JONES, N. C., PEZNER, P., "An Introduction to Bioinformatics Algorithms", MIT Press, USA, 2004

KIM, J., WARNOW, T. (1999), "Tutorial on Phylogenetic Tree Estimation", Yale.University.

LEVIA JR D. F., PAGE D. R. (2000), "The Use of Cluster Analysis in Distinguishing Farmland Prone to Residential Development: A Case Study of Sterling, Massachusetts", Environ Manage. V. 25 n. 5, pp. 541-548

LINDEN, R.,2005, "Um Algoritmo Híbrido Para Extração De Conhecimento Em Bioinformática", PhD Thesis, COPPE-UFRJ, Rio de Janeiro, Brazil

MARTÍNEZ-DÍAZ, R. A., ESCARIO, J. E., NOGAL-RUIZ, J. J., *et al* (2001), "Relationship between BiologicaI Behaviour and Randomly Amplified Polymorphic DNA Profiles of *Trypanosoma cruzi* Strains", Mem. Inst. Oswaldo Cruz vol.96 no.2, pp. 251-256, Brazil

MATIOLI, S. R. (ed.), Biologia Molecular e Evolução, Holos Editora, 2001

OJA, M., NIKKILA, J., TORONEN, P., *et al*, 2002, "Exploratory clustering of gene expression profiles of mutated yeast strains" in *Computational and Statistical Approaches to Genomics, Zhang, W. e Shmulevich, I.* (eds), pp. 65-78, Kluwer Press

SHARAN, R., SHAMIR R 2000. "CLICK: A clustering algorithm with applications to gene expression analysis", Proc. Int. Conf. Intell. Syst. Mol. Biol. 8: 307–316.

YEOH, E., ROSS, M. E., SHURTLEFF, S. A. *et al.*, 2002, "Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling", Cancer Cell, v. 1, n. 1, pp. 133-143.