Contents

Chapter 1

Clustering

Abstract

Clustering is a widely used technique to partition data in homogeneous groups. It finds applications in many fields from information retrieval to bio-informatics. The main goal of clustering algorithms is to discover the hidden structure of data and group them without any a-priori knowledge of the data domain. Clustering is often used for exploratory tasks.

The intuition behind partitioning data is that if two objects are closely related and the former is also related to a third object, then more likely also the latter has a similar relation. This idea is known as the cluster hypothesis*.*

In this chapter we survey the principal strategies for clustering, the main clustering objective functions and related algorithms, the main definitions for similarity and the clustering validation techniques.

1.1 Introduction to clustering

Clustering is a technique to split a set of objects in groups such that *similar* objects are grouped together, while objects that are not similar fall in different clusters. The choice of the notion of similarity (or distance) among objects that are needed to be clustered is of crucial importance for the final result.

Clustering algorithms have no a-priori knowledge about the data domain, its hidden structure and also the number of hidden classes in which data are divided is unknown. For this characteristic, clustering is often referred as un-supervised learning in contrast to classification (or supervised learning) in which the number of classes is known and for each class a certain number of examples are given.

The independence of clustering algorithms from the data domain is at the same time the secret of its success and its main drawback. In fact since clustering does not need any a-priori knowledge of the data domain, it can be applied to a wide range of problems in different application areas. In contrast, general purpose procedures do not allow to apply (even trivial) problem dependent optimizations and consequently they typically perform worse then ad-hoc solutions.

1.1.1 Metric space for clustering

The choice of how to represent the data objects one wants to cluster, together with the choice of the clustering strategy, is critical for the clustering result. The representation schema depends from the type of data we are working on. In some fields de-facto standards are widely used.

In *text retrieval* the vector space model is the most commonly used. Documents in this model are represented as vectors of weighted terms called *bag of words*. For weighting, many approaches are used: binary schema (in which the weight of a term is 0 if the term does not appear in the document, 1 otherwise), the tf-idf scoring and so on. In *video retrieval* frames are represented as vectors in the HSV color space. In *bio-informatics*, DNA microarrays are matrices in which each gene is stored in a row and each column corresponds to a probe.

In all the above cited cases, a set of objects $O = \{o_1, \ldots, o_n\}$ are represented with m-dimensional vectors which are stored in a matrix M of n rows and m columns, where n is the number of objects in the corpus while m is the number of features of the objects. These vector spaces endowed with a distance function define a metric space. The most widely used distance functions are:

- **Cosine similarity:** it is defined as the cosine of the angle between o_a and o_b. More formally
$$s(o_a, o_b) = \frac{o_a \cdot o_b}{\|o_a\| \cdot \|o_b\|}$$
A distance can be easily derived from cosine similarity by:
$$d(o_a, o_b) = \sqrt{1 - s^2(o_a, o_b)}$$

The most important property of cosine similarity is that it does not depend on the length of the vectors: $s(o_a, o_b) = s(\alpha o_a, o_b)$ for $\alpha > 0$. This property makes the cosine similarity widely used in text information retrieval.

- **Jaccard coefficient:** in its basic form it is defined as:

$$J(o_a, o_b) = \frac{\#(o_a \cap o_b)}{\#(o_a \cup o_b)}$$

where $o_a \cap o_b$ is the set of features in common between o_a and o_b and $o_a \cup o_b$ is the total set of features (this approach assumes binary features). Many variants of the Jaccard coefficient were proposed in the literature. The most interesting is the *Generalized Jaccard Coefficient* (GJC) that takes into account also the weight of each term. It is defined as

$$GJC(o_a, o_b) = \frac{\min_{i=1}^{m}(o_{a,i}, o_{b,i})}{\max_{i=1}^{m}(o_{a,i}, o_{b,i})}$$

GJC is proven to be a metric [Charikar, 2002]. The Generalized Jaccard Coefficient defines a very flexible distance that works well with both text and video data.

- **Minkowski distance:** it is defined as:

$$L_p(o_a, o_b) = (\sum_{i=1}^{m} |o_{a,i} - o_{b,i}|^p)^{1/p}$$

It is the standard family of distances for geometrical problems. Varying the value of the parameter p, we obtain different well known distance functions. When $p = 1$ the Minkowski distance reduces to the Manhattan distance. For $p = 2$ we have the well known Euclidean distance

$$L_2(o_a, o_b) = \sqrt{\sum_{i=1}^{m}(o_{a,i} - o_{b,i})^2}$$

When $p = \infty$ this distance becomes the infinity norm defined as:

$$L_\infty(o_a, o_b) = \max_{i=1}^{m}(o_{a,i}, o_{b,i})$$

- **Pearson correlation:** it is defined as follows:

$$P(o_a, o_b) = \frac{\sum_{k=1}^{m}(o_{a,k} - \mu_a)(o_{b,k} - \mu_b)}{\sqrt{\sum_{k=1}^{m}(o_{a,k} - \mu_a)^2} \cdot \sqrt{\sum_{k=1}^{m}(o_{b,k} - \mu_b)^2}},$$

where μ_a and μ_b are the means of o_a and o_b, respectively. Pearson coefficient is a measure of similarity. In particular it computes the similarity of the shapes between the two profiles of the vectors (it is not robust against

outliers - potentially leading to false positive, assigning high similarity to a pair of dissimilar patterns -, it is sensible to the shape but not to the magnitude). To compute a distance, we define $d_{o_a,o_b} = 1 - P(o_a, o_b)$. Since $-1 \leq P(o_a, b_b) \leq 1$, for all o_a, o_b, we have $0 \leq d_{o_a,o_b} \leq 2$. This distance is not a metric since both the triangular inequality and small self-distance ($d_{o_a,o_b} = 0$) do not hold. However, the square root of $1 - P(o_a, o_b)$ is proportional to the Euclidean distance between o_a and o_b [Clarkson, 2006], hence only the small self-distance condition fails for this variant, and metric space methods can be used.

1.2 Clustering strategy

Clustering algorithms can be classified according with many different characteristics. One of the most important is the strategy used by the algorithm to partition the space:

- **Partitional clustering**: given a set $O = \{o_1, \ldots, o_n\}$ of n data objects, the goal is to create a partition $C = \{c_1, \ldots, c_k\}$ such that:
  - $\forall i \in [1, k] \qquad c_i \neq \emptyset$
  - $\bigcup_{i=1}^{k} c_i = O$
  - $\forall i, j \in [1, k] \wedge i \neq j \qquad c_i \cap c_j = \emptyset$
- **Hierarchical clustering**: given a set $O = \{o_1, \ldots, o_n\}$ of n data objects, the goal is to build a tree-like structure (called *dendrogram*) $H = \{h_1, \ldots, h_q\}$ with $q \leq n$, such that: given two clusters $c_i \in h_m$ and $c_j \in h_l$ with h_l ancestor of h_m, one of the following conditions hold: $c_i \subset c_j$ or $c_i \cap c_j = \emptyset$ for all $i, j \neq i, m, l \in [1, q]$.

Partitional clustering is said **hard** if a data object is assigned uniquely to one cluster, **soft** or **fuzzy** when a data object belongs to each cluster with a degree of membership.

1.2.1 Partitional clustering

When the data representation and the distance function d have been chosen, partitional clustering reduces to a problem of minimization of a given target function. The most widely used are:

- K**-center** minimizes the maximum cluster radius

$$\min \max_{j} \max_{x \in c_j} d(x, C_j)$$

- K**-medians** minimizes the sum of all the point-center distances

$$\min \sum_j \sum_{x \in c_j} d(x, \mu_j)$$

- K**-means** minimizes the sum of squares of inter-cluster point-center distances

$$\min \sum_j \sum_{x \in c_j} (d(x, \mu_j))^2$$

where $C = \{c_1, \ldots, c_k\}$ are k clusters such that C_j is the center of the j-th cluster and μ_j is its centroid.

For all these functions it is known that finding the global minimum is NP-hard. Thus, heuristics are always employed to find a local minimum.

1.2.1.1 FPF algorithm for the k-center problem

As said in 1.2.1 one of the possible goal for partitional clustering is the minimization of the largest cluster diameter solving the k-center problem. More formally the problem is defined as:

Definition 1. ***The k-centers problem:*** Given a set O of points in a metric space endowed with a metric distance function d, and given a desired number k of resulting clusters, partition O into non-overlapping clusters $C_1, \ldots, C_k$ and determine their "centers" $c_1, \ldots, c_k \in O$ so that $\max_j \max_{x \in C_j} d(x, c_j)$ (i.e. the radius of the widest cluster) is minimized.

In [Feder and Greene, 1988] it was shown that the k-center problem is NP-hard unless $P = NP$. In [Gonzalez, 1985; Hochbaum and Shmoys, 1985] two-approximated algorithms are given.

The FPF algorithm Given a set O of n points, FPF increasingly computes the set of centers $c_1 \subset \ldots \subset c_k \subseteq O$, where C_k is the solution to the k-center problem and $C_1 = \{c_1\}$ is the starting set, built by randomly choosing c_1 in O. At a generic iteration $1 < i \leq k$, the algorithm knows the set of centers C_{i-1} (computed at the previous iteration) and a mapping μ that associates, to each point $p \in O$, its closest center $\mu(p) \in C_{i-1}$. Iteration i consists of the following two steps:

1. Find the point $p \in O$ for which the distance to its closest center, $d(p, \mu(p))$, is maximum; make p a new center c_i and let $C_i = C_{i-1} \cup \{c_i\}$.
2. Compute the distance of c_i to all points in $O \setminus C_{i-1}$ to update the mapping μ of points to their closest center.

After k iterations, the set of centers $C_k = \{c_1, \ldots, c_k\}$ and the mapping μ define the clustering. Cluster $\mathcal{C}_i$ is the set of points $\{p \in O \setminus C_k$ such that $\mu(p) = c_i\}$, for $i \in [1, k]$. Each iteration can be done in time $O(n)$, hence the overall cost of the algorithm is $O(kn)$. Experiments have shown that the random choice of c_1 to initialize C_1 does not affect neither the effectiveness nor the efficiency of the algorithm.

FPF:
Data: Let O be the input set, k the number of clusters
Result: $\mathcal{C}$, k-partition of O
$C = x$ such that x is an arbitrary element of O;
for $i = 0$; $i < k$; $i++$ **do**
  Pick the element x of $O \setminus C$ furthest from the closest element in C;
  $C_i = \mathcal{C}_i = x$;
end
forall $x \in O \setminus C$ **do**
  Let i such that $d(c_i, x) < d(c_j, x), \forall j \neq i$ $\mathcal{C}_i$.append (x);
end
Algorithm 1: The furthest point first algorithm for the k-center problem.

1.2.1.2 K-means

The k-means algorithm [Lloyd, 1957] is probably the most widely used in the literature. Its success comes from the fact it is simple to implement, enough fast for relatively small datasets and it achieves a good quality. The k-means algorithm can be seen as an iterative cluster quality booster.

It takes as input a rough k-clustering (or, more precisely, k candidate centroids) and produces as output another k-clustering, hopefully of better quality.

K-means, as objective function, attempts to minimize the sum of the squares of the inter-cluster point-to-center distances. More precisely, this corresponds to partition, at every iteration, the input points into non-overlapping clusters $C_1, \ldots, C_k$ and determining their centroids $\mu_1, \ldots, \mu_k$ so that

$$\sum_{j=1}^{k} \sum_{x \in C_j} (d(x, \mu_j))^2$$

is minimized.

It has been shown [Selim and Ismail, 1984] that by using the sum of squared Euclidean distances as objective function, the procedure converges to a local minimum for the objective function within a finite number of iterations.

The main building blocks of k-means are:

- **the generation of the initial k candidate centroids**: In this phase an initial choice of candidate centroids must be done. This choice in critic because

both the final clustering quality and the number of iterations needed to converge are strongly related to this choice. In the next section we will survey the most important initialization strategies. A more complete survey and comparison can be found in [Bradley and Fayyad, 1998; Peña *et al.*, 1999].

- **the main iteration loop**: In the main iteration loop, given a set of k centroids, each input point is associated to its closest centroid, and the collection of points associated to a centroid is considered as a cluster. For each cluster, a new centroid that is a (weighted) linear combination of the points belonging to the cluster is recomputed, and a new iteration starts[1].
- **the termination condition**: Several termination conditions are possible; e.g. the loop can be terminated after a predetermined number of iterations, or when the variation that the centroids have undergone in the last iteration is below a predetermined threshold.

The use of k-means has the advantage that the clustering quality is steadily enough good in different settings and with different data. This makes k-means the most used clustering algorithm. Due its importance, there is a vast literature that discusses its shortcomings and possible improvements to the basic framework.

A lot of efforts was spent to reduce the k-means computational time that depends on the size of the dataset, the number of desired clusters and the number of iterations to reach convergence. Some methods attempt to use clever data structures to cache distances [Elkan, 2003; Smellie, 2004], others exploit the triangular inequality for avoiding distance computations [Phillips, 2002]. For small datasets or when only few iterations are enough to achieve the desired output quality, the performance of k-means is acceptable, but for nowadays needs clustering time has become a shortcoming.

Another well-known shortcoming is that some clusters may become empty during the computation. To overcome this problem, the "ISODATA" [Tou and Gonzalez, 1977] technique was proposed. Essentially when a cluster becomes empty, ISODATA splits one of the "largest" clusters so as to keep the number of clusters unchanged.

Initialize k-means Essentially k-means accepts as input an initial clustering that can be made with any clustering algorithm. It is well-known that the quality of the initialization (i.e. the choice of the initial k centroids) has a deep impact on the resulting accuracy. Several methods for initializing k-means are compared in [Bradley and Fayyad, 1998; Peña *et al.*, 1999]. The three most common initializations are:

RC The simplest (and widely used) initialization for k-means is the one in which the initial centroids are Randomly Chosen among the input points and the

[1] Note that k-means is defined on vector spaces but not in general on metric spaces, since in metric spaces linear combinations of points are not points themselves.

remaining points are assigned to the closest centroid. The resulting clustering is often referred as *random clustering*.

RP In the Random Perturbation, for each dimension d_j of the space, the distribution of the projections on d_j of the data points is computed, along with its mean μ_j and its standard deviation σ_j; the k initial centroids are obtained through k perturbations, driven by the μ_j's and σ_j's, of the centroid of all data points [Peña *et al.*, 1999].

MQ MacQueen's [MacQueen, 1967] proposed a variant of k-means: the initial centroids are randomly chosen among the input points, and the remaining points are assigned one at a time to the nearest centroid, and each such assignment causes the immediate recomputation of the centroid involved. Then k-means is initialized with the resulting clustering. Since it was experimentally shown that this initialization achieves generally a good quality in considerably less time than k-means, this initialization is often used in place of the standard k-means and it is often referred as *one-pass* k-means.

1.2.1.3 PAM: partition around medoids

Partition around medoids [Kaufman and Rousseeuw, 1990] was introduced by Kaufman and Rousseeuw. PAM introduces the concept of *medoid*. A medoid is a point of the input, it means that PAM is particularly suitable in all those cases in which the concept of centroid in not well defined. Moreover, in many cases, the more the number of objects increase, the less centroids tend to be representative; instead medoids are not affected by this problem.

PAM builds a k-clustering and it can be described as follows [Ng and Han, 1994]:

1. Select a set of k random input objects $O = \{o_1, \ldots o_k\}$,
2. for each input object $x \notin O$ compute the cost function $TC(x, o_i)$,
3. select the pair of objects x and o_i that minimize TC,
4. if $TC(x, o_i) < 0$ replace o_i with x and restart from step 2.

The final clustering is obtained using the objects o_i as cluster centers and assigning the input points to the cluster with the nearest center.

PAM is computationally expensive, in fact there are $(n - k)$ different pairs of object for each of th k medoids. It means that for each iteration TC is computed $k(n - k)$ times. Due to its computational cost, many variations and performance improvements were proposed in the literature [Zhang and Couloigner, 2005].

1.2.1.4 SOM: self organizing maps

Self organizing Maps [Kohonen, 2001] were introduced by Teuvo Kohonen as subtype of artificial neural networks used to produce low dimensional representation of the training samples while preserving the topological properties of the input space.

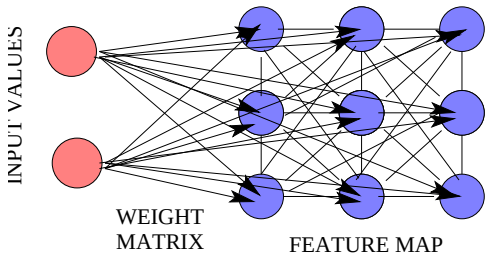

Figure 1.1. A simple 3×3 self organizing map.

A self-organizing map is a single layer feed-forward network, that is a network without direct cyclic paths. Neurons are arranged in a low dimensional grid (typically two-dimensional or tridimensional). Each neuron has associated a vector of weights $w_i = \{w_{i,1}, \ldots, w_{i,m}\}$ of the same size of input vectors. There are two main ways to initialize the weights vectors:

- using small random values,
- using a random perturbation from the subspace spanned by the two largest principal component eigenvectors. This initialization was shown to speed up the training phase of the SOM because they are already a good approximation of the SOM weights.

Self-organizing maps work in two phases:

- **training**: the training phase can be seen as the process in which the self-organizing map attempts to adapt the weight vectors of its nodes to the training data. For this purpose a large number of examples must be fed in input. If a training set is not available the input data are often used to train the network. The training algorithm is based on a *competitive learning* approach: when a new sample $x(t)$ is presented to the network it is compared with all the weights vectors and the neuron with closest weight vector (called Best Matching Unit) is selected (i. e. the neuron i such that $\min_i d(x(t), w_i)$). The weight vector of the BMU and its neighbors, are modified according with the sample. More formally let i the BMU and e a generic neuron of the SOM. Let $h(e, i)$ be a proximity function between the two neurons and $w_e(t)$ be the value of w_e at the epoch t. The weight vector of the generic neuron e is updated according with the following:

$$w_e(t+1) = w_e(t) + \alpha(t) * h(e,i) * (x(t) - w_e(t))$$

where $\alpha(t)$ is a monotonically decreasing learning coefficient.

- **mapping**: in this phase the input vectors are simply assigned to the closest neuron. Borrowing the terminology of k-means the nodes of the network in this phase play the same role of centroids. It is interesting to note that the number of clusters in output depends on the number of neurons in the network. This means that the structure of the SOM drastically influences the clustering results.

Learning:
Data: the SOM $M = \{m_j \forall j \leq TOT_{Nodes}\}$, $\alpha(t)$, $h(-,-)$,
$X = \{x(t) \forall t \leq TOT_{Sample}\}$
Result: the trained SOM M
forall $m \in M$ **do**
  initialize (m);
end
for $t = 1$; $t \leq TOT_{Sample}$; $t++$ **do**
  $i = \arg\min_j d(x(t), m_j)$;
  forall $m_e \in M$ **do**
    $m_e(t+1) = m_e(t) + \alpha(t) * h(e,i) * (x(t) - m_e(t))$
  end
end
return M;

Mapping:
Data: the SOM $M = \{m_j \forall j \leq TOT_{Nodes}\}$, $X = \{x(t) \forall t \leq TOT_{Sample}\}$
Result: The clustering C
for $i = 1$; $t \leq TOT_{Nodes}$; $t++$ **do**
  $C_i = \emptyset$;
end
for $t = 1$; $t \leq TOT_{Sample}$; $t++$ **do**
  $i = \arg\min_j d(x(t), m_j)$;
  $C_i = C_i \cup x(t)$
end
return C;

Algorithm 2: The self-organizing map algorithm.

In the case in which the size of input vectors is higher than the number of nodes in the output grid, SOM becomes a powerful tool to make dimensionality reduction [Tan *et al.*, 2005] (Feature selection).

1.2.2 Hierarchical clustering

The main difference between partitional clustering and hierarchical clustering consists in the fact the latter does not limit only in grouping the data objects in a flat partition, but it also arranges the data into a tree like structure. This structure is known as *dendrogram*. Each data object is assigned to a leaf of the tree, while internal nodes represent groups of objects such that for each pair of elements in such group, their distance is within a certain threshold. The root of the dendrogram contains all the objects. A flat clustering can be easily obtained by cutting the dendrogram at a certain level.

An important characteristic of hierarchical clustering is that it requires the computation of the *proximity matrix* that is the squared matrix of the distances between all the pairs of points in the data set. This makes the time and space complexity of this family of algorithms at least quadratic in the number of data objects. In recent years, a lot of effort was done to improve the hierarchical clustering algorithms performances and make them suitable for large scale datasets. Typical example are: BIRCH [Zhang *et al.*, 1996] and CUTE [Guha *et al.*, 1998].

The two main strategies for hierarchical clustering are:

- **Divisive**: in this case the dendrogram is built from the root to the leafs. Initially all the n objects are in the same cluster. A series of split operations is made until all clusters contains just a single element. The splitting operation is made by computing all the distances between the pairs of objects in the same cluster and selecting the two diametral points as seeds, then all the points in the group are assigned to the closest seed.

- **Agglomerative**: the dendrogram is built from the leaves to the root. At the beginning each object is inserted in a cluster (that represent a leaf of the dendrogram), than a series of merge operations is made until all the points belong to the same cluster. Since the data objects are n and each merge operation reduces the number of objects of one unit, $n - 1$ merge operations are needed. It is important to note that the operations of merge are made between the two closest entities (either objects or clusters). A notion of cluster-cluster distance and cluster-object distance must to be defined.

1.2.2.1 Divisive clustering

As mentioned in section 1.2.2, hierarchical divisive clustering algorithms start with considering the whole input set as a single cluster that is the root of the dendrogram. Before to start the procedure, a threshold distance must be chosen. Once this is done, hierarchical divisive clustering proceeds as follows:

- the proximity matrix M is calculated and for each cluster and the furthest pair of objects is selected,

- if the cluster satisfies the algorithm splitting criterion, (i.e. the distance between the diametral pair is higher than a certain threshold) the cluster is divided into two clusters by using the pair selected in the previous step as seeds,
- when no more clusters must to be splitted, the algorithm stops.

One of the most important issues in divisive hierarchical clustering is the choice of the splitting criterion [Savaresi *et al.*, 2002]. The following strategies are typically used [Karypis *et al.*, 1999]:

- each cluster is recursively splitted until each subcluster contains exactly one element. In this case a complete tree is obtained. The main advantage of this method is that a complete tree is obtained. The main disadvantage is that the final clustering quality is not taken into account by this schema.
- The cluster with the largest number of elements is splitted. Using this approach a balanced tree is obtained.
- The cluster with the highest variance with respect to its "centroid" is splitted. This is a widely used method to choose the cluster to split because it is related to the distribution of the elements inside the cluster.

1.2.2.2 Agglomerative clustering

As mentioned in section 1.2.2, hierarchical agglomerative clustering attempts to cluster a set of n objects providing also a tree like structure built from the leafs to the root.

In the merging operation the two closest entities of the dendrogram (leafs or internal nodes) are joined into a single entity. Considering leafs as clusters containing only an element, the notion of inter-cluster distance must be defined. There are many different possibilities for this choice. The most common ones are based on a linkage criterion (i. e. the distance between two clusters is the distance between two points that are associated to them in such a way). Given two clusters C_i and C_j we have:

- **Single linkage**: $d(C_i, C_j) = \min_{p \in C_i q \in C_j} d(p, q)$ is the distance between the closest pair of objects from different clusters. This method has the drawback that it tends to force clusters together due to a single pair of close objects regardless of the positions of the other elements in the clusters. This is known as *chaining phenomenon*.

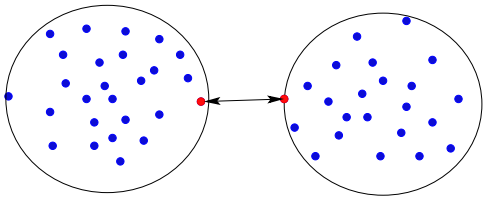

Figure 1.2. Single linkage criterion.

- **Complete linkage**: $d(C_i, C_j) = \max_{p \in C_i q \in C_j} d(p, q)$ is the distance between the farthest pair of objects from different clusters. This method tends to make more compact clusters, but it is not tolerant to noisy data.

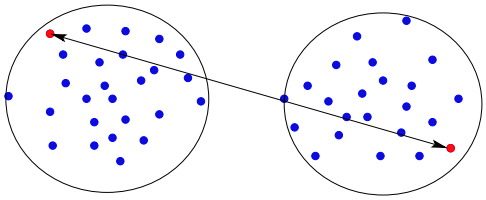

Figure 1.3. Complete linkage criterion.

- **Average linkage**: $d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{p \in C_i} \sum_{q \in C_j} d(p, q)$ is the mean of the distance among all the pairs of objects coming from different clusters. This method is more robust with respect to the previous ones, in fact the impact of outliers is minimized by the mean and the chaining phenomenon is typically not observed.

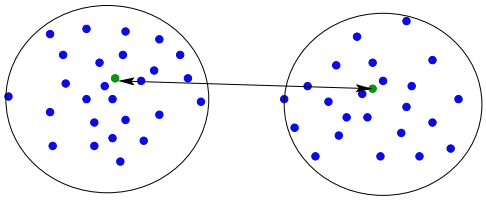

Figure 1.4. Average linkage criterion.

Single linkage and complete linkage can be generalized as suggested by Lance and Williams in [Lance and Williams, 1967] using the following formula:

$$d(C_l, (C_i, C_j)) = \alpha_i d(C_l, C_i) + \alpha_j d(C_l, C_j) + \beta d(C_i, C_j) + \\ + \gamma \mid d(C_l, C_i) - d(C_l, C_j) \mid \qquad (1.1)$$

where d is the distance between two entities, (C_i, C_j) is the cluster coming from the union of C_i and C_j and the four parameters α_i, α_j, β, γ, depend on the specific strategy used. Note that when $\alpha_i = \alpha_j = 1/2$, $\beta = 0$ and $\gamma = -1/2$, formula (1.1) becomes

$$d(C_l, (C_i, C_j)) = \min(d(C_l, C_i), d(C_l, C_j))$$

that is the single linkage formula. Instead, the choice of $\alpha_i = \alpha_j = \gamma = 1/2$ and $\beta = 0$ makes (1.1) be

$$d(C_l, (C_i, C_j)) = \max(d(C_l, C_i), d(C_l, C_j))$$

that is the formula of complete linkage.

The hierarchical agglomerative clustering algorithm can be summarized by the following procedure:

1. Initialize the proximity matrix M such that $M_{i,j}$ is the distance between the i-th and the j-th entity
2. Find i and j such that $i \neq j$ and $\forall h, k$: $h \neq k$, $M_{i,j} \leq M_{h,k}$
3. Join C_i and C_j and update M accordingly
4. Repeat from step 2 until all the clusters are merged

1.2.3 The choice of the number k of clusters

All the algorithms we considered in this chapter are not able to discover the number of groups in which the hidden structure of the input set should be divided. For all the described algorithms, the number of clusters is part of the input. In some cases, like SOMs, the choice of k is subjugated to the algorithm constraints. It is clear that the final clustering quality is strongly dependent from this choice. In fact, a too large number of clusters can have the effect to complicate the analysis of results, while too few clusters can lead to information loss or inaccurate modeling.

Many different techniques were proposed in the literature to find the "right" value for k; the most common approaches are based on: the construction of indices that take into account properties like homogeneity, separation and silhouette (a survey of some of them and an evaluation of their performances can be found in [Milligan and Cooper, 1985]); the optimization of some probabilistic functions and heuristics.

It is also important to note that all those methods, based on the computation of indices or on the optimization of probabilistic functions, must be applied to many choices of k. This makes desirable to have clustering algorithms able to make clusters incrementally without the need to know k in advance and to backtrack if needed. To this aim divisive hierarchical clustering and FPF are more flexible with respect to k-means and SOMs.

1.2.3.1 Stability based techniques

We describe here in more details the stability based technique based on the prediction strength method (developed by Tibshirani et al [Tibshirani *et al.*, 2005]) to estimate the number k of clusters. Then we describe an efficient variant of this schema applied to the FPF algorithm as we adopted in [Geraci *et al.*, 2007]. This approach can be used efficiently for all the incremental cluster algorithms such as the divisive hierarchical clustering.

To obtain the estimate of a good value of k, the method proceeds as follows. Given the set O of n objects, randomly choose a sample O_r of cardinality μ. Then, for increasing values of t ($t = 1, 2, \dots$) repeat the following steps:

1. using the clustering algorithm, cluster both $O_{ds} = O \setminus O_r$ and O_r into t clusters, obtaining the partitions $C_t(ds)$ and $C_t(r)$, respectively;

2. measure how well the t-clustering of O_r predicts co-memberships of mates in O_{ds} (i.e. count how many pairs of elements that are mates in $C_t(ds)$ are also mates according to the centers of $C_t(r)$).

Formally, the measure computed in step 2 is obtained as follows. Given t, clusterings $C_t(ds)$ and $C_t(r)$, and objects o_i and o_j belonging to O_{ds}, let $D[i, j] = 1$ if o_i and o_j are mates according to both $C_t(ds)$ and $C_t(r)$, otherwise $D[i, j] = 0$. Let $C_t(ds) = \{C_{t,1}(ds), \dots, C_{t,t}(ds)\}$, then the prediction strength $PS(t)$ of $C_t(ds)$ is defined as:

$$PS(t) = \min_{1 \le l \le t} \frac{1}{\#pairs \in C_{t,l}(ds)} \sum_{i,j \in C_{t,l}(ds), i<j} D[i, j] \qquad (1.2)$$

where the number of pairs in $C_{t,l}(ds)$ is given by its binomial coefficient over 2. In other words, $PS(t)$ is the minimum fraction of pairs, among all clusters in $C_t(ds)$, that are mates according to both clusterings, hence $PS(t)$ is a worst case measure. The above outlined procedure terminates at the largest value of t such that $PS(t)$ is above a given threshold, setting k equal to such t.

We now describe the modified version of the stability based technique we applied to FPF in [Geraci *et al.*, 2007]. Note that this modified procedure depends only on the ability of the clustering algorithm to create clusters one by one. We first run the clustering algorithm on O_r up to $t = \mu$, storing all the computed centers $c_1, \dots, c_\mu$. In a certain sense, the order in which centers are selected by FPF, is used as a sort of ranking of the points of O_r. In the case of using FPF this step costs $O(\mu|O_r|) = O(\mu^2)$.

We then cluster the input set O_{ds}. Suppose at step t we have computed the clusters $C_{t,1}(ds), \dots, C_{t,t}(ds)$ and suppose, for each $o \in O_{ds}$, we keep the index $i(o, t)$ of its closest center among $c_1, \dots, c_t$. Such index can be updated in constant time by comparing $d(o, c_{i(o,t-1)})$ with $d(o, c_t)$, i.e., the distance of o from the "current" center and that to the new center c_t. Now, for each $C_{t,l}(ds)$, $l \in [1, \dots, t]$ we

can easily count in time $O(|C_{t,l}(ds)|)$ the number of elements that are closest to the same center c_j, $j \in [1, \ldots, t]$, and finally compute the summations in formula 1.2 in time $O(|O_{ds}|)$.

After the last iteration, we obtain the clustering of O by simply associating the points $c_1, \ldots, c_\mu$ to their closest centers in $C_k(ds)$. The overall cost of the modified procedure using FPF as clustering algorithm is $O(\mu^2 + k(n-\mu) + k\mu) = O(kn)$ for $\mu = O(n^{1/2})$. Note that, differently from the original technique, we stop this procedure at the first value of t such that $PS(t) < PS(t-1)$ and set $k = t - 1$. In [Geraci *et al.*, 2007] we have empirically demonstrated that this choice of the termination condition gives good results.

1.3 Clustering validation

Since the clustering task has an ambiguous definition, the assessment of the quality of results is also not well defined. There are two main philosophies for evaluating the clustering quality:

- **internal criterion**: is based on the evaluation of how the output clustering approximates a certain objective function,
- **external criterion**: is based on the comparison between the output clustering and a predefined handmade classification of the data called *ground truth*.

When a ground truth is available, it is usually preferable to use an external criterion to assess the clustering effectiveness, because it deals with real data while an internal criterion measures how well founded the clustering is according with such mathematical definition.

1.3.1 Internal measures

There is a wide number of indexes used to measure the overall quality of a clustering. Some of them (i.e. the mean squared error) are also used as goal functions for the clustering algorithms.

1.3.1.1 Homogeneity and separation

According with the intuition, the more a cluster contains homogeneous objects the more it is a good cluster. Nevertheless the more two clusters are well separated the more they are considered good clusters. Following the intuition, homogeneity and separation [Shamir and Sharan, 2002] attempt to measure how compact and well distanciated clusters are among them.

More formally given a set of objects $O = \{o_1, \ldots, o_n\}$, we denote with $S(o_i, o_j)$ the similarity of the objects o_i and o_j according to a given similarity function. We say that o_i and o_j are mates if they belong to the same cluster. We define:

Homogeneity of a clustering: the average similarity between mates. Let $\mathcal{M}$ be the number of mate pairs:

$$H_{ave} = \frac{1}{\mathcal{M}} \sum_{o_i,o_j \ mates, i<j} S(o_i, o_j)$$

Separation of a clustering: the average similarity between non-mates. As $\mathcal{M}$ is the number of mate pairs, the number of non-mates pairs is given by $n(n-1)/2 - \mathcal{M}$.

$$S_{ave} = \frac{2}{n(n-1) - 2\mathcal{M}} \sum_{o_i,o_j \ non-mates, i<j} S(o_i, o_j)$$

Observe that the higher homogeneity is, the better the clustering is. Analogously, the lower separation is, the better the clustering is.

Alternative definition can be given using distances instead of similarities. In this case a better solution is given with a higher separation and a lower homogeneity.

Finally, homogeneity and separation can be approximated so that they can be calculated in linear time with the number n of objects (instead of quadratic). Given a clustering $C = \{C_1, \ldots, C_k\}$, let $cr(t)$ be the center (or centroid) of cluster C_t:

$$H_{approx} = \frac{1}{n} \sum_{t=1}^{k} \sum_{o_i \in C_t} S(o_i, cr(t)),$$

$$S_{approx} = \frac{1}{\sum_{t<z} |C_t||C_z|} \sum_{t<z} |C_t||C_z| S(cr(t), cr(z)).$$

Again, these measures can be expressed in terms of distances instead of similarities.

These two measures are inherently conflicting, because typically an improvement on one will correspond to a worsening of the other.

1.3.1.2 Average silhouette

Another measure that is worth calculate for a given clustering is the *average silhouette* [Rousseeuw, 1987]: for each element we compute a quantity, called silhouette, that gives an indication of how well the element fits into the cluster it is assigned to. The silhouette is based on homogeneity and separation; in particular we compute the homogeneity of the element with the elements in its cluster and the separation of the element with the closest cluster (among the others). In this way we can see if the element is well placed or if it is better placed in another cluster. The silhouette of object o_i that belongs to cluster $c \in \mathcal{C}$ is given by:

$$sil(o_i) = \frac{b_i - a_i}{\max\{a_i, b_i\}},$$

where a_i is the average distance of o_i to the elements in its cluster, while b_i is the average distance of o_i to the elements of the closest cluster. In formulas:

$$\begin{aligned} a_i &= \frac{1}{|c|} \sum_{o_j \in c} d(o_i, o_j) \\ b_i &= \min_{c' \in \mathcal{C}, c' \neq c} \left\{ \frac{1}{|c'|} \sum_{o_j \in c'} d(o_i, o_j) \right\}. \end{aligned}$$

(The values a_i and b_i can be approximated using the centers (or centroid) of clusters, in the same way as for homogeneity and separation).

Observe that for each element o_i we have $-1 < sil(o_i) < 1$ and that whenever o_i fits in its cluster, then $b_i > a_i$ and $sil(o_i) > 0$, while if o_i fits better in another cluster, then we have $b_i < a_i$ and $sil(o_i) < 0$.

To measure the quality of the whole clustering we use the *average silhouette*:

$$sil(\mathcal{C}) = \frac{1}{n} \sum_{i \in n} sil(o_i).$$

The higher this value is, the better the clustering is.

1. A singleton $\{o_i\}$ has silhouette equal to one because $a_i = 0$ and $b_i > 0$ (each element fits well in a cluster by its own).

2. If there is only one big cluster then for each $o_i \in n$ we have $sil(o_i) = -1$, because $b_i = 0$ and $a_i > 0$ (no element fits well in a cluster with all other elements).

The silhouette is not only used for assessing the clustering quality but can be helpful to guide the clustering task in many ways:

1. Given a cluster, the elements with lower silhouette might be excluded from the cluster to have more homogeneous clusters.

2. Given two clusterings of the same set of objects, done with the same clustering algorithm, but with different number of clusters, the one with higher average silhouette is preferable to the one with lower average silhouette. Thus, it can be used to decide k, the number of clusters in the clustering [Lamrous and Tailerb, 2006]. Experiments show that silhouette index is not very useful for this purpose.

1.3.2 External measures

In the following, we denote with $GT(S) = \{GT_1, \ldots, GT_k\}$ the *ground truth* partition formed by a collection of *classes*; and with $C = \{c_1, \ldots, c_k\}$ the outcome of the clustering algorithm that is a collection of *clusters*.

1.3.2.1 F-measure

The F-measure was introduced in [Larsen and Aone, 1999] and is based on the *precision* and *recall* that are concepts well known in the information retrieval literature [Kowalski, 1997], [Van Rijsbergen, 1979]. Given a cluster c_j and a class GT_i we have:

$$precision(GT_i, c_j) = \frac{|GT_i \cap c_j|}{|c_j|} \quad recall(GT_i, c_j) = \frac{|GT_i \cap c_j|}{|GT_i|},$$

Note that precision and recall are real numbers in the range $[0, 1]$. Intuitively precision measures the probability that an element of the class GT_i falls in the cluster c_j while recall is the probability that an element of the cluster c_j is also an element of the class GT_i. The F-measure $F(GT_i, c_j)$ of a cluster c_j and a class GT_i is the harmonic mean of precision and recall:

$$F(GT_i, c_j) = 2\frac{precision(iGT, c_j) recall(GT_i, c_j)}{precision(GT_i, c_j) + recall(GT_i, c_j)}$$

The F-measure of an entire clustering is computed by the following formula:

$$F = \sum_i \frac{|GT_i|}{n} \max_j (F(GT_i, c_j)),$$

where n is the sum of the cardinality of all the classes. The value of F is in the range $[0, 1]$ and a higher value indicates better quality.

1.3.2.2 Entropy

Entropy is a widely used measure in information theory. In a nutshell we can use the relative entropy to measure the amount of uncertainty that we have about the ground truth provided the available information is the computed clustering. Given a cluster c_j and a class GT_i, we can define

$$p_{i,j} = \frac{|GT_i \cap c_j|}{|GT_i|},$$

$$E_j = \sum_i p_{i,j} \log p_{i,j},$$

$$E = \sum_j \frac{|c_j|}{n} E_j,$$

where n is the number of elements of the whole clustering. The value of E is in the range $[0, \log n]$ and a lower value indicates better quality.

1.3.2.3 Accuracy

While the entropy of a clustering is an average of the entropy of single clusters, a notion of accuracy is obtained using simply the maximum operator:

$$A_j = \max_i p_{i,j}$$

$$A = \sum_j \frac{|c_j|}{n} A_j.$$

The accuracy A is in the range $[0, 1]$ and a higher value indicates better quality.

1.3.2.4 Normalized mutual information

The *normalized mutual information* (see e.g. [Strehl, 2002, page 110]), comes from information theory and is defined as follows:

$$NMI(C, GT) = \frac{2}{\log |C||GT|} \sum_{c \in C} \sum_{c' \in GT} P(c, c') \cdot \log \frac{P(c, c')}{P(c) \cdot P(c')}$$

where $P(c)$ represents the probability that a randomly selected object o_j belongs to c, and $P(c, c')$ represents the probability that a randomly selected object o_j belongs to both c and c'. The normalization, achieved by the $\frac{2}{\log |C||GT|}$ factor, is necessary in order to account for the fact that the cardinalities of C and GT are in general different [Cover and Thomas, 1991].

Higher values of NMI mean better clustering quality. NMI is designed for hard clustering.

1.3.2.5 Normalized complementary entropy

In order to evaluate soft clustering, the *normalized complementary entropy* [Strehl, 2002, page 108] is often used. Here we describe a version of normalized complementary entropy in which we have changed the normalization factor so as to take overlapping clusters into account. The entropy of a cluster $c_j \in C$ is

$$E_j = \sum_{k=1}^{|GT|} -\frac{|GT_k \cap c_j|}{|GT_k|} \log \frac{|GT_k \cap c_j|}{|GT_k|}$$

The normalized complementary entropy of c_j is

$$NCE(c_j, GT) = 1 - \frac{E_j}{\log |GT|}$$

NCE ranges in the interval $[0, 1]$, and a greater value implies better quality of c_j. The complementary normalized entropy of C is the weighted average of the contributions of the single clusters in C. Let $n' = \sum_{j\in 1}^{|C|} |c_j|$ be the sum of the cardinalities of the clusters of C. Note that when clusters may overlap it holds that $n' \geq n$. Thus

$$NCE(C, GT) = \sum_{j\in 1}^{|C|} \frac{|c_j|}{n'} NCE(c_j, GT)$$

Chapter 2

Text document clustering

Abstract

Dealing with text documents is one of the foremost issues in information retrieval. In this context, clustering plays a strategic role. Large text document corpora have become popular with the growth of the Internet and the decrease of price of disk storage space and connection band-width.

Dealing with text documents is a hard task. This is due to some intrinsic characteristics of human languages. For example, the same word can have different meanings according with the context in which it is referred. Moreover the prefix or suffix of a word can vary in different contexts. All the peculiarities of human languages motivate the effort of researchers in the field of text information retrieval.

In this chapter we survey the most important problems and techniques related to text information retrieval: document pre-processing and filtering, word sense disambiguation, vector space modeling, term weighing and distance functions.

2.1 Introduction

The term *Information retrieval* (IR) was firstly introduced by Calvin Mooers at the end of 40's. Then information retrieval has become a very broad field of computer science. It can be intuitively defined as the task of finding interesting and typically unstructured "objects" for the user information needs in a wide collection.

Conceptually, IR systems can be designed to manage almost everything (texts, images, videos), but the data type which has concentrated most of the studies is text. The growth of the Internet has stressed even more the interest in this sense. A lot of efforts have been spent to design general models for text information retrieval. Of course, the question about how similar are two documents has still not found a univocally accepted answer. There are many reasons that make it difficult deal with texts. First of all, the same concept can be expressed in many different ways by different writers. Moreover, the use of synonyms can drastically reduce the number of words shared by two related documents. On the contrary, the same word can assume very different meanings according with the context in which it is used. Nevertheless texts have the intrinsic characteristic that a not negligible part of words are due to grammar rules and do not provide additional information.

Despite text documents do not have a clear structure in general, this can not be considered always true. In fact, text documents are usually divided in sections, begin with a title and, in many standard document file formats, their structure is precisely marked using tags (HTML, XML and RTF just to give some examples). Another example is constituted by semi-structured texts that are documents with a poor structure (i.e. web snippets).

In this chapter we survey the major techniques designed to manage text documents and some considerations for applying clustering algorithms to these data.

2.2 Text representation

A text document is, in its most simplistic representation, a sequence of words. With the purpose of indexing it or computing its similarity with other documents (or equivalently with a text query), it must be preprocessed to remove the noise due to "syntactic sugar" and make it more treatable by computers. Preprocessing typically consists of many steps:

- **Text normalization**: in a document the same word can appear in different forms. For example the beginning of a sentence begins with capital letter. Naturally, these little variations do not affect the semantics of the term. To make it easier for IR systems to manege texts, terms must be normalized by converting them to lower case, remove dashes in multi-words terms, etc. Numbers, dates and punctuation are removed from texts. In the *bag of words* model, where the context of the words is not used, terms are often sorted lexicographically.

- **Stemming**: one of the major differences between human languages and programming languages is that, in the first case, words have a fixed root and a suffix (often also a prefix) that can vary depending on the context, while, in the last case, keywords are invariable. However, the large part of the words semantics is contained in their root, thus terms with the same root should be considered as the same term. The goal of stemming algorithms is to reduce a word to its root. To complicate this task there is the fact that the rules for extracting the root of a word depend from two aspects: the type of word (i. e. verbs, conjugations) and the language (i. e. English, Italian). Moreover human languages admit a lot of exceptions (i. e. the plural form of the word child is children instead of childs as suggested by the standard rule). Martin Porter [Porter, 1980] in 1979 introduced a rule based algorithm that has become the most famous stemmer algorithm for English still in use. Similar word stemmer algorithms are now available in almost all languages. The major perplexity in using stemming is that it can cause misunderstanding and change the meaning of the words. For example a too aggressive stemming can reduce the word “organization” to “organ” or “policy” to “police”. On the other hand, also a mild stemming can modify the original sense of a word.

- **Stop words removal**: to the contrary of computer languages, human languages are rich of words. Most of them have not a meaning by themselves but are used as grammar bricks to build complex sentences (for example articles or prepositions). All these words can be safely removed from the text without any loss of information. There are also words which have a very general meaning or are too popular to be really helpful in the understanding of the semantics of a text or its topic (i. e. above or below). Their removal cause only a marginal loss of information, but has the benefit of reducing the size of the representation of the text (in sections 2.2.1 we will explain the reasons that make this reduction a desirable property). The set of all the terms, removable with a negligible semantics loss, is called stop words list.

- **Vocabulary building**: the growth of the computational power, memory size and bandwidth was followed by an increase of the available textual information. Low hardware costs had the effect that, even small enterprises want to be able to process their internal knowledge base. Vocabularies do not change the models for storing and querying texts, they are only used to produce a more compact representation of texts with the goal of maintain much more documents in main memory. To each distinct term of the corpus an univocal identifier is assigned and it is stored in a table V called vocabulary. Thus, each document can be represented in a more compact form as a list of term identifiers with a consequent reduction of the required storage.

Before to discuss a model for dealing with texts, it is important to understand how words are distributed in documents and what are the implications of this dis-

tribution. Words in a corpus are not evenly distributed. A quite small set of words appear very frequently. Some of them like articles and prepositions are connected to grammar and can be removed as stop words, some others are instead topic dependent. A medium size set of words appear with intermediate frequency and a broad set of words appear rarely. This last set is made wider for example by words containing typos or by very specific words. This distribution, known as *power law*, was observed in all the human languages and is widely accepted as an intrinsic human characteristic. Clearly, words that appear with high frequency are useless because they do not exploit differences among documents (their presence in two documents does not means that those document are similar). Also rare words can be useless (the reason will be more clear in section 2.2.1).

2.2.1 The vector space model

With the term *model* in information retrieval we refer to a representation for documents and queries equipped with a notion of distance/similarity among them. The *vector space model* is a well known model, widely accepted and used, for organizing texts. After preprocessing, described in the previous section, a text document is reduced to a flat list of terms. Moreover, after preprocessing a vocabulary of all the terms in the document becomes available. Thus a document can be stored in a vector that has as many components as vocabulary words. Each component of the vector represents a score for the corresponding word (depending from the chosen weighting schema) or it is 0 if the word is not present in the document. All the documents of the corpus can be arranged in a matrix called *document matrix* such that rows correspond to documents and columns refer to terms.

Let $D = \{d_1, \ldots, d_n\}$ be a corpus of n documents such that V is the vocabulary of all the words in D. Thus D can be arranged in a matrix M such that $m_{i,j}$ corresponds to the term $v_i \in V$ in document $d_j \in D$. There are many possible different weighting schemes proposed in the literature. The most advanced IR systems weight terms according to their importance and characteristics (i.e. frequency in the document and in the corpus). The most used weighting schemas are:

- **Boolean (binary) model**: if v_i is present in d_j, then $m_{i,j} = 1$ otherwise $m_{i,j} = 0$.
- **Term frequency (TF)**: let $tf_{i,j}$ be the number of occurrences of term v_i in document d_j. In the term frequency model we have $m_{i,j} = tf_{i,j}$. It could be more convenient to normalize the weights to be independent from the document length. Let $WC_j = \sum_{i=1}^{|V|} tf_{i,j}$ be the total number of words in d_j, then:

  $$m_{i,j} = \frac{tf_{i,j}}{WC_j}$$

  According to this normalization, $m_{i,j}$ is in the range $[0, 1]$ and can be interpreted as the probability that word v_i appears in document d_j.

- **Term frequency - Inverse document frequency (TF-IDF)**: let w_i the number of documents in which v_i appears. We define $idf_i = \log n/w_i$. According to the TF-IDF schema $m_{i,j} = idf_i * tf_{i,j}$. To normalize the TF-IDF score in the range $[0, 1]$ the following formula is often preferred:

$$m_{i,j} = \frac{tf_{i,j} * idf_i}{\sqrt{\sum_{k=1}^{|V|}(tf_{k,j}^2 * idf_i^2}}.$$

  This scheme assigns high score to those words that appear frequently in a document, but are rare in the corpus. Instead, words that appear in a large portion of the document corpus are not too helpful to exploit differences among documents and thus are considered not important.

The vector space model has two main drawbacks. Since documents are vectors with $|V|$ components, even in the case of small corpora the dimensionality of the resulting vector space is usually high. Moreover, documents are very sparse vectors. A side effect of these two phenomena is that distances among documents tend to become high. In addition, the distance between a pair of similar documents is not so far from the distance between two unrelated ones. The standard technique to reduce vector space dimensionality and make document vectors more dense is the feature selection that will be discussed later in this section.

Another important issue is that the bag of words model does not considerate the context of words. Context is clearly important to extract the sense of a term because the same word could change its meaning in different contexts. This fact became more evident for multi word terms. For example phrasal verbs in English or people names. The problem of assigning the correct meaning to a word, called *word sense disambiguation*, is well studied and many techniques have been proposed in the literature, but they are typically more complex or computationally expensive or their performance depends from a knowledge base and thus are topic dependent. Moreover, none of them has at the moment exploited sufficiently higher performances with respect to the bag of words model to be considered as a valuable alternative.

2.2.1.1 Metric spaces and distance

A natural way to see documents (and queries) in the previously described model is thinking to them as vectors (or points) in a high dimensional Euclidean space. When a normalized weighting schema is applied, documents lay in the positive part of the surface of a iper-sphere of radius 1.

The most natural way to compute distances among documents is using the classical Euclidean distance. This distance is a special case of the *Minkowski distance* in which the parameter p is set to 2. Given two documents d_1 and d_2 and the related

rows of the matrix M, The Minkowski distance is defined as

$$L_p(d_1, d_2) = (\sum_{i=1}^{|V|} |m_{i,1} - m_{i,2}|^p)^{1/p}$$

.

Given two vectors the measure of the angle formed by them can be used as distance between the two represented documents. If the two documents contain exactly the same words, they give raise to the same vector and thus their angle is 0. In case the two documents do not have words in common they produce two orthogonal vectors and thus their distance is maximum. Based on this idea the *cosine similarity* is defined as the cosine of the angle formed by two vector documents. More formally let d_1, d_2 be two documents:

$$s(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\| \cdot \|d_2\|}$$

Note that denominator is used only to normalize the cosine similarity to be in the range $[0, 1]$ independently from the length of the involved vectors. This measure is a similarity score. Similarity is the dual concept with respect to distance. The more two objects are similar (similarity value is high) the more their distance tends to 0 and vice versa. Cosine similarity and all algorithms designed to employ similarity measures can be converted to use distances and vice versa. As noted in [Clarkson, 2006] the inner product of two vectors d_1 and d_2 of length 1 (in norm 2), that is the standard cosine similarity of two normalized vectors, is turned into a distance by $D(d_1, d_2) = 1 - s(d_1, d_2)$. This distance function is not a metric in a strict sense since the triangular inequality does not hold, however the following derivation $\|d_1 - d_2\|_2^2 = d_1 \cdot d_1 + d_2 \cdot d_2 - 2d_1 \cdot d_2 = 2(1 - d_1 \cdot d_2) = 2D(d_1, d_2)$ shows that the square root of the distance is indeed a metric. Equivalently one can say that it satisfies the extended triangular inequality $D(d_1, d_2)^\alpha + D(d_2, d_3)^\alpha \geq D(d_1, d_3)^\alpha$ with parameter $\alpha = 1/2$. Moreover a linear combination of distance functions with positive weights defined on the same space is still a metric space $D(d_1, d_2) = \sum_i w_i D_i(d_1, d_2)$ for $w_i \geq 0$. Thus cosine similarity although not giving rise to a metric in a strict sense is nonetheless closely related to a metric space.

Another commonly used coefficient to measure distance between pairs of documents is the *Jaccard coefficient*. In its original form this measure does not take into account weights and reduces a weighted scheme to a binary one. Let $d_1 \cap d_2$ be the set of terms that d_1 and d_2 have in common and $d_1 \cup d_2$ the set of terms present in at least one of the two documents. The Jaccard coefficient is defined as:

$$J(d_1, d_2) = \frac{\#(d_1 \cap d_2)}{\#(d_1 \cup d_2)}$$

Many variants of Jaccard coefficient were proposed in the literature. The most interesting is the *Generalized Jaccard Coefficient* that takes into account also the

weight of each term. It is defined as

$$GJC(d_1, d_2) = \frac{\sum_{i=1}^{|V|} \min(m_{i,1}, m_{i,2})}{\sum_{i=1}^{|V|} \max(m_{i,1}, m_{i,2})}$$

GJC is proven to be a metric [Charikar, 2002].

2.2.1.2 Word sense disambiguation

All these distance measures have the drawback that, in different ways, the more documents share terms, the more they are considered related. This is not always true for many reasons. Firstly, the same word can have different meanings in different contexts (*lexical ambiguity*), thus having a word in common does not necessarily imply similarity. Secondly, all human languages allow the use of synonyms to express the same concept with different words, therefore two documents can deal with the same topic sharing only few words. Moreover similar concepts can involve the use of complex semantic relationships among the words. For example, after removing stop words, the two sentences: “the apple is on the table” and “there is an orange on my desktop” have no words in common, but both say something about a “fruit on a board”, thus they are not completely unrelated. The above example shows two important notions of similarity:

- **paradigmatic, or substitutional, similarity** when two words may be mutually replaced in a particular context without change the semantics of the text (i. e. the words table and desktop in the previous example),
- **syntagmatic similarity** when two words significantly co-occur in the same context. (i. e apple, orange and fruit in the previous example).

To take into account these similarities among words many techniques have been proposed. The most common approaches are based on the attempt to generate (manually or automatically) an ontology of words. The advantage of ontologies is that they can be used to define a degree of similarity between couples of words, and thus to find relationships among them. In the previous example both the words “orange” and “apple” have as ancestor the term “fruit” and thus they are related. A lot of effort was done to design indexes to measure the degree of similarity between two words in the ontology graph. Many of them take into account the length of the path between two words. In [Agirre and Rigau, 1996] Agirre and Rigau propose the *conceptual density* that also takes into account the depth of the nodes in the hierarchy (deeper are closer) and the density of nodes in the subhierarchies involved (denser subhierarchies are closer)

The most important project for ontologies of words is *WordNet* [Miller, 1990]. Originally proposed by the Cognitive Science Laboratory at Princeton University only for the English language, WordNet has become a reference for all the information retrieval community and similar projects are now available in many other

languages. WordNet is a handmade semantic lexicon that groups words into sets of synonyms called *synsets*. Intuitively one can replace a word in a text with another from the same synset without changing its semantics. A word can appear in more than one synset if it has more than one meaning. Moreover synsets are arranged as nodes in a graph such that there is an edge to connect two nodes if there is a relation between the two synsets. There are different types of possible relations, an exhaustive list of them can be found in the WordNet web site [Miller *et al.*, 2006]. Given two synsets X and Y, the most common types of relations in WordNet are: *hypernym* if every X is a "kind of" Y, *hyponym* if Y is a "kind of" X, *holonym* if X is a part of Y and *meronym* if Y is a part of X. Thus, in our example WordNet has a link between orange and fruit and also between apple and fruit hence it is possible to infer a relation between orange and apple.

Clustering is often used also for grouping words into semantically homogeneous sets. This technique is known as *word clustering* [Dhillon *et al.*, 2002; Li and Abe, 1998]. In this case the set of objects to be clustered are not documents like in the previous case, but only words. Thus, the main issues in this context are the features associated to each word and the definition of the distance among words. In fact the distance should be designed in a manner to take into account all the considerations we made before. There are two main philosophies available in the literature. One is to define a distance over an ontology like [Agirre and Rigau, 1996] and thus the issue of how to create the ontology still remains open. Another opportunity is to use a *distributionally-based semantic similarity* approach. In this last case the key idea is that the semantic content of a word can be predicted studying how the word occurs with other words in a corpus. Two words are considered semantically related if they co-occur in a certain number of documents. In [Brown *et al.*, 1991], a vector containing all the immediately succeeding words in the document, is assigned to each term. For each of these words it is reported the number of times they occur after the considered term in the whole corpus. Then a notion of distance between two terms is defined as the average mutual information among all the pairs of words in the context of the two terms.

The other problem we addressed in this section is that the same word can drastically change its meaning in different contexts, thus it should be disambiguated to avoid misapprehensions. There are two main approaches to solve this problem. In theory, one should attempt exploit a sort of *world knowledge* that makes it possible to determine in which sense a word is used. Moreover, this method must be endowed with an inference mechanism that would make use of the base of knowledge to infer the intended sense of words. Clearly, this approach is not suitable in practice because a computer readable general purpose knowledge base for this task does not exist. Some effort was spent to design knowledge-bases and inference systems, but they are usually very limited since they are essentially handmade. What disambiguation systems do in practice is not to try to understand the meaning of text at all, but simply trying to guess the correct meaning of a word looking at the context in which it appears. The key idea of this approach is that, after observing several contexts in which a word is used, it can be possible to disambiguate a

word only considering its own context. Many different methods were proposed in the literature; a good survey of the most important ones and related problems can be found in [Ide and Veronis, 1998], [Sanderson, 2000], [Stokoe *et al.*, 2003] and [Linden, 2005]. Here we limit to mention two of the most important results.

Despite the fact that many important ideas and algorithms for word sense disambiguation have been proposed in the literature since the 50's, the first working disambiguator was written by Lesk [Lesk, 1986] in 1986. It was the first software that, for its characteristics, could be used for large text corpora. It was based on the use of an on-line dictionary. In order to disambiguate a term in a certain context, all its possible definitions were looked up in the dictionary. Then all the definitions were treated as a document corpus. The context of the term to disambiguate was, instead used as a query. Thus the problem of word sense disambiguation reduced to a ranking problem (or alternatively to a similarity searching problem). Dictionary clues are too small pieces of text and this negatively affects the disambiguator precision. Since large text corpora became available to researcher, they were employed to overcome this problem.

In [Gale *et al.*, 2004], a hybrid approach that uses both a dictionary and a document corpus is proposed. The only requirement is that each document in the corpus must be available in at least two different languages. To this purpose they use for example the Canadian Hansards which are available in English and French. The dictionary is used to translate a tern from a language to the other. Note that an ambiguous word has typically at least one different translation for each meaning. For example the word *duty* is often translated as *droit* when used with the sense of tax and as *devoir* when mean obligation. In this way it is possible to automatically extract a certain number of instances for each meaning of the word. Moreover, all the meanings of a word can be ranked by collecting statistics of their frequencies in the corpus. These data are then arranged in a training set and a test set. Thus, statistical models can be used for word sense disambiguation.

2.2.1.3 Feature selection

Documents in the vector space model are represented by very large and sparse vectors. Using the standard TF-IDF weighting scheme one can see them as points in the surface of the positive region of a iper-sphere. As explained before, the high dimensionality of documents in this representation has the side effect that the distances among documents become high and distances between pairs of similar documents tend to be close to the distance between pairs of unrelated ones. This is due to the summation of the contributes in the distance computation given by uninformative words. The goal of feature selection is to remove (or at least drastically reduce) the dimensionality of the vectors by removing the not informative words. There are two main strategies for feature selection: one dependent from the content of the corpus, the other independent. Stemming and stop word removal go in the latter direction. Despite the fact that these strategies achieve a good filtering, they are unable to remove those words that are uninformative in a particular context.

For this reason a lot of effort was spent to design algorithms that are able to take into account also the corpus content. Some of these methods like Latent Semantic Indexing (LSI) [Deerwester *et al.*, 1990] and Random Projection [Bingham and Mannila, 2001] employ structural properties of the document corpus. Other methods use information theoretic indexes to measure the informativeness of a word and filter those terms out of a certain range.

information theoretic indexes for feature selection Different indexes were proposed in the literature to measure the informative strength of a word, the great majority of them are suitable only for classification. The key idea in this case is to measure the degree of informativeness of the word for each class and then discard those terms with a poor informative power for all classes. Few indexes are suitable for unsupervised learning, in essence they are based on the document frequency. The most common of these measures used for feature selection are:

- **Document Frequency (DF)**: is the number of distinct documents in which a certain word appears in. This number is often normalized to be in the range $[0, 1]$ by dividing it by the total number of documents in the corpus. According to section 2.2, words that appear with high frequency are useless because they do not exploit differences among documents, rare words are also useless because their contribution in the distance computations is negligible. Thus words with document frequency above or below certain thresholds can be discarded.

- **Term Strength**: was originally proposed in [Wilbur and Sirotkin, 1992] for vocabulary reduction in text retrieval. As the document frequency, this index is not task-dependent, thus can be applied also to clustering. Moreover some variants were proposed in the literature specific for the text categorization problem [Yang and Wilbur, 1996]. We describe here its original definition. Term strength collects statistics about the presence of a word in pairs of related documents in a training set, then it uses these statistics to assign a score to the word. A pair of documents is considered to be related if their distance (usually the cosine distance) is under a certain threshold. Let d_1 and d_2 be two related documents and w a word, term strength is defined as:

  $$TS(w, d_1, d_2) = P(w \in d_1 | w \in d_2)$$

  In other words, given two related documents, term strength is the conditional probability that a word occurs in a document given that it occurs in the other. Let (d_i, d_j) be a pair of related documents in the training set, the term strength is:

  $$TS(w) = \frac{\#(d_i, d_j) : w \in d_i \wedge w \in d_j}{\#(d_i, d_j) : w \in d_i \vee w \in d_j}$$

- **Gain**: let n the number of documents in the corpus and df_w the number of documents in which the word w appears in, the *Gain* [Papineni, 2001] function is defined as:

$$Gain(w) = \frac{df_w}{n} * \left(\frac{df_w}{n} - 1 - \log \frac{df_w}{n} \right)$$

In this case the gain function assigns a low score both to rare and to common words. On the contrary of DF which requires a range of admissible values, in this case all the words with gain under a certain threshold are discarded. An important difference between Gain and DF is that in the former case the connection between rare and common filtered words is explicit. Figure 2.1 shows the Gain function.

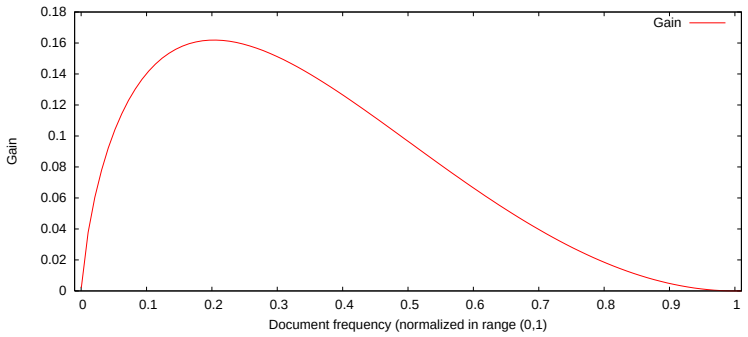

Figure 2.1. The gain function.

- **Information Gain**: is only suitable in the context of classification. The information gain function [Cover and Thomas, 1991; Yang and Pedersen, 1997] measures the contribution in terms of informativeness that the presence or absence of a word gives to a certain class. Let d be a document taken uniformly at random in the set of documents D. $P(v_i)$ is the probability that d contains term v_i, $P(c_j)$ is the probability that d is in category c_j. The complementary events are denoted $P(\bar{v_i}) = 1 - P(v_i)$ and $P(\bar{c_j}) = 1 - P(c_j)$. $P(v_i, c_j)$ is the probability that d is in category c_j and contains term v_i, $P(\bar{v_i}, \bar{c_j})$ is the probability d does not contain v_i and is not in category c_j. $P(v_i, \bar{c_j})$ is the probability that d contains v_i but is not in category c_j $P(\bar{v_i}, c_j)$ is the probability that d does not contain v_i and is in category c_j. Clearly being these mutually disjoint events it holds: $P(v_i, c_j) + P(\bar{v_i}, \bar{c_j}) + P(v_i, \bar{c_j}) + P(\bar{v_i}, c_j) = 1$. The information gain is the contribution of the four terms:

$$IG(v_i, c_j) = \sum_{v \in \{v_i, \bar{v_i}\}} \sum_{c \in \{c_j, \bar{c_j}\}} P(v, c) \log \frac{P(v, c)}{P(v)P(c)}$$

Note that information gain assigns a score to v_i only for a category. In order to have a global score for the term v_i different choices are possible. The most common is:

$$IG_{max}(v_i, c_j) = \max_{c_j \in C} IG(v_i, c_j)$$

In this case the key idea is that, if v_i is highly informative for at least one class, its presence helps to classify documents of that class.

- **Gain Ratio**: attempts to overcome some drawbacks of information gain. In fact the value of the information gain formula does not only depends on w_i and c_j, but also from the entropy of the class. Thus normalizing this factor we obtain the gain ratio formula:

$$GR(v_i, c_j) = \frac{IG(v_i, c_j)}{-\sum_{c \in \{c_j, \bar{c_j}\}} P(c) \log P(c)}$$

- **Mutual Information**: is a measure of the degree of dependence between a document d and a class c. Like in the case of information gain, this index is only suitable in the context of classification. More formally mutual information is defined as:

$$MI(w, c) = \log P(w|c) - \log P(w)$$

where w is a word and c is a class. The main drawback of mutual information is that it is highly influenced by the term $P(w)$. Thus for an equal value of the conditional probability rare terms are highlighted. Similarly to information gain a global score for a term w can be computed by one of the following formulas:

$$MI_{avrg}(w, c) = \sum_{i=1}^{m} P(c_i) MI(w, c_i)$$

$$MI_{max}(w, c) = \max_{i=1}^{m} MI(w, c_i)$$

A comparison of many of the above described measures can be find in [Yang and Pedersen, 1997].

Random projection One of the most simple and effective method to reduce the dimensionality of the vector space is the *random projection* [Kaski, 1998; Lin and Gunopulos, 2003]. The idea behind random projection is to reduce the dimensionality of the document matrix by multiplying it for a random projection matrix. More precisely: let M be the document matrix with n documents and m features. Suppose we want to reduce the vector space to be k-dimensional, with $k < m$.

Let R a matrix formed by m random k-dimensional vectors. We can project the original document vectors onto a lower dimensional space by:

$$A_{[k\times n]} = R_{[k\times m]} \cdot M_{[m\times n]}$$

Random projection is motivated by the Johnson-Lindenstrauss lemma [W.Johnson and j. Lindenstrauss, 1984]:

Theorem 1. *Let n an integer and $0 < \epsilon < 1$ and k such that*

$$k \geq 4\left(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}\right)^{-k} \ln n$$

Then for any set M of n points in $\mathbb{R}^m$ there exist a map $f : \mathbb{R}^m \to \mathbb{R}^k$ such that

$$\forall u, w \in M \qquad (1-\epsilon)\|u-w\| \leq \|f(u)-f(w)\| \leq (1+\epsilon)\|u-w\|$$

In simple words, according with the above lemma a set of points in a high-dimensional Euclidean space can be mapped in a lower-dimensional space such that distances between pairs of points are approximately preserved.

One of the mayor issues in the random projection method is the choice of the vectors of R. In theory, if the random vectors are orthogonal the distances between the original points are exactly preserved, thus an orthogonal matrix is desired. In practice, orthogonalization is a very costly operation, thus a reasonable approximation is used. In the literature many methods were proposed to initialize the elements of R, in the most common case they are Gaussian distributed. In [Achlioptas, 2003] two simple possible alternative initializations were proposed to reduce the computational time needed for the calculation for $R \times M$:

- $r_{i,j} = 1$ with probability $1/2$ otherwise $r_{i,j} = -1$
- $r_{i,j} = \sqrt{3} \cdot \begin{cases} -1 & \text{with prob. } 1/6 \\ 0 & \text{with prob. } 2/3 \\ 1 & \text{with prob. } 1/6 \end{cases}$

Latent semantic indexing

The main idea behind *latent semantic indexing* (LSI) is to project documents into a low-dimensional space with "latent" semantic dimensions. Even in the case in which two documents do not share terms in the original vector space they can still have a high similarity score in the target space as long as they share "semantically" similar words. Latent semantic indexing is based on the *Singular Value Decomposition* (SVD) applied to the document matrix M.

Latent semantic indexing takes the document matrix M and represents it as a matrix $\hat{M}$ in a k dimensional space ($k << m$) such that it minimizes the following:

$$\Delta = \|M - \hat{M}\|_2 \tag{2.1}$$

Let M be the document matrix with n documents and m features. Using SVD, latent semantic indexing decomposes M into three matrices such that:

$$M_{[m\times n]} = T_{[m\times r]}\Sigma_{[r\times r]}(D_{[n\times r]})^T$$

where T and D are orthogonal matrices that contain respectively the left and the right singular vectors of M and represent the terms and documents in the target space. Σ is a diagonal matrix that contains the singular the values of M and r is the rank of M. If values on Σ are sorted in decreasing order, SVD can be seen as a method to rotate the axes of the target space such that to the i-th axis is associated to the direction with the i-th largest variation. Thus singular values in Σ can be used to rank the "importance" of each dimension in the target space. As a direct consequence latent semantic indexing attempts to reduce the dimensionality in a way such that the dimensions of the target space correspond to the axes of greatest variation.

By restricting the matrices T, Σ and D to their first $k < r$ columns we obtain:

$$\hat{M}_{[m\times k]} = T_{[m\times k]}\Sigma_{[k\times k]}(D_{[n\times k]})^T$$

which is the best approximation for equation 2.1. At this point, to move from the m-dimensional space of words to the k-dimensional space of concepts, documents can be represented as the rows of the following matrix:

$$Z_{[k\times n]} = \Sigma_{[k\times k]}(D_{[n\times k]})^T$$

Despite the high computational cost, latent semantic indexing is one of the most powerful techniques for dimensionality reduction. The choice of the value of k is arbitrary and it is still one of the mayor issues for LSI. A too aggressive dimensionality reduction can negatively affect the quality of results while a too mild reduction can leave noise in the vector space. Typical choices for k are in the range of 100 - 150 features.

Bibliography

[Achlioptas, 2003] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

[Agirre and Rigau, 1996] E. Agirre and G. Rigau. Word sense disambiguation using conceptual density. In *Proceedings of COLING'96*, pages 16–22, 1996.

[Bingham and Mannila, 2001] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM.

[Bradley and Fayyad, 1998] Paul S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 91–99, Madison, US, 1998.

[Brown *et al.*, 1991] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. Word-sense disambiguation using statistical methods. In *ACL 29*, pages 264–270, 1991.

[Charikar, 2002] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of STOC-02, 34th Annual ACM Symposium on the Theory of Computing*, pages 380–388, Montreal, CA, 2002.

[Clarkson, 2006] Kenneth L. Clarkson. Nearest-neighbor searching and metric space dimensions. In Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006.

[Cover and Thomas, 1991] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, New York, US, 1991.

[Deerwester *et al.*, 1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[Dhillon *et al.*, 2002] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. Enhanced word clustering for hierarchical text classification. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 191–200, New York, NY, USA, 2002. ACM.

[Elkan, 2003] Charles Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, pages 147–153, 2003.

[Feder and Greene, 1988] Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444, New York, NY, USA, 1988. ACM Press.

[Gale *et al.*, 2004] William A. Gale, Kenneth W. Church, and David Yarowsky. A method for disambiguating word senses in a large corpus, 2004.

[Geraci *et al.*, 2007] Filippo Geraci, Mauro Leoncini, Manuela Montangero, Marco Pellegrini, and M. Elena Renda. Fpf-sb: a scalable algorithm for microarray gene expression data clustering. In *Proceedings of 1st International Conference on Digital Human Modeling*, 2007.

[Gonzalez, 1985] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(2/3):293–306, 1985.

[Guha *et al.*, 1998] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 73–84, New York, NY, USA, 1998. ACM Press.

[Hochbaum and Shmoys, 1985] D. S. Hochbaum and D. B. Shmoys. A best possible approximation algorithm for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[Ide and Veronis, 1998] Nancy Ide and Jean Veronis. Word sense disambiguation: The state of the art, 1998.

[Karypis *et al.*, 1999] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar NEWS. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[Kaski, 1998] Samuel Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of IJCNN'98, International Joint Conference on Neural Networks*, volume 1, pages 413–418, Piscataway, NJ, 1998. IEEE Service Center.

[Kaufman and Rousseeuw, 1990] Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley, 1990. A Wiley-Interscience publication.

[Kohonen, 2001] Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[Kowalski, 1997] Gerald Kowalski. *Information Retrieval Systems: Theory and Implementation*. Boston Kluwer Academic Publishers, 1997.

[Lamrous and Tailerb, 2006] Sid Lamrous and Mounira Tailerb. Divisive hierarchical k-means. In *CIMCA '06: Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce*, page 18, Washington, DC, USA, 2006. IEEE Computer Society.

[Lance and Williams, 1967] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies 1. Hierarchical systems. *The Computer Journal*, 9(4):373–380, February 1967.

[Larsen and Aone, 1999] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of KDD-99, 5th ACM International Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.

[Lesk, 1986] Michael Lesk. Automatic sense disambiguation: How to tell a pine cone from an ice cream cone. In *Proc. of the 1986 SIGDOC Conference*, pages 24–26, New York, 1986. Association for Computing Machinery.

[Li and Abe, 1998] Hang Li and Naoki Abe. Word clustering and disambiguation based on co-occurrence data. In *Proceedings of the 17th international conference on Computational linguistics*, pages 749–755, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[Lin and Gunopulos, 2003] J. Lin and D. Gunopulos. Dimensionality reduction by random projection and latent semantic indexing. In *Proceedings of the Text Mining Workshop at the 3rd SIAM International Conference on Data Mining*, 2003.

[Linden, 2005] Krister Linden. *Word Sense Discovery and Disambiguation*. PhD thesis, University of Helsinki, Faculty of Arts, Department of General Linguistics, 2005.

[Lloyd, 1957] S.P. Lloyd. Least squares quantization in PCM. Technical report, Bell Laboratories, 1957. Reprinted in *IEEE Transactions on Information Theory* IT-28(2), 1982, pp. 129–137.

[MacQueen, 1967] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[Miller *et al.*, 2006] George A. Miller, Christiane Fellbaum, Randee Tengi, Pamela Wakefield, Rajesh Poddar, Helen Langone, and Benjamin Haskell. Wordnet: A lexical database for english, 2006.

[Miller, 1990] George A. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 1990.

[Milligan and Cooper, 1985] Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, V50(2):159–179, June 1985.

[Ng and Han, 1994] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, pages 144–155, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann Publishers.

[Papineni, 2001] Kishore Papineni. Why inverse document frequency? In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[Peña *et al.*, 1999] José Manuel Peña, Jose Antonio Lozano, and Pedro Larrañaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.

[Phillips, 2002] Steven J. Phillips. Acceleration of k-means and related clustering algorithms. In *Proceedings of ALENEX-02, 4th International Workshop on Algorithm Engineering and Experiments*, pages 166–177, San Francisco, US, 2002.

[Porter, 1980] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[Rousseeuw, 1987] P.J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal ofCcomputational and Applied Mathematics*, 20:53–65, 1987.

[Sanderson, 2000] Mark Sanderson. Retrieving with good sense. *Information Retrieval*, 2(1):49–69, 2000.

[Savaresi *et al.*, 2002] Sergio M. Savaresi, Daniel Boley, Sergio Bittanti, and Giovanna Gazzaniga. Cluster selection in divisive clustering algorithms. In Robert L. Grossman, Jiawei Han, Vipin Kumar, Heikki Mannila, and Rajeev Motwani, editors, *SDM*. SIAM, 2002.

[Selim and Ismail, 1984] S.Z. Selim and M.A. Ismail. K-means type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):81–87, 1984.

[Shamir and Sharan, 2002] R. Shamir and R. Sharan. *Algoritmic Approaches to Clustering Gene Expression Data, Current Topics in Computational Molecular Biology*. MIT Press, 2002.

[Smellie, 2004] A. Smellie. Accelerated k-means clustering in metric spaces. *Journal of Chemical Information and Modeling*, 44(6):1929–1935, 2004.

[Stokoe *et al.*, 2003] Christopher Stokoe, Michael P. Oakes, and John Tait. Word sense disambiguation in information retrieval revisited. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 159–166. ACM Press, 2003.

[Strehl, 2002] Alexander Strehl. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, University of Texas, Austin, US, 2002.

[Tan *et al.*, 2005] Xiaoyang Tan, Songcan Chen, Zhi-Hua Zhou, and Fuyan Zhang. Feature selection for high dimensional face image using self-organizing maps. In *PAKDD*, pages 500–504, 2005.

[Tibshirani *et al.*, 2005] R. Tibshirani, G. Walther, D. Botstein, and P. Brown. Cluster validation by prediction strength. *Journal of Computational and Graphical Statistics*, 14:511–528, 2005.

[Tou and Gonzalez, 1977] J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, Reading, MA, 1977.

[Van Rijsbergen, 1979] Cornelis J. Van Rijsbergen. *Information Retrieval*. Butterworths, London, UK, second edition, 1979.

[Wilbur and Sirotkin, 1992] W. John Wilbur and Karl Sirotkin. The automatic identification of stop words. *J. Inf. Sci.*, 18(1):45–55, 1992.

[W.Johnson and j. Lindenstrauss, 1984] W.Johnson and j. Lindenstrauss. Extensions of lipschitz mapping into hilbert space. *Contemp. Math*, 26:189–206, 1984.

[Yang and Pedersen, 1997] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997.

[Yang and Wilbur, 1996] Yiming Yang and John Wilbur. Using corpus statistics to remove redundant words in text categorization. *J. Am. Soc. Inf. Sci.*, 47(5):357–369, 1996.

[Zhang and Couloigner, 2005] Qiaoping Zhang and Isabelle Couloigner. A new and efficient k-medoid algorithm for spatial clustering. In *ICCSA (3)*, pages 181–189, 2005.

[Zhang *et al.*, 1996] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.