Semi-Supervised and Unsupervised Machine Learning

Semi-Supervised and Unsupervised Machine Learning

Novel Strategies

Amparo Albalate
Wolfgang Minker

ISTE WILEY

First published 2011 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

Library of Congress Cataloging-in-Publication Data

Albalate, Amparo.
Semi-supervised and unsupervised machine learning / Amparo Albalate, Wolfgang Minker.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-84821-203-9 (hardback)
1. Data mining. 2. Discourse analysis--Statistical methods. 3. Speech processing systems. 4. Computational intelligence. I. Minker, Wolfgang. II. Title.
QA76.9.D343A3393 2011
006.3--dc22

2010040730

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN 978-1-84821-203-9

Printed and bound in Great Britain by CPI Antony Rowe, Chippenham and Eastbourne.

Table of Contents

Part 1
State of the Art

Chapter 1

Introduction

The main objective of this book is to develop machine learning (ML) tools that help minimize the (costly) human supervision required for the analysis of large volumes of data. To address such an objective, the research work developed in this book focused on two major fields in ML: unsupervised and semi-supervised learning. Both ML areas have been widely used in a large number of applications such as the clustering and semi-automatic annotation of large datasets of documents and the dimensionality reduction of microarray matrices for the analysis and interpretation of genomic data. In these examples, owing to the complexity and/or size of the large amounts of data to be processed, a fully supervised analysis without the help of semi- or unsupervised ML tools would become prohibitive.

Thus, the first aim of this book focused on the development of new algorithms in the field of semi-supervised ML. Semi-supervised learning provides an alternative to fully supervised classification. In supervised classification, a so-called training phase is performed using only labeled data. Typically, the labels for the training observations are

Semi-Supervised and Unsupervised Machine Learning: Novel Strategies Amparo Albalate and Wolfgang Minker

manually compiled by human annotators. Then, a supervised algorithm is capable of inferring prediction rules or models from the available training data and consequently delivering the most probable label for a new observation, not necessarily observed in the training data. However, a major limitation of supervised algorithms is related to the availability of large corpora labeled in order to achieve accurate predictions. As it is generally accepted in the ML literature, the performance of supervised classifiers can drastically drop down if only training sets of small dimensions are available.

In [CAS 95] it was shown that some advantage could be, however, gained if a large amount of unlabeled data is available. In particular, this is possible to the degree to which class labels fulfill *certain assumptions* that allow us to identify the class structure from both labeled and unlabeled data. The framework of classification algorithms designed to use both labeled and unlabeled data to generate their prediction models is known as *semi-supervised classification*.

Nowadays, the semi-supervised learning field is rapidly evolving, as evidenced by the large amount of semi-supervised approaches available in the machine learning literature, including generative models, co-training, self-training, and graph-based models etc. Frequently, the learning strategy followed by many semi-supervised algorithms can be summarized as follows: (1) select a supervised algorithm with a certain learning rule for labeled data and (2) modify the learning rule by including unlabeled data so that a common objective is attained. A drawback of such a strategy is the algorithms' stability/robustness with respect to the existence of labeling errors. Given the human effort involved in the manual labeling task, training sets are not exempted from potential labeling errors. These may occur depending on the degree of expertise of the human annotators. Even for expert labelers, the confidence in annotating patterns

with a certain degree of ambiguity may drop drastically. Hence, a subjective bias in annotating this kind of pattern is unavoidable. Depending on the nature of the classification task and corpora, subjective biases may become a commonly faced problem, as happens in the recognition of emotional states.

Given the aforementioned statement, in this book two different approaches to semi-supervised classification are described which rely on unsupervised clustering as a prior step to classification. By clearly separating the clustering and classification objectives, the proposed algorithms may gain some robustness under labeling errors with respect to other existing semi-supervised algorithms. The first algorithm has been developed for utterance corpora. It exploits the semantic feature variability by means of prior feature clustering, which is combined with a "fully unsupervised" algorithm for pattern disambiguation. The second approach performs the clustering in the pattern space to extract the underlying class structure and uses the labeled sets to automatically annotate the clusters.

The second aim of this book is to identify the underlying classes in a dataset in a fully unsupervised way, i.e. under the absence of labels. The field of unsupervised learning has witnessed an accelerated growth since the mid-1940s (see Chapter 2 for detail information), resulting in a large pool of clustering algorithms in the ML literature. However, the first question that arose with the use of a clustering algorithm is the optimum number of clusters to be selected. Most clustering algorithms are parametric approaches, which may explicitly require the number of clusters k as an input parameter, or implicitly, other types of parameters that also require appropriate estimation. A number of cluster validation techniques have been also proposed for an attempt to estimate the optimum k, but most of them are reported

to provide individual performances *ad-hoc* which depend on the clustering algorithm and distance functions selected. A number of authors in the unsupervised literature have claimed that finding the optimum k still remains an open research topic.

In this respect, this book also provides some attempts to solve the problem of the number of clusters. The first developed algorithm is a non-parametric clustering tool, which automatically identifies the number of clusters during the clustering process itself. The second approach is a cluster validation strategy that attempts to overcome the individual performances of other existing cluster validation techniques through an adequate combination of them.

1.1. Organization of the book

The book is divided into three main parts:

– The first part (Chapters 1 and 2) states the main objectives and motivation of the contributed tools, and describes the state of the art of unsupervised and semi-supervised methods.

– The second part (Chapters 3 and 4) describes the main contributions of the book in the field of unsupervised learning. Semi-supervised tools take advantage of both labeled and unlabeled data. In Chapter 3, a semi-supervised scheme for the categorization of utterances by using a unique labeled example per category is described. Unlabeled data is then used to identify clusters of synonym words from the vocabulary. This way, the initial small vocabulary available to the classifier is expanded to the clusters of words. Furthermore, the main contribution of the book, which provides important improvements in categorization accuracy, consists of a new unsupervised scheme for the reallocation of ambiguous utterances based on the identification of the most informative words.

Chapter 4 describes a semi-supervised classification approach based also on prior clustering, but this time applied to patterns (e.g. utterances) instead of words. Then, a minimum number of labels were used to automatically annotate the patterns inside the obtained clusters. An optimum cluster labeling is achieved by defining a cost matrix in terms of overlapping labels inside the clusters and applying the Hungarian algorithm to derive the optimum assignment of labels to clusters. This way, the initial small training set is automatically extended to the whole clustered data. This semi-supervised approach has been evaluated and compared to a fully supervised approach in which the initial labeled seeds were directly used to train a supervised classifier.

– The third part of the book (Chapters 5 and 6) describes the contributions in unsupervised learning, particularly focused on the automatic detection of the number of clusters. Chapter 5 focuses on an existing algorithm, the pole-based overlapping clustering (PoBOC), which is, to the author's knowledge, the only fully non-parametric existing algorithm that is able to detect the number of clusters. Moreover, a hierarchical alternative, namely, *hierarchical pole-based clustering (HPoBC)*, is proposed to overcome a major limitation of PoBOC related to the extraction of clusters based on a concept of globally defined neighborhood. The new alternative applies PoBOC recursively to find clusters based on local neighborhoods. Both approaches have been compared with other traditional algorithms introduced in Chapter 2.

In Chapter 6, the detection of the optimum number of clusters (k) is addressed through cluster validation strategies. In particular, a combination approach is described which attempts to find the optimum k from the combination of validation curves obtained through traditional methods.

1.2. Utterance corpus

The first datasets are two corpora of utterance transcripts collected from spoken language dialogue systems (SLDSs). SLDS emerged in the mid-1990s as a new, important form of human–machine communication. In essence, the typical architecture of an SLDS can be observed in Figure 1.1.

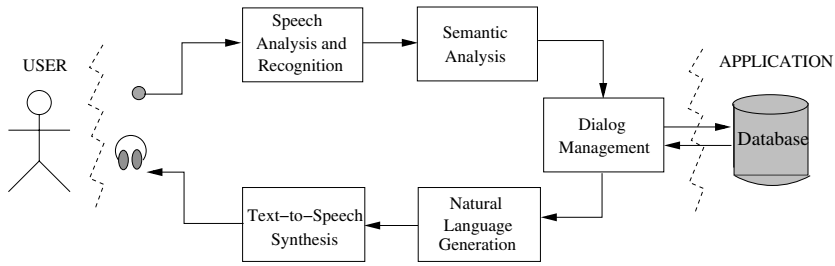


Figure 1.1. *Overview of an SLDS*

First, input acoustic vectors generated from the speech signal are processed by an Automatic Speech Recogniser (ASR), resulting in a raw text transcription [1] of the input utterance. Subsequently, the transcribed text is interpreted in a semantic analysis block which extracts the utterance meaning in form of an appropriate semantic structure. This semantic representation is processed by the dialog manager which also communicates directly with an external application, namely a database interface. The dialog manager keeps control of the overall interaction progress towards the task completion. During this process, the user may be queried for confirmations, disambiguations, necessary additional information, etc. Finally, the interaction result is presented to the user in form of speech (text-to-speech synthesis or prerecorded prompts).

The utterance corpora used in this book have been obtained from a particular type of SLDS known as *automated*

1. Most probable sequence of words detected.

troubleshooting agents. These are SLDS specially designed to perform customer care issues over the telephone, in a similar way as human agents do. One essential component of this type of SLDSs is the *semantic analysis* block. Typically, the semantic analysis in troubleshooting agents is carried out at the utterance level. The users are presented an open prompt, such as *"please briefly described the reason for your call."* Then, the semantic analysis block maps the unconstrained, *natural language* user response into one of the possible *problems* from a predefined list. This is done by means of statistical classifiers. This particular kind of semantic analysis is generally known as *Statistical Spoken Language Understanding (SSLU)*.

The machine learning tools developed in this book are focused on semi-supervised utterance classification for SSLU and the unsupervised discovery of potential symptoms. Both utterance corpora presented in the following are referred to different application domains of commercial troubleshooting agents and have been made available by SpeechCycle Labs [ACO 07].

The first corpus is related to the "Internet" domain. Some examples of transcribed utterances and their corresponding (manual) annotations "symptom categories" are shown in Table 1.1

It should be noted that an utterance is defined as "a complete unit of speech in spoken language." Although transcribed utterances may have a similar form as text sentences, utterances share the characteristics of natural language, in contrast to sentences in written language. Thus, it is not rare to observe ill-formed or gramatically incomplete utterance transcriptions (in particular when they reflect the users' responses to the system's prompts).

Utterance transcript	Annotation
Remotes not working	Cable
Internet was supposed to be scheduled at my home today	Appointment
Billing Information	Billing
I'am having Internet problems	Internet
I need to get the hi-def cable box installed	Cable
I need to talk to a representative	Operator

Table 1.1. *Some examples of utterances and their corresponding manual labels (annotations) in the utterance corpora*

The *Internet* utterance corpus is composed of a total of 34,848 utterances with $k = 28$ symptom categories. From these, 3,313 utterances (9.96%) have been used as test utterances, and 31,535 utterances (90.4% of the corpus) as the developing set.

The second utterance corpus is refered to *video* troubleshooting. Both test and training datasets are composed of 10,000 utterance transcripts and their corresponding symptom annotations. In total, $k = 79$ symptoms can be distinguished.

1.3. Datasets from the UCI repository

The UCI machine learning repository is one of the most popular collections of real word and simulated datasets to be used for machine learning experiments. In this book, the following datasets have been selected, with different characteristics (number of features and classes). The projections of the datasets into the three principal components can be observed in Figures 1.2 and 1.3.

1.3.1. *Wine dataset (wine)*

The wine set consists of 178 instances with 13 attributes, representing three different types of wines.

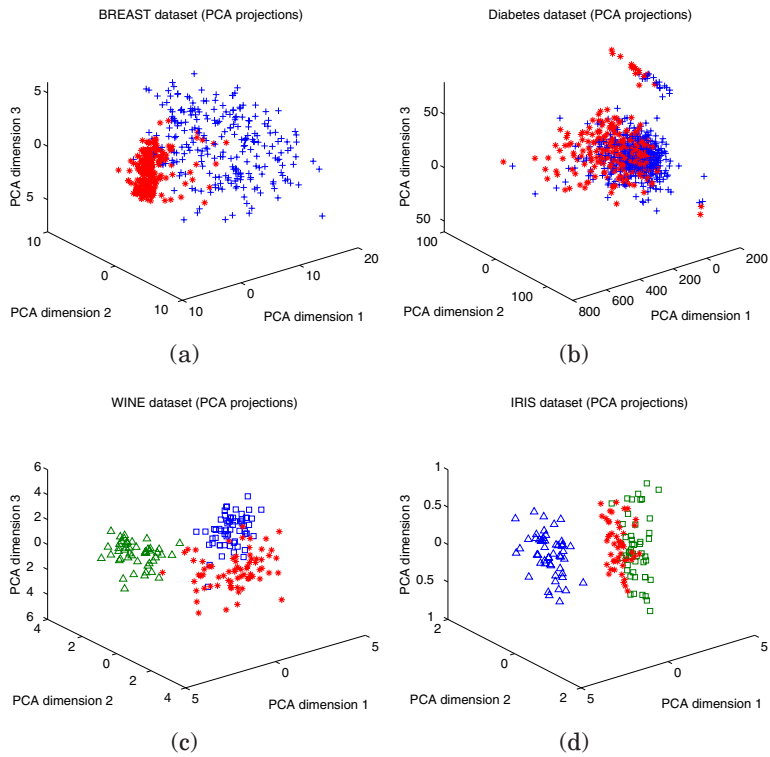


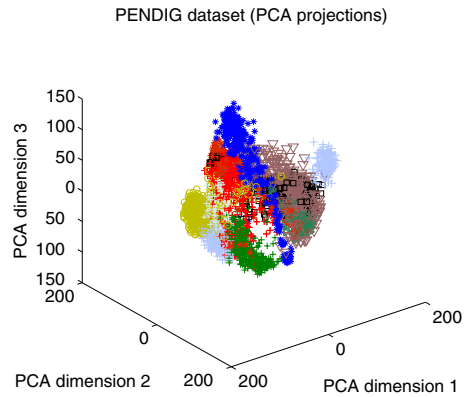
Figure 1.2. *Projection of the datasets on the three principal components: (a) breast dataset, (b) diabetes dataset, (c) wine dataset, and (d) Iris dataset*

1.3.2. *Wisconsin breast cancer dataset (breast)*

This dataset contains 569 instances in 10 dimensions, denoting 10 different features extracted from digitized images of breast masses. The two existing classes are referred to the possible breast cancer diagnosis (malignant, and benign).

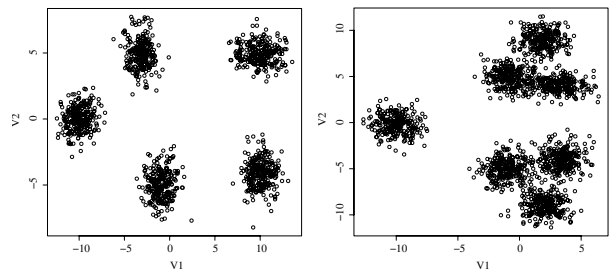
1.3.3. *Handwritten digits dataset (Pendig)*

The third real dataset is for pen-based recognition of handwritten digits. In this work, the test partition has been



(a)

Figure 1.3. *Pendig dataset (projection on the three principal components)*



(a) (b)

Figure 1.4. *Mixture of Gaussians datasets: (a) five Gaussians, and (b) seven Gaussians*

used, composed of 3,498 samples with 16 attributes. Ten classes can be distinguished for the digits 0–9.

1.3.4. *Pima Indians diabetes (diabetes)*

This dataset comprises 768 instances with eight numeric attributes. Two classes denote the possible diagnostics (the patients show or do not show signs of diabetes).

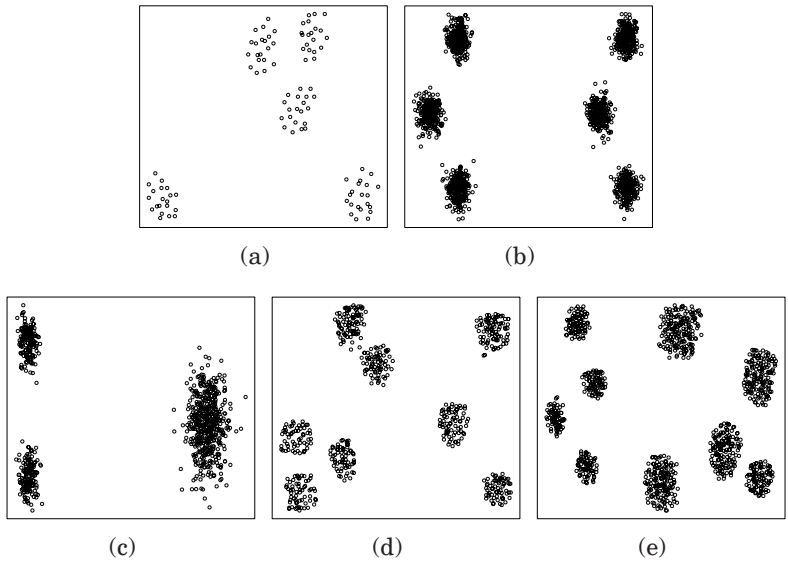


Figure 1.5. *Spatial data bases. (a) Dataset with 100 points in 5 spatial clusters (100p5c), (b) mixture of 6 Gaussians, 1500 points (6Gauss), (c) mixture of three Gaussians (3Gauss), (d) 560 points, 8 clusters (560p8c) and (e) 1000 points, 9 clusters (1000p9c)*

1.3.5. *Iris dataset (Iris)*

The Iris set is one of the most popular datasets from the UCI repository. It comprises 150 instances iris of three different classes of iris flowers (Setosa, Versicolor, and virginica). Two of these classes are linearly separable, the third one is not whereas linearly separable from the second one.

1.4. Microarray dataset

The *NCI60 dataset* [ROS 00] of the University of Standford has been also used. This dataset is publicly available at [NCI 06]. It consists of gene expression data for 60 cell lines derived from different organs and tissues. The data is a

1,375x60 matrix where each row represents a gene, and each column a cell line, related to a human tumor. Nine known tumor types and one unknown can be distinguished. The cancer types include: leukemia, colon, breast, prostate, lung, ovarian, renal, central nervous system, and melanoma.

1.5. Simulated datasets

Finally, simulated datasets in two dimensions have also been used to illustrate the behavior of the proposed algorithms and their outcomes.

1.5.1. *Mixtures of Gaussians*

The first datasets are two mixtures of five and seven Gaussians in two dimensions, where a certain amount of overlapping patterns can be observed.

1.5.2. *Spatial datasets with non-homogeneous inter-cluster distance*

These datasets comprise a number of patterns in two dimensions which build a hierarchy of groups, or groups non-homogeneouly distributed in the data space. Essentially, these datasets have been conceived for the demonstration of the *HPoBC* algorithm in Chapter 5.

To synthesize the data, a java applet has been implemented. The application is available online.

Chapter 2

State of the Art in Clustering and Semi-Supervised Techniques

2.1. Introduction

This chapter provides a survey of the most popular techniques for unsupervised machine learning (clustering) and their applications and introduces the main existing approaches in the field of semi-supervised classification.

2.2. Unsupervised machine learning (clustering)

Exploring large amounts of data to extract meaningful information about its group structure by considering similarities and differences between data entities has been a relevant subject under investigation since the early 1930s.

Each one of the data groups to be identified by the exploratory analysis is referred to as "cluster," and the framework of related techniques is known under the name "cluster analysis."

Semi-Supervised and Unsupervised Machine Learning: Novel Strategies Amparo Albalate and Wolfgang Minker

Formally, cluster analysis has been defined as:

> "the organisation of a collection of patterns (usually represented as a vector of measurements, or a point in a multi-dimensional space) into clusters based on similarity." [JAI 99]

2.3. A brief history of cluster analysis

The first studies about cluster analysis were conducted in the field of analytical psychology. There is some controversy in the literature about the attribution of the initial clustering schemes. However, according to D. E. Bailey [TRY 70], it seems to be Robert C. Tryon who originally conceived cluster analysis and its first application to psychological data. Tryon envisaged and formulated algorithms to group a set of intellectual abilities in humans, and provided a collection of scorings measured by the *Holzinger* ability tests [TRY 68]. Tryon referred to this particular practice of cluster analysis as *variable analysis* or *V-analysis*. The main purpose was to identify composites of abilities that could serve as more "general" and relevant descriptors than the whole set of scores for a more accurate analysis of human differences. This clustering method, called "key-cluster factor analysis," was proposed as an alternative to the factor analysis generally accepted at the time.

According to Bailey, this early form of cluster analysis was devised by Tryon in 1930. However, it was only three decades later, in 1965, that the method was implemented as part of a software package (BCTRY), following the introduction of the first modern computer at the University of California.

In those days, the field of classical taxonomy was also starting to be the object of important critiques about its conventional principles and practices. For Sokal and Michener [SOK 63], the main concern was the subjectivity of the

scientists involved in the elaboration of a taxonomy, and thus the individual biases associated with the systematic classification of organisms. This problem appeared to be aggravated by the advances in related sciences such as serology, chromatology, and spectroscopy, which led to the discovery of a vast amount of new variables (also called characters in biology) to describe the organisms. The resulting number of available characters turned out to make it infeasible for a taxonomist to estimate the affinities between taxonomic units (organisms, species, or higher rank taxa). In consequence, the scientists felt impelled to select a manageable group of characters from the complete set of available characters, which added a further subjective bias to the classical taxonomy procedures.

Having posed the aforementioned "illnesses" of classical taxonomy, Sneath and Sokal envisaged a series of numerical methods as the key to solve the weaknesses of classical taxonomy:

> "It is the hope of numerical taxonomy to aim at judgements of affinity based on multiple and unweighted characters without the time and controversy which seems necessary at present for the maturation of taxonomic judgements." [SOK 63]

The innovations introduced in the field of "numerical taxonomy" included the mathematic formulation of similarity metrics that enabled an objective comparison of specimens on the basis of their characters, as well as the application of hierarchical clustering to provide accurate representations of the new similarity matrices in the form of hierarchical trees or "dendograms".

According to [BLA 55], clustering analysis has witnessed an "exponential growth" since the initial contributions of Tryon and Sokal. Numerous clustering approaches arose in those

years for different applications, specially motivated by the introduction of different mathematical equations to calculate the distances/similarities between data objects [SOK 63].

One example was the application of clustering in the field of psychology, for grouping individuals according to their profile of scores, measured by personality or aptitudes tests, among others. Tryon, who initially devised a clustering scheme for grouping variables, also proposed the use of clustering algorithms for extracting different groups of persons with similar aptitudes (or composites of them) in the Holzinger study of abilities. He referred to this type of cluster analysis as "object analysis" or O-analysis, in contrast to its application to variable analysis. Later, he performed other studies, applying both V- and O-analyses to other domains. For example, one application was to cluster neighborhoods according to census data, in which the extracted clusters were referred to as social areas (the social area study [TRY 55]). Another application was to cluster a number of persons, among them psychiatric patients with psychological disorders as well as other normal subjects, into different groups of personalities (the Minnesota Multiphasic Personality Inventory (MMPI) Study [GRA 05]). To compute affinities between individuals, Tryon applied the Euclidean distance function.

Other contributions to cluster analysis were the works by Cox [COX 61, COX 62] and Fisher [FIS 58], who described hierarchical grouping schemes based on a single variable or the Ward's method, an extension to deal with the multivariate case. Ward proposed an objective metric based on the sum of squares error (SSE) to quantify the loss of information by considering a group (represented by its mean score) rather than its individual member scores.

Although psychology and biology (taxonomy) were the major fields of application of the initial clustering schemes, the exponential evolution of cluster analysis is patent in the

broad coverage of disciplines where clustering data has been an essential component (vector quantization [BUZ 80,EQU 89, GER 91], image segmentation [CLA 95, POR 96, ARI 06], marketing [SHE 96], etc.). According to [JAI 04], the number of clustering techniques developed from the early studies to date is "overwhelming."

2.4. Cluster algorithms

This section provides an overview of some popular clustering approaches. In particular, five different types of clustering algorithm are distinguished: hierarchical, partitional (specially competitive methods), model-based, density-based, and graph-based algorithms.

2.4.1. *Hierarchical algorithms*

In hierarchical clustering, a tree or hierarchy of clusters is incrementally built [ALD 64, HAS 09, WAR 63]. Assuming that the data objects are the leaves of the hierarchical tree, the hierarchical tree is build using two basic approaches: agglomerative (bottom-up) or divisive (top-down). These methods are described in the following sections.

2.4.1.1. *Agglomerative clustering*

Agglomerative methods build the hierarchy tree in a bottom-up manner, starting at the bottom hierarchy level (each object composes a singleton cluster) and successively merging clusters until a unique cluster is found at the top level.

The agglomerative algorithm is described in the following steps:

1. Start with each object in its individual cluster. Initially, the distances between the clusters correspond to the dissimilarities between data objects.

Method	Distance between clusters
Single	$d(C_a, C_b) = \min_{i,j} d(x_i \in C_a, x_j \in C_b)$
Complete	$d(C_a, C_b) = \max_{i,j} d(x_i \in C_a, x_j \in C_b)$
Average	$d(C_a, C_b) = \sum_{x_i \in C_a, x_j \in C_b} d(x_i, x_j)$
Centroid	$d(C_a, C_b) = d(\overline{C_a}, \overline{C_b})$

Table 2.1. *Cluster distances applied by the different hierarchical agglomerative algorithms*

2. Update the distances between the clusters, according to one of the criteria in Table 2.1.

3. Merge the pair of clusters with the minimum distance, calculated in Step 2.

4. Stop if all the data elements are enclosed in a unique global cluster (the top hierarchy level is reached). Otherwise, record the pair of clusters previously merged in Step 3 in the tree structure and repeat from Step 2.

Table 2.1 describes four different types of agglomerative approaches, depending on the criterion to calculate the distance between clusters before each cluster merging. The notation $d(\overline{C_a})$ indicates the centroid, or center, of cluster $\overline{C_a}$.

The hierarchy tree is typically visualized in the so-called dendogram plot, which depicts the pair of clusters that are merged at each iteration and the corresponding dissimilarity level. Figure 2.1 shows the dendogram plot obtained on a dataset with 20 different mammal species. From the hierarchy tree, a cluster partition can be achieved by specifying a distance level or a desired number of clusters to cut the dendogram.

2.4.1.1.1. Comparison of agglomerative criteria

In the single linkage, the clusters with the closest members are merged. Such merging criterion is local, as the global cluster structure is ignored. As can be observed in the

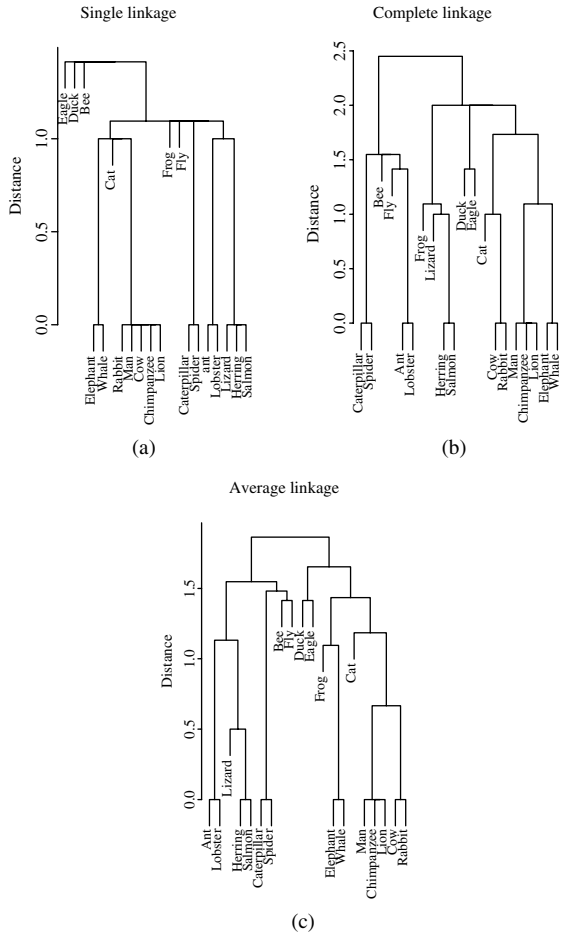


Figure 2.1. *Dendogram plots obtained by the single, complete, and average link agglomerative algorithms on a dataset of 20 different animals: (a) single linkage, (b) complete linkage, and (c) average linkage*

dendogram of Figure 2.1(a), the single linkage usually results in large clusters and may lead to erroneous splits of one cluster if two or more clusters are close to each other, specially if the clusters show different pattern densities (in this case, the pairwise distances in the low-density cluster may be

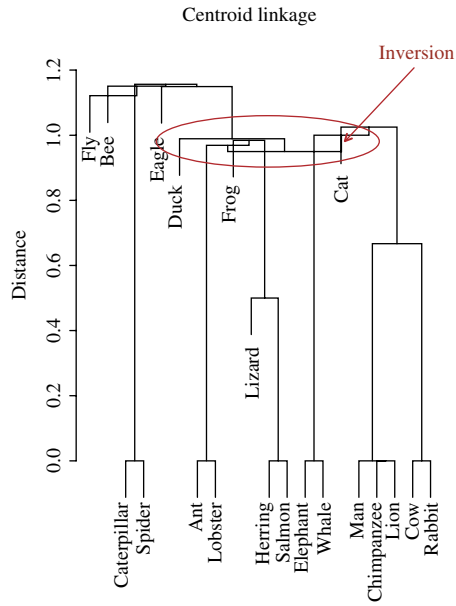


Figure 2.2. *Dendogram plot obtained by the centroid agglomerative algorithm on a dataset of 20 species of animals*

larger than the distance between the two clusters). On the other hand, the complete link criterion merges two clusters according to the most distant objects. This criterion overcomes the problem of large clusters obtained by the single linkage. However, the complete link criterion is sensitive to outliers. If a cluster is merged to an outlier at a certain iteration, its inter-cluster distances will become considerably larger at the next iteration. This is due to the presence of the outlier pattern. This fact may prevent merging the cluster to other members of the same cluster in further iterations. A good compromise seems to be the average linkage criterion. It merges two clusters according to the average pairwise distances between their respective members. Finally, in the centroid linkage criterion the distance between two clusters is calculated as the distance between the centroid elements. One important characteristic of the centroid linkage method

is that the cluster distances do not show increasing monotony with increasing iterations, in contrast to the single, complete, and average criteria. As an example, consider three globular, equally sized clusters, spatially arranged so that their centroids occupy the vertices of an approximately equilateral triangle, $\hat{abc}$. The distance of the first two clusters, say C_a and C_b (edge $\overline{ab}$), is only marginally inferior to the length of the other two edges in the triangle, so that the clusters C_a and C_b are merged at iteration t. The centroid of the new cluster will be then placed in the middle point of the edge $\overline{ab}$. The centroid of the third cluster corresponds to the third vertex. Thus, new distance between the new cluster C_{ab} and C_c, merged at iteration $t+1$, corresponds to the height of the triangle, $h \sim \frac{\sqrt{3}}{2}\overline{ab}$, which is smaller than the distance between the clusters C_a and C_b merged in the previous iteration. This non-monotonic distance effect is clearly patent in the new dendogram plot. It can be observed how mergings at a certain iteration can occur at a lower dissimilarity level than a previous iteration. Such effect is also known as dendogram *inversion*.

2.4.1.2. *Divisive algorithms*

Divisive algorithms build the cluster hierarchy in a top-down approach, starting at the top level, where all the patterns are enclosed in a unique cluster, and successively splitting clusters until each pattern is found in a singleton cluster. A greedy divisive algorithm requires the evaluation of all $2^{|C|-1}-1$ possible divisions of the cluster C into two subclusters (initially, $|C|=n$). However, Smith *et al.* proposed an alternative algorithm, called divisive analysis (DiANA), which avoids the inspection of all possible cluster partitions. The algorithm is described as follows:

1. Start with all objects in a unique cluster, C_g.

2. Select the cluster C' with the highest diameter (distance between the most distance objects inside the cluster). Initially,

start with the cluster C_g containing all patterns in the dataset, i.e. $C'(t=0) = C_g$.

3. For each object in C', calculate its average distance to all other objects in the cluster.

4. Select the pattern with the highest average distance in C'. This pattern originates into a new cluster C'' and is thus removed from C'.

5. For each one of the remaining objects, $x_i \in C'$, calculate the average distance to the rest of objects in C' and C''. Let this average distances be denoted as $d_{\text{avg}}(x_i, C')$ and $d_{\text{avg}}(x_i, C'')$. If $d_{\text{avg}}(x_i, C') > d_{\text{avg}}(x_i, C'')$, the pattern has more affinity to the new formed cluster. Select the pattern $\hat{x}$ whose affinity to C'' is the highest in comparison to C'.

$$\hat{x} = \underset{x_i \in C'}{\text{argmax}}(d_{\text{avg}}(x_i, C') - d_{\text{avg}}(x_i, C'')) \qquad [2.1]$$

Remove $\hat{x}$ from C' and attach to C''.

6. Repeat Step 5 until no object in C' can be found such that $d_{\text{avg}}(x_i, C') > d_{\text{avg}}(x_i, C'')$. At this point, the partition of the cluster C' is finished. Record the resulting clusters C' and C'' into the corresponding dendogram level.

7. Stop if each data pattern is in its own cluster (the bottom hierarchy level is reached). Otherwise, repeat from Step 2 to evaluate the subsequent cluster to be partitioned.

2.4.2. *Model-based clustering*

In model-based clustering, it is assumed that the data are drawn from a probabilistic model, λ, with parameters θ. Thus, the estimation of the model parameters θ is a crucial step for identifying the underlying clusters. A typical example of model-based clustering is the expectation maximization (EM) algorithm.

2.4.2.1. *The expectation maximization (EM) algorithm*

As mentioned above, the EM algorithm [DEM 77, NIG 99, MAC 08] is based on the assumption that the dataset $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is drawn from an underlying probability distribution, with a probability density function $p(X|\theta)$. The objective of the EM algorithm is to provide an estimation of the parameter vector $\hat{\theta}$ that maximizes the log-likelihood function

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}(\theta) = \log(p(\mathcal{X}|\theta)) = \log(\prod_{i=1}^{N}(p(x_i|\theta))) \qquad [2.2]$$

In a simple univariate Gaussian distribution, θ may be a vector of the mean and variance, $\theta = [\mu, \sigma^2]$. An analytical solution to the maximum log-likelihood problem can be directly found by taking derivatives of equation [2.2] with respect to μ and σ. However, for more complex problems, an analytical expression may not be possible. In such cases, the EM algorithm performs an iterative search for the numerical solution of [2.2].

Typically, an additional unobserved variable vector Z is introduced. The introduction of this variable allows us to simplify the mathematical analysis to derive the EM updating equations

$$p(\mathcal{X}|\theta) = \sum_{z} p(\mathcal{X}|z, \theta)P(z|\theta) \qquad [2.3]$$

where z denotes a particular value of the variable vector Z. Let $\theta^{(t)}$ denote the guess of the parameters θ at iteration t. The derivation of the EM formulation follows the Jensen's inequality

$$\log(\sum \lambda_i x_i)) \geq \sum (\log(\lambda_i x_i) \qquad [2.4]$$

with constants $\lambda_i \leq 1$, such that $\sum_i \lambda_i = 1$. At iteration t, the parameters $\theta^{(t)}$ are known. Thus, the constants λ_i can be defined as follows:

$$\lambda_i = p(z|X, \theta^{(t)}) \qquad [2.5]$$

with $\sum_i p(z|X,\theta^{(t)}) = 1$. These constants can be introduced in the log-likelihood formulation (equation [2.2]) as follows:

$$\begin{aligned} L(X|\theta) &= \sum_z \log p(X|z,\theta)\cdot p(z|\theta)\cdot \frac{p(z|X,\theta^{(t)})}{p(z|X,\theta^{(t)})} \\ &= \sum_z \log\left(p(z|X,\theta^{(t)})\left(\frac{p(X|z,\theta)\cdot p(z|\theta)}{p(z|X,\theta^{(t)})}\right)\right) \end{aligned} \quad [2.6]$$

Thus, by applying Jensen's inequality, it yields

$$\begin{aligned} L(X|\theta) &= \sum_z \log\left(p(z|X,\theta^{(t)})\left(\frac{p(X|z,\theta)\cdot p(z|\theta)}{p(z|X,\theta^{(t)})}\right)\right) \\ &\geq \sum_z P(z|X,\theta^{(t)}) \log\left(\frac{P(X|z,\theta)\cdot P(z|\theta)}{P(z|X,\theta^{(t)})}\right) \end{aligned} \quad [2.7]$$

Moreover, it can be shown that the log-likelihood equals equation [2.7] for $\theta = \theta^{(t)}$. Thus, equation [2.7] is a lower bound of the log-likelihood to be maximized, and equals this likelihood for $\theta = \theta^{(t)}$. This observation is key for the definition of an EM iteration. If a value $\theta^{(t+1)}$ is found to increase the value of equation [2.7], it will necessarily result in an increment of the log-likelihood value. The maximization objective can be thus reformulated as

$$\begin{aligned} & \underset{\theta}{\operatorname{argmax}}\left\{\sum_z p(z|X,\theta^{(t)}) \log\left(\frac{p(X|z,\theta)\cdot P(z|\theta)}{p(z|X,\theta^{(t)})}\right)\right\} \\ = \ & \underset{\theta}{\operatorname{argmax}}\left\{\sum_z p(z|X,\theta^{(t)}) \log\left(p(X|z,\theta)\cdot p(z|\theta)\right)\right\} \\ - \ & \underset{\theta}{\operatorname{argmax}}\left\{\sum_z p(z|X,\theta^{(t)}) \log\left(p(z|X,\theta^{(t)})\right)\right\} \end{aligned} \quad [2.8]$$

At iteration $t+1$, the last term in equation [2.8] is a constant factor with respect to θ. Thus, this term can be discarded in the maximization formulation, which can be rewritten as

$$\begin{aligned}\underset{\theta}{\text{argmax}}\{L(X|\theta)\} &= \underset{\theta}{\text{argmax}}\left\{\sum_z p(z|X,\theta^{(t)})\log\left(p(X|z,\theta)\cdot p(z|\theta)\right)\right\} \\ &= \underset{\theta}{\text{argmax}}\left\{\sum_z p(z|X,\theta^{(t)})\log\left(p(X,z|\theta)\right)\right\} \\ &= E_{z|X,\theta^{(t)}}\left\{\log\left(p(X,z|\theta)\right)\right\} \qquad [2.9]\end{aligned}$$

Thus, the objective of the EM algorithm is the maximization of the expectation of $\log p(X,z|\theta)$ over the current distribution of Z values given X under $\theta^{(t)}$. Typically, a definition of the variable vector Z is used so that each z realization can be calculated from the current estimate of the parameters, $\theta(t)$. The iteration $(t+1)$ of the EM algorithm consists of two alternating steps:

1. E-step: determine the expectation $E_{z|X,\theta^{(t)}}\left\{\log\left(p(X,z|\theta)\right)\right\}$. In practice, it requires the calculation of the probabilities $p(z|X,\theta^{(t)})$ given the data $\mathcal{X}$ and the previous parameter vector estimate $\theta^{(t)}$.

2. M-step: find the current estimation $\theta^{(t+1)}$ by maximising the expectation:

$$\theta_{(t+1)} = \underset{\theta}{\text{argmax}}\, E_{Z|X,\theta^{(k)}}\left\{\log p(X,z|\theta)\right\} \qquad [2.10]$$

An example of the EM algorithm for estimating the parameters of a mixture of Gaussians is described in the following.

2.4.2.1.1. Example: mixtures of Gaussians

For a mixture of Gaussians with M components, $G = \{g_1, g_2, \ldots, g_M\}$, the random variable vector Z is composed of N components, $\{z_i\}_{i=1}^{N}$, where $z_i \in [1, \ldots, M]$ denotes the Gaussian component of the corresponding data element x_i. The parameter vector θ is thus composed of the Gaussians means and variances (μ, Σ), as well as their marginal probabilities $p(g)$. Thus, θ is composed of M components, $(\theta_1, \ldots, \theta_M)$ for the M Gaussians, with $\theta_m = (\mu_m, \Sigma_m, p(g_m))$. The expectation in equation can be formulated as

$$E_{z|x,\theta^{(t)}}\left\{\log p(X, z|\theta)\right\} = \sum_{z}\left\{p(z|x, \theta^(t))\log p(x, z|\theta)\right\}$$

$$= \sum_{z1=1}^{M}\sum_{z2=1}^{M}\cdots\sum_{zN=1}^{M}\left(\prod_{x_i} p(z_i|x_i, \theta^(t))\left(\sum_{x_i}\log p(x_i, z_i|\theta)\right)\right) \quad [2.11]$$

After some manipulations, a more simple expression can be derived for equation [2.11] as follows:

$$E_{z|x,\theta^{(t)}}p(X, z|\theta) = \sum_{m=1}^{M}\sum_{i=1}^{N} p(z_i = m|x_i, \theta^{(t)})\log(p(x_i, z_i = m|\theta)$$

$$= \sum_{m=1}^{M}\sum_{i=1}^{N} p(z_i = m|x_i, \theta^{(t)})\log(p(x_i|\theta_m)p(z_i = m) \quad [2.12]$$

where $p(z_i = m)$ denotes the probability of the mth Gaussian component, $p(g_m)$.

In the E-step, the first part of the expectation in equation [2.12], $p(z_i = m|x_i, \theta^{(t)})$ can be calculated given the previous estimation of the parameters $\theta^{(t)}$:

$$p(z_i = m|x_i, \theta^{(t)} = \frac{p(x_i|\theta_m)p(z_i = m)}{p(x_i, \theta)}$$

$$= \frac{p(x_i|\theta_m)p(z_i = m)}{\sum_{m=1}^{M} p(x_i|\theta_m)p(g_m)} \quad [2.13]$$

After the calculation of the probabilities $p(z_i = m|x_i, \theta^{(t)}$, the Maximization of the expectation performed the M step, by taking the partial derivatives of equation [2.12] with respect to the parameters θ and setting their values to 0.

$$\sum_{m=1}^{M}\sum_{i=1}^{N} p(z_i = m|x_i, \theta^{(t)})\frac{\partial}{\partial\theta}\log(p(x_i|\theta_m)p(z_i = m) \qquad [2.14]$$

By considering the partial derivatives with respect to the individual components in θ (μ, σ and $p(g)$), the following solutions yield for $\mu^{(t+1)}, \Sigma^{(t+1)}$ and $p(g)^{(t+1)}$:

$$p(g_j)^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i|\theta_j^{(k)})}{\sum_{m=1}^{M} p(x_i|\Theta_m^{(t)})} \qquad [2.15]$$

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{N} x_i p(z_i = j|x_i, \theta^{(t)})}{\sum_{i=1^N} p(z_i = l|x_i, \theta^g)} \qquad [2.16]$$

$$\Sigma_j^{(t+1)} = \frac{\sum_{i=1}^{N} p(z_i = j|x_i, \theta^{(t)})(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{N} p(z_i = j|x_i, \theta^{(t)})} \qquad [2.17]$$

The E- and M-steps are iterated until the parameter vector $\theta^{(t+1)}$ remains unaltered with respect to the vector in the previous iteration, $\theta^{(t)}$. In this convergence state, a "soft" clustering solution is provided by the probabilities $p(z_i = j|x_i, \theta)$. However, a hard solution can be easily obtained by selecting the values of z_i which maximize these probabilities. The EM algorithm is considered as a generic modality of the k-means algorithm. In fact, both E- and M- steps are closely related to the k-means adaptation steps. The E-step recalculates the membership probabilities $p(z_i|x_i)$, given the previous parameter estimate (means and covariances).

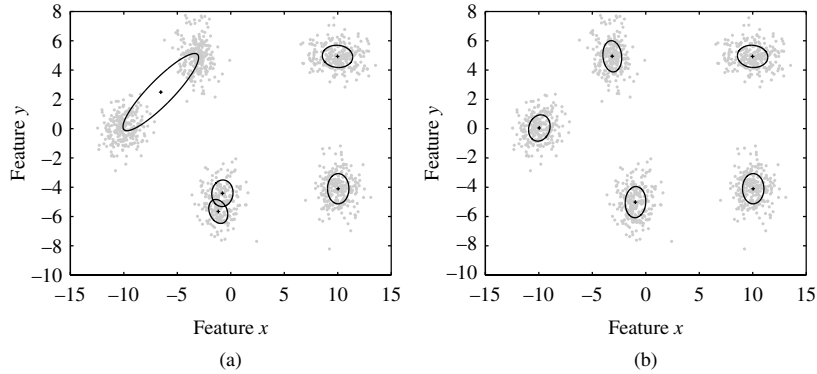


Figure 2.3. *Illustration of the parameter estimation on a mixture of five bivariate Gaussians through the Expectation Maximization Algorithm (EM), using two different random initializations. As can be observed in the plot 2.3(a), two of the five Gaussians are merged in the convergence state. Thus, the EM algorithm may tend to a local optimum with an inadequate initialization of the parameters. However, with an appropriate initialization of the parametes, the EM algorithm is capable to fit the parameters to the five existing Gaussians*

This step can be viewed of as a soft version of the pattern reallocation in the Voronoi cells of the respective means in k-means. Finally, the recalculation of the means in k-means correspond to the update of the $\theta^{(t+1)}$ in the M-step.

2.4.3. *Partitional competitive models*

Competitive models aim at discovering a set of reference prototypes, $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ with coordinates $\{w_1, w_2, \ldots, w_D\}$, which optimally represent the data distribution $\mathcal{D}$.

2.4.3.1. *K-means*

In general terms, assuming that the elements of $\mathcal{D}$ are drawn from a certain probability distribution, $P(x)$, the k-means [ART 06,BAR 00,HAR 05] objective is to find the values

of the prototype vectors to minimize the error in

$$E(\mathcal{D},\mathcal{P}) = \sum_{p_i \in \mathcal{P}} \int_{\mathcal{V}_i} \parallel x - w_i \parallel^2 p(x)\mathrm{d}x \qquad [2.18]$$

where $\mathcal{V}_i$ is the *Voronoi cell* associated with the reference vector p_i, i.e. the region of points in $\mathcal{R}^D$ that are closer to p_i than to any other reference vector in $\mathcal{P}$:

$$V_i = \{x \in \mathcal{R}^N \quad | \quad \underset{p_j \in \mathcal{P}}{\operatorname{argmin}} \parallel x - w_j \parallel\} = p_i \qquad [2.19]$$

If the probability distribution of the data, $P(x)$, is not known *a priori*, but only a discrete set of data points $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ is available, the error surface in equation [2.18] can be reformulated as

$$E(\mathcal{D},\mathcal{P}) = \sum_{p_i \in \mathcal{P}} \sum_{x \in \mathcal{V}_i} \parallel x - w_i \parallel^2 \qquad [2.20]$$

In the k-means algorithm, the search for the optimum weights of the prototypes is performed iteratively through a stochastic gradient descent on the error surface. At iteration t, the contribution of the data element $x(t)$ to the update of the prototype weights is thus given by

$$\Delta w_i = \epsilon(t) \cdot \delta_{ii}(x(t)) \cdot [x(t) - w_i] \qquad [2.21]$$

where $x(t)$ denotes the data point visited at time t, $\epsilon(t)$ is the step size parameter that controls the adaptation rate, and δ_{ii} is the Kronecker delta function defined as follows:

$$\delta_{ii}(x(t)) \begin{cases} 1, & \text{if } x(t) \in \mathcal{V}_i \\ 0, & \text{otherwise} \end{cases} \qquad [2.22]$$

Hence, the function $\delta_{ii}(x(t))$ deactivates any possible contribution of elements that lie outside the Voronoi region

of p_i. Owing to this "hard decision," the k-means algorithm is casted as a *hard competitive* model, which means that only points for which p_i is found to be the "*winning prototype*" account for the update of the prototype weights.

It should also be noted that, after each update of the prototype weights, the Voronoi region of the prototype p_i is recalculated. Moreover, the parameter $\epsilon(t)$ is not constant, but obeys a decaying pattern:

$$\epsilon(t) = \frac{1}{n(t)} \qquad [2.23]$$

where $n(t)$ is the number of visited data elements that have contributed to the update of the weights of p_i up to the current iteration t.

As it can be demonstrated, the prototypes' weight locations converge asymptotically to the mean expected value of their respective Voronoi cells:

$$w_i(t \to \infty) = \int_{\mathcal{V}_i} x\, p(x) \mathrm{d}x \qquad [2.24]$$

or equivalently, for a discrete dataset $\mathcal{D}$

$$w_i(t \to \infty) = \frac{\sum_{x_j \in \mathcal{V}_i} x_j}{|\mathcal{V}_i|} \qquad [2.25]$$

A batch implementation of the k-means clustering algorithm can be described as follows:

1. Initialize the set of prototypes $\mathcal{P}$ with k elements,

$$\mathcal{P}^{(t=0)} = \{p_1, p_2, \ldots, p_k\} \qquad [2.26]$$

whose weights or cooordinates w_i are randomly chosen according to $p(x)$ or selected by the user.

2. For each iteration t, build a set of clusters around the respective prototypes:

$$\mathcal{C}^{(t)} = \{C_1, C_2, \ldots, C_k\} \qquad [2.27]$$

in such a way that each cluster is composed by the data points belonging to the Voronoi cell of the winning prototype

$$C_i^{(t)} = \mathcal{V}_i = \{x_j \in \mathcal{D} \quad | \quad \underset{p_k \in \mathcal{P}}{\operatorname{argmin}} \parallel x_j - w_k \parallel = p_i\} \qquad [2.28]$$

3. Re-calculate the prototype weights or coordinates to coincide with the most central point of the Voronoi cell:

$$w_i^{(t)} = \frac{\sum_{x_j \in \mathcal{V}_i} x_j}{|\mathcal{V}_i|} \qquad [2.29]$$

4. Calculate the error at iteration t, $E^{(t)}$, according to equation [2.20].

5. Calculate the decrement of the error E at iteration t with respect to $t-1$:

$$\Delta_E = E^{(t)} - E^{(t-1)} \qquad [2.30]$$

6. Stop if the error difference Δ_E is lower than a specified value, $\Delta_E^{\min}$ (a convergence status is reached), or the maximum number of iterations is reached. Otherwise, continue from Step 2.

Figure 2.4 shows an example of the clustering solution obtained by the k-means algorithm on a mixture of five Gaussians with two different random initialization. As can be observed, the extracted clusters depend on the choice for the initial prototypes. An inappropriate initialization may lead to solutions far from the optimum (Figure 2.4(a)).

2.4.3.1.1. Advantages and drawbacks

One of the advantages of k-means is the fast convergence and thus the low computational cost of the algorithm

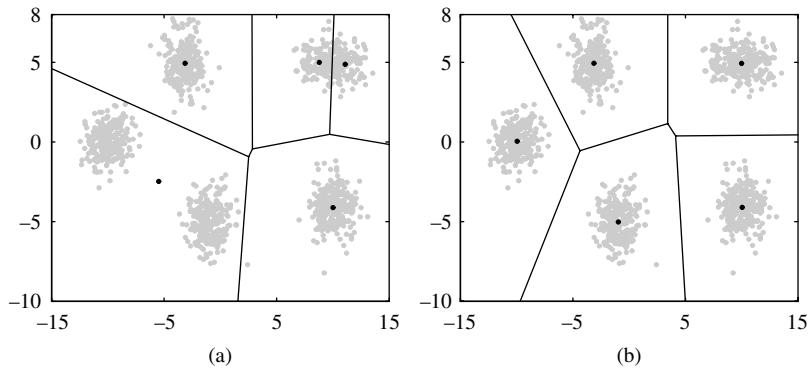


Figure 2.4. *Illustration of k-means with two different random initializations. The final centroids as well as the corresponding Voronoi cells can be observed*

[HAR 05]. However, the main limitation associated with the k-means algorithm may converge to a local optimum, depending on the choice of the initial prototypes. This problem is commonly addressed through multiple algorithm calls with different random initializations in a trial-and-error approach. The solution that provides the minimum sum of squared errors is selected.

Another limitation is associated with the sum of squared errors objective function. Patterns with large distances may considerably attract the cluster centers toward them. Consequently, the algorithm shows a high sensitivity to *outliers*.

A third limitation of k-means is the shape and size of clusters that can be detected. As the algorithm divides the dataset into Voronoi regions, the k-means algorithm extracts convex (globular) clusters. However, it may fail to identify other cluster shapes or clusters of non-homogeneous sizes, specially if those clusters lie close to each other.

Finally, the k-means algorithm requires the number of clusters k to be *a priori* determined. The correct identification of the optimal number of clusters existing in a dataset can be achieved through the evaluation of the cluster partitions obtained from multiple runs of the clustering algorithm with different values of k; see Chapter 6 for other possibilities.

2.4.3.2. *Neural gas*

As described in the preceding paragraphs, the weight adaptation of each prototype p_i by the k-means algorithm is only affected by the data points whose winning prototype is p_i. In contrast, in the neural gas approach [COT 00, FRI 95, FYF 06], each visited data point, $x_j \in \mathcal{D}$, has a certain impact on the adaptation of the weights for all prototypes, on a "*rank order*" basis (equation [2.31]):

$$\Delta w_i = \epsilon(t) \cdot h_\lambda(k_i(x(t), \mathcal{P})) \cdot (x(t) - w_i) \qquad [2.31]$$

As can be noted, the main difference with respect to the weight adjustment in k-means (equation [2.21]) is the substitution of the delta function by an exponentially decaying function (h_λ). Basically, this new function determines the relative (soft) importance of each presented data point at iteration t, $x(t)$, for the adaptation of the prototype p_i:

$$h_\lambda(k_i(x(t), \mathcal{P})) = e^{(-k_i(x(t),\mathcal{P})/\lambda)} \qquad [2.32]$$

where k_i indicates the rank order of the prototype p_i from all prototypes in $\mathcal{P}$ with respect to the data point $x(t)$ under consideration. In other words, k_i is the number of prototypes closer to $x(t)$ than p_i. For example, the closest prototype is assigned a rank order $k = 0$, and thus, $h_\lambda(k_i(x(t), \mathcal{P})) = 1$; Analogously, the next, second, third, etc., closest prototypes up to the farthest one are assigned rank values $k = 1, 2..., k - 1$. The λ parameter, $\lambda \in [0, 1]$, controls the decay rate of the h_λ

function, and thus, the "softness" of the neural gas model. For example, if $\lambda \rightarrow 0$, the function h_λ is equivalent to the Kronecker delta δ_{ii} and thus the resulting model becomes analogous to the k-means algorithm.

It can be shown that the adaptation rule in equation [2.31] responds to a gradient descent optimum search on the error function in equation [2.33]:

$$E_{NG}(\mathcal{D}, \mathcal{P}) = \frac{1}{2C(\lambda)} \sum_{i=1}^{k} \int h_\lambda k_i(\xi, \mathcal{P}) \parallel x - w_i \parallel^2 p(x) dx \quad [2.33]$$

For a discrete dataset, equation [2.33] is equivalent to [2.34]:

$$E_{NG}(\mathcal{D}, \mathcal{P}) = \frac{1}{2C(\lambda)} \sum_{i=1}^{k} \sum_{x_k in \mathcal{D}} h_\lambda k_i(x_k, \mathcal{P}) \parallel x_k - w_i \parallel^2 \quad [2.34]$$

The normalization factor $C(\lambda)$ in both equations [2.33] and [2.34] is a constant value that only depends on λ:

$$C_\lambda = \sum_{i=0}^{k-1} h_\lambda(k) \quad [2.35]$$

2.4.3.2.1. Advantages and drawbacks

The main advantage of the neural gas with respect to the k-means algorithm is the robustness with respect to the initialization. The neural gas can be viewed as a generalized version of Kohonen maps which applies to any topology. However, the algorithm requires, besides the number of clusters k, the specification of other parameters, such as the adaptation rate and the neighbor functions.

2.4.3.3. *Partitioning around Medoids (PAM)*

The Partitioning around Medoids (PAM) algorithm, proposed by Rousseau [LEO 05] is an alternative to the k-means algorithm based on *medoid* prototypes. In a similar way as k-means, the PAM algorithm aims at discovering a set of k representative objects that are more centrally located in the clusters. However, PAM differs from the k-means in several ways:

1. PAM searches for a set of k-medoid prototypes instead of k-centers. Centers (or centroids) are vectors located in the exact cluster "center of masses" and do not necessarily coincide with an object in the dataset. In contrast, medoids correspond to the *objects* in a dataset closer to the cluster means.

2. The objective function minimized by the PAM algorithm is the total sum of distances, instead of the sum of squared errors in k-means.

3. The PAM algorithm may be used with any arbitrary distance function. It also accepts the matrix of data dissimilarities as input instead of the dataset coordinates.

The search for the minimum sum of distances is performed in two main steps: build and swap.

2.4.3.3.1. Build step

This step performs a sequential search for an appropriate initial set of medoids $\mathcal{P}$, which are more "centrally located" [LEO 05] in the dataset:

1. Select the object $\hat{x}^{(0)}$ with the minimum sum of distances to the rest of elements in the dataset. Initially (at iteration $t = 0$), the set of prototypes is composed by the object $x^{(0)}$.

$$\hat{x}^{(0)} = \underset{x_i \in \mathcal{D}}{\operatorname{argmin}} \left(\sum_{x_k} d(x_i, x_k) \right) \qquad [2.36]$$

$$\mathcal{P}^{(t=0)} = \{\hat{x}^{(0)}\} \qquad [2.37]$$

2. Select a candidate object outside the current prototype set, $x_i \notin \mathcal{P}$. A quality score $S(x_i)$ is computed for the selected object. It measures the distances between all other objects and x_i in relation to their distances to $\mathcal{P}$:

$$S(x_i) = \sum_k C_{ik} \quad [2.38]$$

$$C_{ik} = \max(0, d_{\min}(x_k, \mathcal{P}) - d(x_k, x_i)) \quad [2.39]$$

where $d_{\min}(x_k, \mathcal{P})$ denotes the distance of x_k to the closest prototype in $\mathcal{P}$.

3. Repeat Step 2 to compute the scores $S(x_i)$ for all candidate objects outside $\mathcal{P}$. At iteration t, select the object $\hat{x}(t)$ with the maximum score $S(x_i)$ and update the set of prototypes with $\hat{x}$:

$$\hat{x}(t) = \underset{x_i \notin \mathcal{P}}{\operatorname{argmax}}\, S(x_i) \quad [2.40]$$

$$\mathcal{P} = \mathcal{P} \cup \{\hat{x}(t)\} \quad [2.41]$$

4. Repeat Steps 2 and 3 while incrementally adding new prototypes to $\mathcal{P}$ until k prototypes are found, i.e. $|\mathcal{P}| = k$.

2.4.3.3.2. Swap phase

In the second phase of the algorithm, all possible exchanges, or *swaps* between the prototypes and the rest of objects, are evaluated to minimize the objective function. If a swap ($x_i \in \mathcal{P} \leftrightarrow x_j \in \mathcal{P}$) is carried out, the object x_i is removed from the set of prototypes in favor of x_j, which is attached to $\mathcal{P}$.

1. For each possible swap in the form ($x_i \in \mathcal{P} \leftrightarrow x_j \notin \mathcal{P}$), the "goodness" of the swap is evaluated by a score $T(x_i, x_j)$, calculated as the sum of individual contributions $C_{i,j,k}$ for all other objects, $x_k \notin \mathcal{P}$:

$$T(x_i, x_j) = \sum_{x_k} C_{ijk} \quad [2.42]$$

$$C_{ijk} = \begin{cases} 0 & \text{if } d(x_k, x_i) > d_{\min}(x_k, \mathcal{P} - \{x_i\}) \\ & \text{and } d(x_k, x_j) > d_{\min}(x_k, \mathcal{P} - \{x_i\}) \\ d(x_k, x_j) - d(x_k, x_i) & \text{if } d(x_k, x_i) = d_{\min}(x_k, \mathcal{P}) \\ & \text{and } d(x_k, x_j) < d_{\min}(x_k, \mathcal{P} - \{x_i\}) \\ d(x_k, \mathcal{P} - \{x_i\}) - d(x_k, x_i) & \text{if } d(x_k, x_i) = d_{\min}(x_k, \mathcal{P}) \\ & \text{and } d(x_k, x_j) \geq d_{\min}(x_k, \mathcal{P} - \{x_i\}) \\ d(x_k, x_j) - d_{\min}(x_k, \mathcal{P}) & \text{if } d(x_k, x_i) > d_{\min}(x_k, \mathcal{P} - x_i) \\ & \text{and } d(x_k, x_j) < d_{\min}(x_k, \mathcal{P}) \end{cases} \quad [2.43]$$

where $d_{\min}(x_k, \mathcal{P})$ denotes the minimum distance of x_i to any of the objects in $\mathcal{P}$.

2. Select the pair with the minimum score $T(x_i, x_j)$:

$$(\hat{x_a}, \hat{x_b}) = \underset{x_i \in \mathcal{P}, x_j \notin \mathcal{P}}{\operatorname{argmin}} T(x_i, x_j) \quad [2.44]$$

3. Stop if $T(\hat{x_a}, \hat{x_b}) \leq 0$. Otherwise, carry out the swap $(\hat{x_a} \leftrightarrow \hat{x_b})$ and repeat from Step 1 to evaluate further potential swaps.

2.4.3.3.3. Advantages and drawbacks

In comparison to the k-means algorithm, the PAM reaches the global optimum regardless of the medoids initialization. However, this is achieved at the expense of higher computational cost. Moreover, since the optimum criterion is to minimize the total sum of distances instead of squared distances, the PAM algorithm is more robust against outliers. As in k-means, the PAM extracts Voronoi regions around the medoid objects and may thus fail to detect non-convex clusters.

2.4.3.4. *Self-organizing maps*

The self-organizing map (SOM) is a popular tool for exploratory data analysis, devised by Teuvo Kohonen [KOH 01, KOH 90, VES 00]. Kohonen's model belongs to the category of algorithmic approaches inspired by biological processes. The SOM principle shows a direct analogy

with the logical perception of different input stimuli by the human brain. Numerous studies in the field of neurobiology have ascertained how the different sensorial stimuli, in the form of multi-dimensional signals, are projected in specific regions of the cerebral cortex. It has been observed that, within each region, the information is mapped into certain topological structures, which are typically configured in linear of planar (bi-dimensional) maps. This peculiar characteristic of the brain is commonly referred to as the "topology preserving" property and was exploited by Kohonen in the SOM model.

The basic SOM model can be observed in Figure 2.5. In contrast to the previous generative algorithms, the set of prototypes $\mathcal{P}$ is organized in a topology (ordered map) of a lower dimensionality m than the input dataset. Following the typical structures in the brain, such maps are usually one or two dimensional. The SOM prototypes can thus be considered as the nodes in the ordered map, also referred to as neurons. In compliance with the topology preserving condition, the ordering of the nodes in the map (Kohonen layer) is fixed at all stages of the algorithm. In addition, each node p_i is associated with a weight vector $w_i \in \mathcal{R}^n$ that corresponds to any point in the input data space. In a learning phase, the nodes' weights are iteratively updated, until the final prototype weights reflect the input data distribution. In other words, the distribution of the final prototype weights in the input layer is the most representative for the input data distribution.

Initially, the prototype weights are assigned random values. At each iteration t, an input vector from the dataset is selected $x(t)$. The prototype p_k in the Kohonen layer with the closest weight vector to $x(t)$ is then selected as the winning prototype. In SOM models, the winning prototype is referred to as the *best matching unit* (BMU).

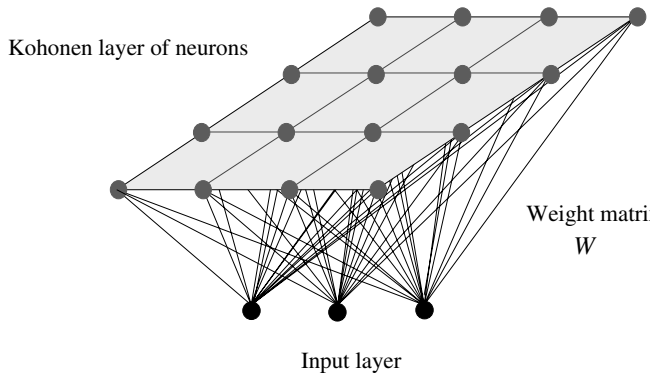


Figure 2.5. *Self-organizing maps (SOM). Example of neuron and input layers. The model's prototypes or neurons are distributed in an ordered map, typically linear or planar topologies. Each neuron is associated with a weight vector, in such a way that each point in the data space is mapped to a neuron and vice versa. While the values of neuron weights keep constantly adapting during the algorithm, the neuron position in the map is always fixed (topology preserving)*

The weight vectors of the BMU (w_k) and its neighbors are displaced toward the direction of the input element $x(t)$, using the weight update rule:

$$w_i(t+1) = w_i(t) + \alpha(t)\, h_{k,i}\, (x(t) - w_i(t)) \qquad [2.45]$$

where $\alpha(t)$ denotes the SOM learning rate, which decreases monotonically with each iteration. $h_{k,i}$ is a spatial neighborhood function that determines the degree to which the input vector $x(t)$ "attracts" the BMU's neighbor neurons in the Kohonen layer. At an attempt to emulate the *biological lateral interaction* between adjacent neurons, a Gaussian function is frequently selected for the neighborhood function $h_{k,i}$:

$$h_{k,i} = \exp\left(\frac{-||r_i - r_k||^2}{\sigma(t)^2}\right) \qquad [2.46]$$

where r_i and r_k denote m-dimensional vectors with the spatial coordinates of the nodes k (BMU) and i in the map. As with

the learning rate, the standard deviation σ in equation [2.46] is also a decreasing function with the number of iterations t. Thus, during the learning progress, the map becomes more "specialized" by increasing the spatial resolution of the neighborhood function.

The steps of the SOM learning algorithm are as follows:

1. Given an input finite dataset $\mathcal{D}$, and a selected topology, initialize the codebook of prototypes $\mathcal{P}(t=0) = \{p_1, p_2, \ldots, p_k\}$. If a random initialization is selected, each jth prototype variable takes k random values uniformly distributed in the original range of the corresponding variable in $\mathcal{D}$.

2. Select an input pattern $x(t)$ from the finite dataset $\mathcal{D}$. Calculate the distance (typically, the Euclidean distance metric is used) of the input pattern to all prototypes in $\mathcal{P}$. The winner prototype – BMU – is identified as :

$$\text{BMU}(t) = \underset{p_i \in \mathcal{P}}{\text{argmin}}\{||x(t) - w_i||\} \qquad [2.47]$$

3. Update the prototype weights of the BMU and its spatial node neighbors in the map toward the direction of the pattern $x(t)$, according to the weight update rule (equation [2.45]).

4. Stop if the maximum number of iterations is not exceeded $(t < maxiter)$. Otherwise repeat from Step 2 with $t = t + 1$.

To obtain final prototype weights which are good representatives of the input data distribution, an adequate choice for the maximum number of iterations $maxiter$ is a multiple of the input dataset cardinality $maxiter := N \cdot |\mathcal{D}|$, so that each data element $x_i \in D$ is exactly presented N times to the SOM learning algorithm.

The multiplier factor N is also referred to as the number of *cycles* or *epochs*. Figure 2.6 shows an example of the prototype weights adaptation (input layer) on the mixture of five Gaussians over 100,000 iteration. Grey dots depict

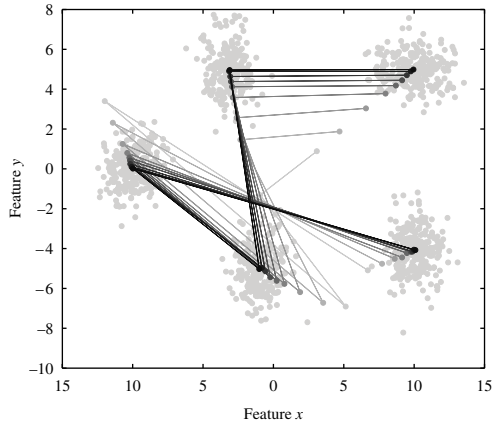


Figure 2.6. *Illustration of the SOM learning iterations on a mixture of five spherical Gaussians in two dimensions. The nodes, which in this example are projected into the data space, are depicted as filled circles. The lines connecting the nodes indicate adjacent nodes in the map (Kohonen layer). The lines and nodes' colors, from light to dark, are used to indicate the sequence of iterations, starting from the random initialization, until the convergence state, after 100,000 iterations*

the nodes distribution at different stages of the algorithms, whereas black dots depict the dinal convergence positions. Figure 2.7 shows another example using a 10x10 sheet topology, after the initialization (Figure 2.7(a)) and after 10,000 iterations (Figure 2.7(b)). The corresponding U-matrices are also depicted (Figures 2.7(c) and 2.7(d)).

2.4.3.4.1. Advantages and drawbacks

The main advantage of Kohonen SOMs relies on their capability to fit any type of distribution. The SOMs for clustering have several possibilities. If the number of clusters, k, is known, a topology (e.g. linear) with k nodes can be selected. At the convergence state, a cluster can be considered as the patterns which share the same BMU unit. Alternatively, a larger topology may be selected in combination with any k-clustering approach to partition the

nodes in the Kohonen layer. Finally, if the number of clusters is unknown, the SOM can be used as a visualization tool. The task to recognize the number of clusters is left to the users who analyze the so-called U-matrix (as in Figure 2.7(d)).

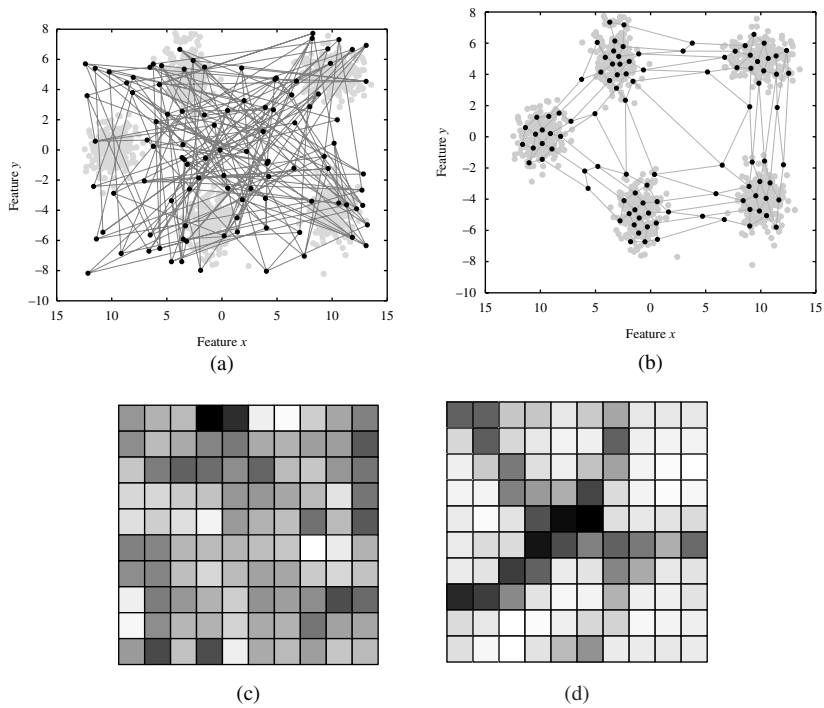


Figure 2.7. *Illustration of the SOM learning iterations and U-matrix on a mixture of five spherical Gaussians in two dimensions. A 10*10 rectangular sheet topology is selected. Top plots show the data and map nodes projected in the data space. (a) is referred to the initialization state, while (b) shows the organized map after 10,000 iterations. Bottom plots show nodes in the Kohonen layer (output space). The different nodes' colors in the gray scale indicate similarities. For example, if two nodes have a high pair-wise similarity, these nodes have similar colors (close to white), while dark colors represent distant nodes. In the convergence state, approximately five regions with light colors can be noticed, corresponding to the five existing clusters*

However, this task has some limitations. For example, although the SOM does not require the number of clusters, the final distribution of nodes – as well as the U-Matrix – also depends on the topology and the map size, which has to be determined beforehand. It is also known that, to obtain accurate results, large maps are usually required, which increases the algorithm complexity. Finally, the analysis of the U-matrix to determine the number of clusters has the limitations of a subjective task. Such decision may vary from user to user, specially for close clusters or for large ks.

2.4.4. *Density-based clustering*

In contrast to the classical hierarchical and divisive approaches, in which clusters arise based on the objects' distances, density algorithms build the clusters based on a density concept [EST 96, SAN 98]. This means that clusters are located in regions with roughly homogeneous densities.

The most popular approach to density clustering is probably the *Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise* (DBSCAN), proposed in [EST 96]. The density concept in the DBSCAN algorithm is indicated by the input parameters (minpts and eps), which denote the minimum number of points to be enclosed in a neighborhood of radius eps, respectively.

In the DBSCAN algorithm, two types of objects are distinguished: cluster and noise. To formulate the notion of cluster in DBSCAN, the concepts of *direct density reachability* and *density connection* were introduced, which are illustrated in Figure 2.8.

2.4.4.1. *Direct density reachability*

Figure 2.8(a) shows an hypothetical spatial dataset, as well as the ϵ neighborhood of the point p_1. Moreover, assuming a

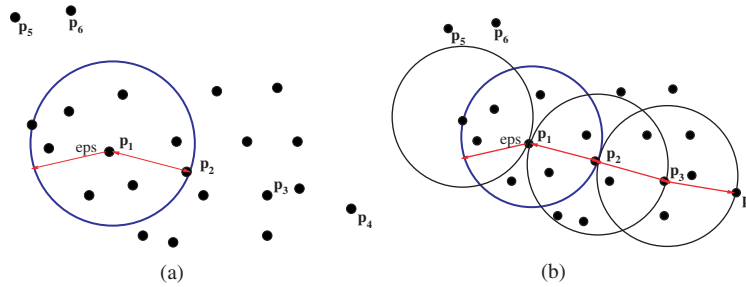


Figure 2.8. *Illustration of the concepts of density reachability and density used in the DBSCAN algorithm. The concept of direct density reachability is depicted in (a) for an hypothetical spatial dataset, with the eps neigborhood of the point p_1. (b) shows the concept of density reachability and density connection*

particular value of $minpts = 4$, the point p_1 is said to be a *core point*, and all points in its ϵ neighborhood, such as p_2, are *density reachable* from p_1.

2.4.4.2. *Density reachability*

Figure 2.8(b) shows the concept of *density reachability* and *density connection* with respect to $minpts$ and ϵ. For example, p_3 is not directly reachable from p_1 since it falls outside the eps neighborhood of p_1. However, it is density reachable from p_1, because it is possible to find one or more points p_i directly reachable from p_{i-1}, starting at p_3 and ending at p_1. In this particular case, p_3 is density reachable from p_1 because it is directly reachable from p_2, which in turn is directly reachable from p_1. Likewise, we can argue that p_4 is also density reachable from p_1 through p_3 and p_2.

2.4.4.3. *Density connection*

Furthermore, p_3 and p_1 are also said to be *density connected*, because there exists at least one intermediate point (in this case p_2), from which both p_1 and p_3 are density reachable. In addition, p_4 is density connected to $p1$ through p_3

or p_2. It should be noted that density connection does always imply density reachability. For example, in Figure 2.8(b), although p_1 is density connected to p_4, it is not reachable from p_4, since p_4 is not a core point.

2.4.4.4. *Border points*

In Figure 2.8(b), p_4 is of a different type to p_3, p_2, and p_1. The latter objects are called *core points* because we can always find a number of points greater than $minpts$ in their neighborhood. In contrast, the point p_4 is called a *border point* because its neighborhood contains less than $minpts = 4$.

2.4.4.5. *Noise points*

It should be noted that the concept of border point implies that the point is density reachable from at least one point, as with p_4. In contrast, the points p_5 and p_6, which also contain less than $minpts$ in their eps neighborhoods, are not density reachable from any point in the dataset. These are thus located in a non-dense area and are treated as *noise* points.

2.4.4.6. *DBSCAN algorithm*

Each cluster found by the DBSCAN algorithm is thus composed of a set of density connected points. Assume $\mathcal{X}$ denotes the input dataset, $\mathcal{U}$ and $\mathcal{N}$ the sets of unclassified and noise points, respectively, and $\mathcal{C}$ the final set of clusters, the DBSCAN algorithm can be described in the following steps:

1. Initially, the set of unclassified objects is equal to the input dataset, and the set of noise patterns is the empty set:

$$\mathcal{U} = \mathcal{X}, \quad \mathcal{N} = \emptyset \tag{2.48}$$

2. Select a non-classified object $x_i \in \mathcal{U}$ around which a new potential cluster is to be build, C_i:

$$C_i = x_i, \quad \mathcal{U} = \mathcal{U} - \{x_i\} \tag{2.49}$$

3. Calculate the set of data elements $\mathcal{S}$ with lower distance to x_i than eps.

4. If $|S| < minpts$, x_i is not a core object. The candidate cluster C_i is discarded.

$$C_i = \emptyset, \quad \mathcal{U} = \mathcal{U} \cup \{x_i\} \tag{2.50}$$

Go to Step 1 to explore for other initial objects.

5. If $|S| \geq minpts$, all objects in $|S|$ are direct density reachable from x_i. The new cluster C_i will thus contain at least the objects in $\mathcal{S}$. Remove the objects from the set of unclassified patterns and go to Step 5

$$\mathcal{U} = \mathcal{U} - \{\mathcal{S}\} \tag{2.51}$$

6. Select a random object from $\mathcal{S}$, x', and calculate the set of points $\mathcal{S}'$ which are less or equal distance than eps from x'. Remove x' from the set $\mathcal{S}$ of points to be further visited, adding x' to the growing cluster C_i:

$$C_i = C_i \cup \{x'\}, \quad \mathcal{S} = \mathcal{S} - \{x'\} \tag{2.52}$$

7. If $|\mathcal{S}'| > minpts$, the chain of density reachable points from x can be extended by exploring the elements of $\mathcal{S}'$. Thus, add these elements to the set of points to be visited $\mathcal{S}$ and remove them from the set of unclassified objects:

$$\mathcal{S} = \mathcal{S} \cup \mathcal{S}', \quad \mathcal{U} = \mathcal{U} - \mathcal{S}' \tag{2.53}$$

8. Repeat Steps 6 and 7 until $\mathcal{S} = \emptyset$. This means that the chain of density reachable objects from x has been completed, and thus, no more points need to be further visited. Add the new cluster C_i to the set of clusters:

$$\mathcal{C} = \mathcal{C} \cup C_i \tag{2.54}$$

9. Repeat from Step 1 to search for more possible clusters, until no further data elements can be found in $\mathcal{U}$ with more than $minpts$ in their eps neighborhoods. In this case, the remaining unclassified objects in $\mathcal{U}$ compose the set of noise patterns $\mathcal{N}$ ($\mathcal{N} = \mathcal{U}$).

2.4.4.6.1. Advantages and drawbacks

In contrast to the previous algorithms, the DBSCAN algorithm is based on density parameters instead of distances. Thereby, DBSCAN is not only able to extract globular clusters but also can identify cluster of different shapes and/or sizes. One limitation of the DBSCAN algorithm is, however, related to the a-priory determination of the minimum cluster density to be extracted (input parameters $minpts$ and eps).

2.4.5. *Graph-based clustering*

In a graph-based clustering, the data objects are arranged as vertices $\mathcal{V}$ of a graph $G(\mathcal{V}, \mathcal{E})$, with a set of edges E. Different clustering algorithms exist based on graphs, such as Metis, Hmetis [KAR 98], or mincut [FEN 09]. In this book, the pole-based overlapping algorithm has been considered. This approach extracts the final clusters without any *a priori* parameter.

2.4.5.1. *Pole-based overlapping clustering*

The pole-based overlapping clustering (PoBOC) is an overlapping, graph-based clustering technique proposed by Cleizou [CLE 04a, CLE 04b]. The algorithm takes the matrix of object dissimilarities as single input and builds the output clusters in four main steps: (i) definition of dissimilarity graph, (ii) construction of poles, (iii) pole restriction, and (iv) affectation of objects to poles.

2.4.5.1.1. Definition of a dissimilarity graph

Let $\mathcal{X}$ denote the set of objects in the dataset and D the dissimilarity matrix, computed over $\mathcal{X}$. The mean distance of an object x to all other objects in $\mathcal{X}$, $d_{\text{mean}}(x, \mathcal{X})$, is formulated as:

$$d_{\text{mean}}(x; X) = \frac{1}{n-1} \sum_{xi \in X} D_{x,x_i} \qquad [2.55]$$

The dissimilarity graph $G(\mathcal{X}, \mathcal{E}, D)$ is then specified by (i) the dissimilarity matrix D and (ii) the data points or vertices, $\mathcal{X}$, and a set of edges $\mathcal{E}$ between all pairs of vertices (x_i, x_j) corresponding to mutual neighbor points.

DEFINITION 2.1.– Neighborhood of a point x. *The neighborhood of a point x, denoted by $N(x)$ is composed of all points of $\mathcal{X}$ whose dissimilarity to the point is smaller than the mean distance of the object x to all other objects in $\mathcal{X}$ ($d_{\text{mean}}(x; \mathcal{X})$):*

$$N(x) = \{x_j \in \mathcal{X} | D_{x_j,x} < d_{\text{mean}}(x; \mathcal{X})\} \quad [2.56]$$

DEFINITION 2.2.– Mutual neighborhoods. *Two points (x_i, x_j) are mutual neighbors, and thus connected by an edge in $\mathcal{E}$, if each one belongs to the neighborhood of the other:*

$$(x_i, x_j) \in \mathcal{E} \leftrightarrow x_i \in N(x_j); x_j \in N(x_i) \quad [2.57]$$

2.4.5.1.2. Pole construction

This procedure builds incrementally a set of poles $\mathcal{P} = \{P_1, P_2 \ldots, P_k\}$ over $\mathcal{X}$ based on the dissimilarity matrix D and the dissimilarity graph $G(\mathcal{X}, \mathcal{E}, D)$. Let $\mathcal{O}$ denote the cumulated set of objects that belong to any of the extracted poles up to the current state (initially the empty set).

The poles are grown from initial points $\hat{x_i}$, which are the points with maximum mean distance to the cumulated set of poles $\mathcal{O}$. Initially, the object $\hat{x_0}$ with maximum distance to $\mathcal{X}$ is selected:

$$\hat{x_0} = \underset{x \in \mathcal{X}}{\text{argmax}}(d_{\text{mean}}(x, \mathcal{X})) \quad [2.58]$$

Each P_i pole is then grown from the corresponding initial object

$$\hat{x}_i = \underset{x \in \mathcal{X} \setminus \mathcal{O}}{\operatorname{argmax}}(d_{\text{mean}}(x, \mathcal{O})) \quad [2.59]$$

in such a way that all the pole members are mutual neighbors of each other. This is implemented in the build-pole procedure:

Algorithm 2.1 Build-pole ($\hat{x}$, $G(\mathcal{X}, \mathcal{E}, D)$)

Input: initial point $\hat{x}$, dissimilarity graph $G(\mathcal{X}, \mathcal{E}, D)$
Output: pole built around $\hat{x}$, P
Initialize: $P = \hat{x}$
Obtain neighborhood of P:
$N(P) = \{x \in \mathcal{X} | \forall x_i \in P, (x, x_i) \in \mathcal{E}\}$
while $N(P) \neq \emptyset$ **do**
  attach the object x to P such that:
  $x \in N(P)$ and $x = \operatorname{argmax}_{x_i \in N(P)} d_{\text{mean}}(x_i, P)$
  Update $N(P)$
end while
Return P

The selection of the initial object $\hat{x}_i$ and the construction of the corresponding pole P_i are iteratively repeated until all objects in the dataset are contained in any of the poles, $\mathcal{O} = \mathcal{X}$, or no initial object can be found which is sufficiently distant from the set of poles.

2.4.5.1.3. Pole restriction

After the pole construction, overlapping objects may be obtained. These are objects that simultaneously belong to two or more poles. Overlapping compose the residual set $\mathcal{R}$. The pole restriction procedure removes the residual objects from the original poles, resulting in a new set of reduced, non-overlapping poles $\tilde{\mathcal{P}} = \mathcal{P} - \mathcal{R}$.

2.4.6. *Affectation stage*

The set of residual objects $\mathcal{R}$ obtained at the pole restriction stage require some post-processing strategy to be re-allocated into one or more of the restricted poles. This re-allocation of objects in PoBOC is called affectation. First, the membership of each object x to each $\tilde{P}_i$-restricted pole, $u(x, \tilde{P}_i)$, is computed as:

$$u(x, \tilde{P}_i) = 1 - \frac{d_{\text{mean}}(x, \tilde{P}_i)}{D_{\max}} \qquad [2.60]$$

Next, the objects are affected to one or more poles. In a single-affectation approach, each object x is assigned to the pole maximizing the membership $u(x, \tilde{P}_j)$. In a multi-affectation approach, the object is affected to the poles whose memberships are greater than some reference values given by a linear approximation on the set of object memberships, sorted in decreasing order.

2.4.6.1. *Advantages and drawbacks*

The main advantage of the PoBOC algorithm is that no-apriori knowledge is required to obtain the cluster solution. The algorithm is able to extract the clusters with no input parameter. However, one limitation of PoBOC is related to the internal parameter used by the algorithm to build the dissimilarity graph, namely, the average object distances. Due to this internal parameter, PoBOC may fail to detect clusters in hierarchies or with non-homogeneneous inter-cluster distances. A solution to this limitation is provided in Chapter 5.

2.5. Applications of cluster analysis

One outstanding characteristic of cluster analysis is the multi-disciplinary application of the clustering techniques.

This fact is primarily related to the diverse nature of the data which can be subjected to exploratory analysis. The only requisite for the input dataset is the multivariate formulation of the entities to be clustered, in terms of a set of properties or measurements (variables or features).

2.5.1. *Image segmentation*

Image segmentation techniques divide an input image into a set of non-overlapping regions that are roughly homogeneous based on features of interest, such as texture, color, intensity, and sometimes, the pixels' positions [LO 07, OUY 09]. Image segmentation is used for multiple purposes, typically as a pre-processing component that allows us to recognize small segments to be posteriorly labeled or classified. Different approaches for image segmentation can be encountered in the image processing and computer vision literature, including watershed techniques using mathematical morphology [YE 09], edge finding [YAN 09], region growing [MAN 00], artificial neural networks [MUK 08], thresholding, and *clustering* [DES 09]. According to Jain, thresholding is a particular case of clustering in a binary form (using only two target clusters).

One example application of image segmentation is for medical images. Clustering – among the rest of techniques – can be used to analyze different image modalities: magnetic resonance images (MRI) of the brain, computed tomographies (CT), chest radiographies or digital mammographies, among others, to distinguish different types of tissue or localize suspicious elements – tumors – for posterior classification or diagnosis.

A second example of segmentation using clustering is shown in Figure 2.9. The plots show two-digital slide brain images (a,d), segmented using the k-means (b,e), and EM

(c,f) algorithms. Among the output clusters, different brain tissues can be distinguished (white mass, gray mass, and cerebrospinal fluid).

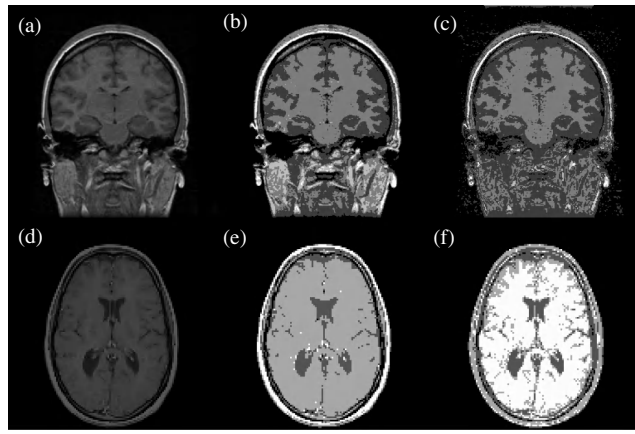


Figure 2.9. *Example of clustering for image segmentation in two digital slide brain images (a,d), segmented using k-means (b,e) and EM (c,f). Among the output clusters, different brain tissues can be distinguished (white mass, gray mass, and cerebrospinal fluid)*

Another application of clustering-based image segmentation is for content-based image retrieval in which users query a database of images at the level of objects, such as traffic signs, cars, faces, and buildings. The objective of segmentation for content-based image retrieval is to identify such regions in the image that possibly correspond to objects or parts of them [PAR 07]. Among the different segmentation modalities for object-based image retrieval, cluster analysis can be particularly applied to the so-called 'Blobworld' representation. The *blobs* are defined as the segmented regions, typically extracted via clustering, where the segmentation features (color, texture, etc.) remain approximately homogeneous. Each blob is finally represented in terms of its intrinsic descriptors, such as the dominant color and mean texture. When the users want to initiate

the search for a particular object, they are asked to present an image somehow related to the object of interest. The segmentation algorithm is then applied to the query image, and its output is presented back to the user, together with the set of descriptors associated with the extracted blobs. The user is now able to specify the blob or a combination of them to be retrieved from the database or refine the query by some relevant descriptor values. It should be noted that, when the query response is obtained, the user has access to a complete blob information related to the retrieved segmented images, which also allows for more refined searches, in comparison with other image retrieval strategies. The blobword representation was introduced by [CAR 02], who used the EM algorithm to segment images based on the aforementioned features (color, texture, and spatial characteristics). Finally, clustering analysis has also been recently applied to face recognition, even though clustering is not a common practice in segmentation for face detection nor face feature extraction. Nonetheless, in [CHE 09], a hierarchical agglomerative clustering algorithm was applied in conjunction with a statistical classifier (support vector machines (SVMs)) to improve the efficiency and effectiveness of the recognition task. In particular, the agglomerative algorithm was used to group similar faces in the training set, followed by a two-step SVM using Bayesian modeling of face differences. Thereby, a manageable number of SVM hyperplanes was attained, despite the large number of faces (classes) in the database.

2.5.2. *Molecular biology*

During the past decade, the analysis of deoxyribonucleic acid (DNA) microarray data, which encodes the expression levels of a number of genes in cells from different tissues or organisms, has become a major common field of study in the areas of molecular biology and computer science.

2.5.2.1. *Biological considerations*

Gene expression or DNA microarray data is obtained by means of DNA microarray assays in which the activity of thousands of genes is monitored simultaneously [THE 09].

In a microarray experiment [CHU 02], a DNA microarray (DNA chip) is prepared. Basically, a DNA chip is a substrate slide, like glass or nylon. It contains an array of spots, pre-filled with oligonucleotide probes for a series of genes to be monitored. First, two cell samples are extracted from different populations to compare. For example, if the goal is to analyze the expression of responsible genes for a certain tumor, it is common to select a sample from a tumor cell and a second one from a normal cell. The mRNA from both cells is then extracted and marked with two different fluorescent dyes, typically blue and red. Next, the microarray chip is flooded with the mixed mRNAs, which hybridize with the spot nucleotides. By analyzing the microarray outcomes under a laser microscope with different wavelengths – read and green lights – two digital images are generated, where the spots hybridized with each colored mRNA can be detected. Finally, for each spot, the relative abundance of the gene expression in the first population (e.g. tumor cells) with respect to the second (e.g. normal cells) can be measured by,

$$Is = \log\left(\frac{I1(\text{red})}{I2(\text{green})}\right) \qquad [2.61]$$

Typically, microarray data are arranged as a n by m matrix of expression levels, $\mathbf{S}$, in which rows are related to genes and columns to different experimental samples (also called

	D1	D2	D3	D4	D5
t1: Algorithm	0	1	0	0	1
t2: Analysis	0	0	1	0	0
t3: Application	1	0	0	0	0
t4: Cluster	0	1	1	1	1
t5: Data	1	1	0	0	0
t6: Document	0	0	0	1	1
t7: Mine	1	0	0	0	0
t8: Text	0	0	0	1	0

Table 2.2. *Example of Boolean representation of documents with an hypothetical set of book titles*

conditions) (Table 2.2):

$$\mathbf{S} = \begin{pmatrix} s_{11} & \dots & s_{1j} & \dots & s_{1m} \\ \vdots & \ddots & \vdots & & \vdots \\ s_{i1} & \dots & s_{ij} & \dots & s_{im} \\ \vdots & & \vdots & & \ddots \\ s_{n1} & \dots & s_{nj} & \dots & s_{nm} \end{pmatrix}$$

The elements s_{ij} are real values denoting the relative abundance of the gene's mRNA i under condition (sample) j [BRO 99a].

The fundamental objectives of DNA microarray data analysis are to identify genes involved in certain biological processes, such as diseases, and specially, cancers, or to classify/identify different types and subtypes of cancer tissues on the basis of the gene activity profiles in tumor cells [HAS 00, BRO 99b].

The advances in DNA microarray technologies have made possible to obtain gene expression measurements for thousands of genes. Last generation sequencing

techniques have even succeeded in generating complete genomic sequences. As is commonly accepted, the human genome is composed of approximately 30,000 genes (cite in gene selection for microarray data). However, only small groups of genes are expected to participate in active cellular processes of interest. For example, around 57 genes have been found to be responsible for breast cancers and around 158 for pancreatic cancers. Given the huge amounts of gene expression measurements collected from DNA microarray experiments, clustering techniques have been broadly applied to the efficient analysis, organization, and understanding of microarray tables.

One of the first references for clustering microarray data was the paper by [EIS 98]. They illustrated the usefulness of hierarchical clustering schemes in grouping together genes with similar functions, applied to gene expression profiles in the budding yeast *Saccharomyces cerevisiae*. A graphical representation of the clustered data was also proposed as an intuitive tool to facilitate the interpretation of the clustering results. Tavauoie *et al*. [BAG 06] used partitional k-means to extract groups of functional genes from the budding yeast. In this case, the samples were obtained at 15 different time points corresponding to different phases of the cells' cycles. SOMs have been also widely applied to DNA microarray data. For example, [TAM 99] used spherical SOMs (sSOMs) for clustering genes in mouse brain tumor cell lines (Figure 2.10).

The aforementioned contributions are examples of *gene-based clustering*, where the objective is, exclusively, to find groups of genes that share similar properties through different samples. A high number of research works on DNA microarray data analysis can be circumscribed in this category, considered as the standard approach for gene expression data.

As mentioned earlier, a second purpose of microarray data analysis is to identify groups of samples that share similar

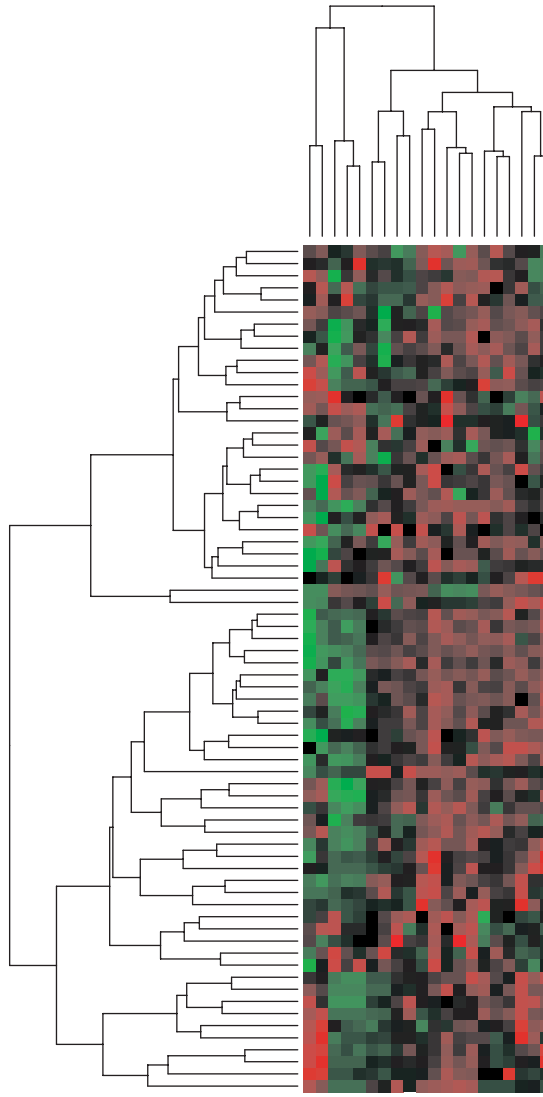


Figure 2.10. *Example of a gene expression matrix and hierarchical trees obtained by hierarchical average clustering on the rows (genes) and columns (samples)*

gene expression patterns. This approach, called *sample-based clustering*, has gathered notably fewer contributions in comparison with gene-based clustering, specially due to the cost of microarray chips used in different experiments (samples). More recently, new gene clustering approaches have been implemented, which aim at clustering genes and samples (rows and columns of the gene expression matrix) simultaneously. These approaches are commonly referred to as *subspace clustering* or *bi-clustering* [CHE 00, GET 00].

2.5.3. *Information retrieval and document clustering*

The term information retrieval (IR) refers to the search of information in unstructured databases (typically collections of documents) [RIJ 79, MAN 08]. A canonical example is the search of documents from the World Wide Web (www) [WUL 97]. The interfaces and search tools to retrieve relevant documents from the Web upon a user query are called Web crawlers or Web search engines (Google, Yahoo, Altavista, etc.). Other classical applications of IR is the search for books in digital libraries [JON 07, CHU 99]. Essentially, the users query the document database by typing one or more keywords that define the topic of interest. Some Web search engines support more complex queries also with different logical operators AND, OR, etc. The IR system then tries to match the documents in the database against the query words and retrieve the set of documents ranked as more relevant for the user's query.

More recently, due to the progress of automatic speech recognition (ASR) [RAB 93], the *vector-based call routing* approach has emerged as a new, speech-based modality of IR for call routing applications [CHU 99, SAL 71]. The goal is to redirect the user call to the most relevant module destination according to the caller utterance. The possible call destinations are modeled by "virtual documents" (created during training phases of the system) and the callers' query

utterances are transcribed to textual words by the ASR. Thus, the task is analogous to a classical IR problem and can be resolved using IR tools and procedures.

One of the main tasks performed by IR systems is the representation of documents and queries that allows us to quantify the semantic "similarities" between documents or between documents and queries. Typically, documents are represented on the basis of word descriptors. The set of term descriptors is known as *vocabulary* or *lexicon* $\mathcal{W}$. It is commonly extracted through certain pre-processing steps applied to a collection of (training) documents. These are described in the following sections.

2.5.3.1. *Document pre-processing*

Typically, the pre-processing task is accomplished in three main steps: feature selection, stop-word filtering, and surface form reduction (word lemmatizing or stemming).

2.5.3.1.1. Word selection

The aim of *feature selection* is to reduce the dimensionality of the document space by identifying and removing uninformative, non-discriminative words, while retaining only the informative words. The main characteristic of uninformative words is their roughly uniform distribution over different document categories, in comparison with discriminative words. Word selection is usually addressed by ranking all words in the documents according to a certain relevance score and filtering out words that do not exceed a given relevance threshold. Some word scoring metrics are, for example, IDF, TFIDF, or mutual information [YAN 97,FOR 03, GUY 03].

Owing to the "noisy" character of irrelevant words, not only a dimensionality reduction is attained by means of word selection, but also the effectiveness of IR systems and document classification can be substantially increased.

Another strategy to reduce the dimensionality of the document space is to feature extraction. A common approach applied to document collections is known as "latent semantic analysis (LSA)" [LAN 98, DEE 90]. Basically, LSA performs a singular vector decomposition (SVD) to the term document matrix. As a result, the initial word and document spaces are mapped into two new spaces of eigenvectors of lower dimensionality (**S** and **D**). The new dimension is determined by the non-zero eigenvalues in the matrix **V**. Feature transformation via latent semantic analysis can also help to remove the "noise" induced by words in the term document matrix due to certain lexical properties, namely, synonymy and polysemy. Finally, word feature clustering has been also applied as a feature extraction method for document classification [WUL 97].

2.5.3.1.2. Stop word filtering

In contrast to feature selection, stop word filtering is performed on the basis of manually pre-established lists, such as the SMART stop list [BLA 07, DRA 09]. Stop words are reportedly irrelevant for IR purposes as well as document clustering and classification, as they do not convey any semantic information to a document or text. Some examples of stop words are pronouns (*He, etc.*), prepositions (*of, for, the*), and, in natural language texts (such as the texts transcribed from speech by an ASR), words typical for spontaneous speech (*eh, ehm, uh*).

2.5.3.1.3. Word lemmatizing/stemming

This stage aims to reduce the normal, "surface" form of words into their "canonical" forms: *stems* or *lemmas*. The objective is to eliminate the morphological variability introduced by inflectional and/or derivative suffixes to retain the units of semantic meaning [POR 80,SCH 94]. According to different criteria, either word stemmers or lemmatizers can be applied to extract word stems or lemmas, respectively. A word

stemmer removes any kind of inflectional and derivative suffixes from the surface word form to extract the word stem. This procedure can end up in a different lexical category (also referred to as part of speech (PoS)) with respect to the initial word (for example, *workers* $\rightarrow$ *work*). In contrast, a word lemmatizer eliminates only infectional suffixes from the surface form of the word, e.g. plurals of nouns are reduced to singular forms and conjugated forms of verbs are reduced to the infinitive form. The PoS tag of the initial word is thus preserved. Lemmas are also known as the "dictionary form" of words, since it is the form in which all entries in a dictionary are presented.

The vocabulary W is finally composed of all distinct lemmas or stems observed in the pre-processed documents.

2.5.3.2. *Boolean model representation*

In a Boolean model [KLE 00, LAS 09], each document in the collection can be viewed as a Boolean vector whose components denote the presence or absence of the lexicon terms in the document. For example, imagine a library application where the search is for books relevant to a topic. The document dataset might consist of the following book titles (documents): E.g. D1 = *Data mining applications*; D2 = *Algorithms for clustering data*; D3 = *Clustering analysis*; D4 = *Clustering text documents*; and D5 = *Algorithms for document clustering*. If a word lemmatizer is applied and a unigram approach is selected, the resulting lexicon of terms, in alphabetical order, is $\{W\}$ = Algorithms, Analysis, Applications, Cluster, Data, Document, Mine, and Text. The Boolean representation of the set of documents can be observed in Table 2.2.

A *Boolean query*, Q, is then composed of terms and Boolean operators. The three standard Boolean operators are AND, OR and NOT. As an example, a possible query related to

		D1	**D2**	D3	D4	D5
	t1: Algorithm	0	1	0	0	1
AND	t4: Cluster	0	1	1	1	1
AND NOT	t6: Document	0	0	0	1	1
	Search result	0	**1**	0	0	0

Table 2.3. *Example of retrieval using the Boolean model*

the previous example could be $\mathcal{Q}$ = [Clustering AND NOT Documents AND Algorithms]. The result is obtained by computing the bitwise logical AND between the terms $t4$ (Cluster) and $t1$ (Algorithm) and the Boolean negative of the term $t6$ (Document) (see Table 2.3).

As can be noted, the search result to the Boolean query $\mathcal{Q}$ is the document (book title) D2 (*Algorithms for Clustering Data*). In Boolean retrieval, all *exact matches* are typically returned to the user, although there have been proposals for improving the presentation of results by ranking the documents according to the frequency of the query terms in the documents.

2.5.3.3. *Vector space model*

In the so-called vector-model representation [SAL 75, STE 00, PRI 03], used in IR, and the close areas document clustering or classification, a collection of documents is typically arranged as a matrix D and a query as a term vector Q:

$$\mathcal{D} = \begin{pmatrix} d_{11} & \dots & d_{1j} & \dots & d_{1m} \\ \vdots & \ddots & \vdots & & \vdots \\ d_{i1} & \dots & d_{ij} & \dots & d_{im} \\ \vdots & & \vdots & & \ddots \\ d_{n1} & \dots & d_{nj} & \dots & d_{nm} \end{pmatrix}$$

$$\mathcal{Q} = (q_1, \dots, q_i, \dots, q_m) \qquad [2.62]$$

The matrix $\mathcal{D}$ is also commonly referred to as the document term matrix, in which rows represent documents in the collection and columns refer to the lexicon terms $\{\mathcal{W}\}$. These two dimensions of the matrix (documents and words) are also referred to as the *document space* and the *word space*, respectively.

In a vector space model, the retrieved documents are typically ranked on the basis of their relevance to the user's query, which can be estimated by measuring the cosine similarity between their respective vectors $(\mathcal{Q}, \mathcal{D}[i])$. It should be noted that the cosine similarity metric is equivalent to the dot product similarity (equation [2.63]), if both $D[i]$ and $\mathcal{Q}$ vectors are normalized to have unit lengths:

$$Sim(\mathcal{Q}, \mathcal{D}[i]) = \sum_j q^j \cdot d_i^j \qquad [2.63]$$

2.5.3.4. *Term weighting*

In a similar way as in Boolean retrieval, the entries d_{ij} of the document term matrix can be just set to "0" or "1", denoting the presence of absence of the w_j word in the ith document in the collection. This Boolean notation can be also used for the query elements q_i. In such a case, the query is referred to as *Boolean query*, although this term should not be confused with the definition of Boolean queries used in Boolean retrieval models.

More sophisticated approaches apply *term weighting* methods to compute the scores d_{ij} and q_j, which denote the relative significance of each lexicon term w_j in the query and/or document vectors [SAL 88]. To estimate the importance of terms with respect to the document or query, a term weighting formulation consists typically of three different components.

2.5.3.4.1. Term frequency component

The basic idea behind the *term frequency* [RAM 00] component is that a term that occurs more often in a document is more important to describe the contents of the text. Different notations have been used in the IR literature for the term frequency component

$$\mathrm{TF}_{(a)}(w_j, d_i) = C(w_j, d_i) \quad [2.64]$$

$$\mathrm{TF}_{(b)}(w_j, d_i) = \frac{C(w_j, d_i)}{\sum_k C(w_j, d_i)} \quad [2.65]$$

$$\mathrm{TF}_{(c)}(w_j, d_i) = 0.5 + 0.5\frac{\mathrm{TF}_{(a)}}{\max(\mathrm{TF}_{(a)})} \quad [2.66]$$

where $C(w_j, d_i)$ denotes the counts of occurrences of the w_j word in the document. The first formulation, $\mathrm{TF}_{(a)}$, is also referred to as the *raw term frequency*, while the second and third, $\mathrm{TF}_{(b)}$ and $\mathrm{TF}_{(c)}$, are referred to as the *relative term frequency* and *augmented term frequency*, respectively.

2.5.3.4.2. Collection frequency component

The collection frequency factor captures the "informativeness" of words with respect to the complete collection of documents. The main assumption is that words that tend to occur in many documents are in general less discriminative for such documents than words occurring in a small number of documents and vice-versa. Different metrics have been proposed for the collection frequency component such as the Inverse document frequency (IDF) and the residual inverse document frequency (RIDF).

The most popular metric for the collection frequency component is the IDE [SAL 88], which measures, the inverse of the ratio of documents in which a word occur, with respect to the whole number of documents in the collection.

Two different formulations have been proposed for the IDE:

$$\text{IDF}_{(a)}(w_j) = \log \frac{N}{n} \quad [2.67]$$

$$\text{IDF}_{(b)}(w_j) = \log \frac{N-n}{n} \quad [2.68]$$

where the parameter N denotes the total number of documents in the collection (number of rows in matrix $\mathcal{D}$), and n refers to the number of documents in which the term w_j occurs. According to [SAL 88], the formulation $\text{IDF}_{(a)}$ is the raw IDF, while $\text{IDF}_{(b)}$ is referred to as the probabilistic IDF.

One variant of the IDF, particularly proven effective for automatic text summarization [MUR 08, MUR 07], is the RIDF. This metric, proposed in [MUR 08], quantifies the degree to which the IDF score (equation [2.67]) exceeds the expected value $\widehat{\text{IDF}}$, according to the Poisson model

$$\text{RIDF} = \text{IDF} - \widehat{\text{IDF}} \quad [2.69]$$

$$\widehat{\text{IDF}} = -\log(1 - \text{e}^{-\lambda_w}) \quad [2.70]$$

where λ_t denotes the parameter of the Poisson distribution, calculated as the average occurrence of the w term across all N documents: $\lambda_w = \sum_j n_j/N$. The main advantage of a RIDF metric is that rare terms [1] are not assigned high scores, in contrast to IDF.

2.5.3.4.3. Length normalization component

The *length normalization component* [SIN 96b] is intended to compensate for the natural bias placed by the previous weighting schemes towards longer documents, with a larger number of terms' occurrences. In other words, this component

1. Terms which occur in a small number of documents as a consequence of their low probability of occurrence in general.

equalizes the retrieval chances of documents of varying lengths. Some popular metrics for length normalization are the *cosine normalization, the byte length normalization, and the Pivoted length normalization.*

One of the most popular methods for document length normalization in vector-based IR is the well-known *cosine normalization*. This factor is defined as the Euclidean norm of the vector of raw term weights, calculated using any combination of the first and second term weighting components, described earlier.

$$Cosine\ Norm. = \sqrt{(w_1^2 + w_2^2 + \cdots + w_N^2)} \qquad [2.71]$$

By using the cosine normalization factor to both query and document vectors, the dot product of their corresponding vectors equals the cosine similarity between them. The components w_i in equation [2.71] can be equally applied to both document and query components, d_i^j and q^j.

However, a reported limitation of cosine normalization is related to the use of raw term weights (the term frequency and document frequency components) as part of the normalization component. Thus, any inadequate term score produced by such term weighting components has an impact in the normalization factor. As an example, [SIN 96a] observed how the presence of a rare term in the document, with characteristic high IDF sores, may entail a substantial increment in the document cosine normalization factor. As a result, the final weights assigned to the document terms might be much lower due to the impact of the rare term, and so would be the chances of retrieval of the document (using a dot product similarity to the query). To overcome this limitation, the authors proposed a normalization based on the number of bytes of the documents, also referred to as *byte size*

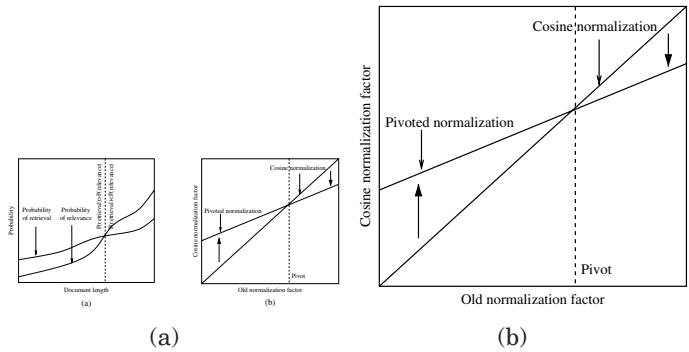


Figure 2.11. *Illustration of the pivoted length normalization. Smoothed curves of the likelihood of retrieval versus likelihood of relevance against document length. The cross point between the two likelihood curves is the pivot*

normalization [SIN 96a]

$$Byte\ Size\ Norm. = \text{byte size}^{\alpha} \qquad \alpha < 1 \qquad [2.72]$$

By selecting $\alpha = 0.3750$, the byte length normalization scores are comparable to the cosine normalization factor. In [SIN 96a], the normalization *byte size*$^{0.3750}$ was proven to yield substantial retrieval improvements in degraded documents, such as the texts scanned through optical character recognitions (OCR). The process results in a large number of rare words due to errors.

A second limitation of cosine normalization is that "it tends to favour documents of short lengths" [SIN 96a]. This conclusion was stated in [SIN 96a] after some observations regarding both likelihoods of retrieval and relevance of documents, with respect to the document average lengths. The relevance-retrieval likelihood curves found to be non-proportional, but similar to the curves of Figure 2.11.

As can be observed in Figure 2.11, the probability of retrieval is lower than the relevance probability for documents

with length (length<pivot) and greater than the relevance probability for length>pivot. This fact is attributed to the cosine normalization factor: for short lengths, cosine normalization scores produce too high normalized term weights when compared with the real relevance of the documents. As a consequence, the probability of retrieval is superior than the probability of relevance. The opposed effect is observed for long documents, which can be associated with normalization scores excessively high for the documents relevance.

The proposed solution to deal with these observations was to rotate or "tilt" the cosine normalization line around the pivot point, in such a way that the new normalization factor is greater than the cosine normalization for length<pivot, and lower than the cosine factor for length>pivot. This fact is illustrated in Figure 2.11(b).

The conversion of the cosine normalization (old factor) to the new pivoted normalization is given by equation [2.73]:

$$\text{Pivoted norm} = (1 - \text{slope})\ x\ \text{pivot} + \text{slope}\ x\ \text{old factor} \quad [2.73]$$

where *slope* refers to the slope of the new pivoted normalization line. The *slope* and *pivot* parameters are to be determined from available training data. After some manipulations, equation [2.73] can be expressed by the reduced form of equation [2.74]

$$\text{Pivoted norm} = (1 - \text{slope}) + \text{slope}\ x\ \frac{\text{old factor}}{\text{average old factor}} \quad [2.74]$$

which allows us to simplify the training process, since *slope* is now the only parameter that needs to be learned from training data.

The final term weights can be obtained by using any combination of scores from the term frequency, collection

		Term freq.	Collection freq.	Normalization
Best fully weighted system	Document	$TF_{(a)}$	IDF_a	Cosine Norm
	query	$TF_{(c)}$	IDF_a	1 (None)
Best weighted probabilistic weight	Document	$TF_{(c)}$	1	1
	query	1	IDF_b	1
Binary term independence	Document	1	1	1
	query	1	IDF_b	1
Coordination level	Document	1	1	1
	query	1	1	1

Table 2.4. *Typical combination for term weighting in IR approaches as described in [SIN 09]*

frequency, and normalization factors. Furthermore, the selected combinations for the document vectors usually differ from the term weights of the query vector. Note also that the normalization factor is absent in the query term weights, since it only scales the similarities between the query and all the document vectors by a constant term.

Among the classical term weighting schemes, Table 2.4 shows some of the typical combinations described in [SAL 88] for the query and document vectors.

2.5.3.5. *Probabilistic models*

While vector space models rely on the similarity between documents and query vectors to rank and retrieve relevant documents, in probabilistic IR [MAR 60, FUH 89], documents are ranked in terms of decreasing probability of relevance to the query (the so-called probability ranking principle (PRB)).

2.5.3.5.1. Binary independence retrieval model

One of the simplest approaches to probabilistic IR is the *binary independence model* (BIR) [ROE 00] in which documents and queries are viewed as binary vectors – in a similar way as Boolean vector representation of documents in vector-based retrieval – only accounting for the absence or presence of terms in the query and documents.

Instead of computing direct probabilities, the parameters to be estimated are the *odds* of the document relevance with respect to the query:

$$O(R|d,q) = \frac{P(R|d,q)}{P(\bar{R}|d,q)} = \frac{P(R|d,q)}{1 - P(R|d,q)} \quad [2.75]$$

where O denotes the odd of relevance of document d given the query q, and the term $P(\bar{R}|d,q)$ refers to the probability that the document is not relevant given the query. Using the Bayes theorem, equation [2.75] can be rewritten into the form of equation [2.76]:

$$O(R|d,q) = \frac{P(R|d,q)}{P(\bar{R}|d,q)} = \frac{P(R|q)}{P(\bar{R}|q)} \cdot \frac{P(d|R,q)}{P(d|\bar{R},q)} = O(R|q) \cdot \frac{P(d|R,q)}{P(d|\bar{R},q)} \quad [2.76]$$

In the BIR model, it is assumed that the incidence of one term in a document or query is independent of the occurrence of any other terms. This term independence assumption allows us to reformulate equation [2.76], by decomposing the document probabilities by the product of the terms probabilities, as follows:

$$O(R|d,q) = O(R|q) \cdot \frac{P(d|R,q)}{P(d|\bar{R},q)} = O(R|q) \cdot \prod_{t_i \in \mathcal{W}} \frac{P(t_i|R,q)}{P(t_i|\bar{R},q)} \quad [2.77]$$

The product in [2.77] can be further decomposed by considering the presence and absence of query terms in the document separately:

$$O(R|d,q) = O(R|q) \cdot \prod_{t_i \in d} \frac{P(t_i = 1|R,q)}{P(t_i = 1|\bar{R},q)} \cdot \prod_{t_i \in \mathcal{W}} \frac{P(t_i = 0|R,q)}{P(t_i = 0|\bar{R},q)} \quad [2.78]$$

Let $p_i = P(t_i = 1|R,q)$ denote the probability that one term is present in the document ($t_i = 1$) and is relevant to the query;

$q_i = P(t_i = 1|\bar{R}, q)$ the probability that one term present in the document is not relevant to the query. Analogously, $1 - p_i = P(t_i = 0|R, q)$, the probability of absence of the term in the document, given the term's relevance to the query, and $1 - q_i = P(t_i = 0|R, q)$, the probability of absence of the term in the document, given that the term is not relevant to the query. Equation [2.78] can be thus be expressed in terms of p_i and q_i:

$$O(R|d, q) = O(R|q) \cdot \prod_{t_i \in d} \frac{p_i}{q_i} \cdot \prod_{t_i \in \mathcal{W}} \frac{1 - p_i}{1 - q_i} \qquad [2.79]$$

Finally, the two product indices in equation [2.79] can be slightly modified in such a way that the first component accounts for the terms present in a document, while the second factor reflects the common odds of all terms in the lexicon:

$$O(R|d, q) = O(R|q) \cdot \prod_{t_i \in d} \frac{p_i(1 - q_i)}{q_i(1 - p_i)} \cdot \prod_{t_i \in \mathcal{W}} \frac{1 - p_i}{1 - q_i} \qquad [2.80]$$

As can be noted, the second product includes all lexicon terms and therefore remains a constant value for all documents, since it only depends on the query search under consideration. Likewise, the relevance odd $O(R|q)$ is also common to all documents in the collection for the given query. Thus, the only factor on which a document's relevance to the query depends is the fist product element in equation [2.80]. Hence, this factor is used to calculate the document ranking with respect to the query, also referred to as the document's *retrieval status value* (RSV):

$$RSV(d, q) = \log \prod_{t_i \in d} \frac{p_i(1 - q_i)}{q_i(1 - p_i)} \qquad [2.81]$$

The parameters p_i and q_i can be estimated, for example, by asking the user to categorize a initial set of presented documents into relevant and non-relevant categories. In

probabilistic IR, this kind of approaches are known as *relevance feedback*. For example, if r is the number of relevant documents from a total number of documents N, n_i is the number of documents where the term t_i occurs, and r_i is the total number of relevant documents containing the term, then p_i can be estimated as $ti = r_i/R$ and q_i is approximated as $q_i = (N_i - r_i)/N - r$. Using these estimates, the RSV can be reformulated as:

$$RSV(d,q) = \sum t_i \in d \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} \quad [2.82]$$

The summands in equation [2.82] are also known as the Robertson/Spark Jones weights, named after the authors who first proposed this formulation, and it is often used as the basis factor in term weighting approaches to probabilistic IR, such as the Okapi term weighting (see further details below). The 0.5 terms that have been added in equation [2.82] are smoothing factors to address the zero probability problem.

2.5.3.5.2. The 2-Poisson model

In contrast to the binary model that only considers the presence or absence of terms in a document, the 2-Poisson model was proposed to provide a more refined estimation of the term probabilities that accounts for the number of occurrences of terms in the documents [NA 09].

According to a Poisson distribution, the probability that a document with k occurrences of term t_j belongs to a certain category, $\mathcal{C}_j$ (in this case, one of the relevant/non-relevant categories) is defined as:

$$P(tf_i = k|d \in \mathcal{C}_j) = \frac{\lambda_{ij}^k}{k!} e^{-\lambda_{ij}} \quad [2.83]$$

where the parameter λ_{ij} denotes the average number of occurrences of the term t_i over documents with category $\mathcal{C}_|$, according to a Poisson distribution.

2.5.3.5.3. Okapi weighting

The BM25 weighting, also named Okapi weighting after the system where it was first implemented, is an extension of the Robertson/Spark Jones weights in equation [2.82]. Besides the binary presence/absence of terms, the BM25 scheme incorporates the term frequency and normalization factors typically applied in vector-based approaches. The final term weights (equation [2.84]) were proposed after successive refinements, contributed by the authors to a series of term weighting (TREC) conferences [BEA 97, ROB 92, ROB 99, ROB 00].

$$RSV = \sum_{t_i \in Q} w_i \cdot \frac{(k_1+1)TF(t_i,d)}{k_1((1-b)+b(dl/avgl))+TF(t_i,d)} \cdot \frac{(k_3+1)TF(t_i,q)}{k_3+TF(t_i,q)} \quad [2.84]$$

where w_i is the Robertson/Spark Jones weights in equation [2.84], dl denotes the document length and $avgl$ is the average length on all documents. Thus, the parameter $b \in [0,1]$ controls the different emphasis placed on document length normalization. $b = 0$ implies no length normalization, while $b = 1$ means normalization by the value given by $dl/avdl$. Likewise, $TF(t_i,d)$ denotes the number of occurrences of the term t_i in the document d, and k_1 is a parameter that controls the amount of term frequency component that contributes to the final RSV scores. $k_1 = 0$ removes any term frequency effect, while $k_1 = 1$ implies full normalization by TF(t,d). (It should be noted that the second factor equals TF(t_i,d) for $b = 0$ and $k_1 = 1$.) Finally, TF(t_i,q) indicates the number of occurrences of the term t_i in the query, a factor which can be tunned by k_3, in a similar way as with the previous factors. For example, for short queries, it is reasonable to discard the term frequency in the query, thus k_3 should be set to 0.

2.5.4. *Clustering documents in information retrieval*

Modern IR systems incorporate different clustering schemes to assist the retrieval and/or presentation of results in response to a user query. Clustering methods have been applied in IR applications with different purposes:

2.5.4.1. *Clustering of presented results*

The majority of Web search engines return the set of retrieved documents in response to a query as a ranked list in decreasing order of relevance. However, if the query terms are polysemic or can be found in different conceptual categories (such as *jaguar(animal)* and *jaguar(car)*), poor precision values are to be expected in the presented results. This drawback of conventional presentation approaches can be overcome by clustering the retrieved documents and return the clustered results to the user. An example Web crawler that provides a cluster-based presentation of the search results is vivisimo (www.vivisimo.com).

2.5.4.2. *Post-retrieval document browsing (Scatter-Gather)*

This approach provides a new interface to the user, who does not need to type a search query. Thus, it is also known as "search without typing" [HEA 96, CUT 92]. The total of documents in the collection are clustered and the user only needs to select the group or groups of documents of interest. In case more than one cluster is selected, the clusters' documents are merged into a document set, which is in turned re-clustered. The process is iteratively repeated until the user selects a single cluster. Since this type of clustering application requires on-line processing, a number of requirements must be met by the clustering methods, in particular *speed* and *scalability* – clustering must be performed incrementally as documents are being received over the web. To comply with the speed requirement, clustering algorithms with linear complexity are usually

selected, such as the k-means or the Rocchio algorithm [ZAM 98]. Fractionation approaches are based on hierarchical agglomerative schemes but reduce the quadratic or cubic complexities of hierarchical algorithms to linear time complexity by restricting the search of the pairs of clusters to be merged to local regions. Another clustering scheme proposed for the Scatter-Gather scenario is the suffix tree clustering (STC). The algorithm meets both scalability and liner complexity criteria by identifying phrases common to a group of documents [WAN 08].

2.6. Evaluation methods

Evaluating the partitions obtained through cluster analysis is a more difficult task than the evaluation of supervised classifiers. However, some strategies have been proposed in the cluster literature for assessing the outcome of a clustering algorithm. If reference labels are available for the clustered data, external cluster validation can be used to compare the cluster partition with the set of reference labels. In the absence of reference labels, internal validation criteria can be applied to obtain an "estimation" of the clustering quality on the basis of the result clusters' distances.

2.7. Internal cluster evaluation

For a comprehensive evaluation of the discussed algorithms, their cluster solutions have been also compared with the reference category labels, available for evaluation purposes, using three typical "external" cluster validation methods: entropy, purity, and normalized mutual information (NMI).

2.7.1. *Entropy*

The cluster entropy [BOL 99] reflects the degree to which the clusters are composed of heterogeneous patterns, i.e. patterns that belong to different categories. According to the entropy criterion, a good cluster should be mostly aligned to a single class, which means that a large among of the cluster objects belong to the same category. This quality condition corresponds to low entropy values. The entropy of a cluster i is defined as

$$E_i = -\sum_{j=1}^{L} \log_2(p_{ij}) \qquad [2.85]$$

where L denotes the number of reference categories, and p_{ij} the probability that an element of category j is found in cluster i. This probability can be formulated as $p_{ij} = \frac{m_j}{mi}$, denoting m_j the number of elements of class j in the cluster i and m_i the total number of elements in the cluster i.

The total entropy of the cluster solution C is obtained by averaging the cluster entropies according to equation [2.86] (m denotes the total number of elements in the dataset):

$$E(\mathcal{C}) = -\sum_{i=1}^{k} \frac{m_i}{m} E_i \qquad [2.86]$$

As discussed earlier, "good" cluster solutions yield small entropy values.

2.7.2. *Purity*

Like entropy, purity [BOL 99, WU 09] is another metric to measure the extent to which a cluster contains elements of a single category. The purity of a cluster i is defined in terms of the maximum class probability, $P_i = \max_j(p_{ij})$.

The overall purity of a cluster solution is calculated by averaging the cluster purities:

$$P(\mathcal{C}) = -\sum_{i=1}^{k} \frac{m_i}{m} P_i \qquad [2.87]$$

Higher purity values indicate a better quality of the clustering solution, up to a purity value equal to one, which is attained when the cluster partition is perfectly aligned to the reference classes.

2.7.3. *Normalized mutual information*

The NNMI was proposed in [STR 02] as a metric of the agreement between two partitions of the data, $\lambda^{(a)}$ and $\lambda^{(n)}$ (equation [2.88]).

$$NMI(\lambda^{(a)}, \lambda^{(b)}) = \frac{\sum_{h=1}^{k(a)} \sum_{l=1}^{k(b)} n_{h,l} \log\left(\frac{n \cdot n_{h,l}}{n_h^{(a)} n_l^{(b)}}\right)}{\sqrt{\left(\sum_{h=1}^{k(a)} n_h^{(a)} \log\left(\frac{n_h^{(a)}}{n}\right)\right)\left(\left(\sum_{l=1}^{k(b)} n_l^{(b)} \log\left(\frac{n_l^{(b)}}{n}\right)\right)\right.}} \qquad [2.88]$$

Denoting n, the number of observations in the dataset, $k^{(a)}$ and $k^{(b)}$, the number of clusters in the partitions $\lambda^{(a)}$ and $\lambda^{(b)}$; $n_h^{(a)}$ and $n_l^{(b)}$, the number of elements in the clusters C_h and C_l of the partitions $\lambda^{(a)}$ and $\lambda^{(b)}$, respectively, and $n_{h,l}$, the number of overlapping elements between the clusters C_h and C_l. The NMI can be used as a external quality metric of a cluster partition by comparing the cluster solution $\mathcal{C}$ with the reference class labels $\mathcal{L}$, $NMI(C, \mathcal{L})$.

2.8. External cluster validation

As outlined in section 2.4, the determination of the number of clusters in a dataset is a principal problem associated with many clustering algorithms.

In the following, we denote $\mathcal{C} = \{C_1, \ldots, C_k\}$, a cluster partition composed of k clusters, and N, the total number of objects in a dataset. The cluster validation indexes applied in our experiments are the following.

2.8.1. *Hartigan*

This metric was proposed by J. A. Hartigan for detecting the optimum number of clusters k to be applied in the k-means clustering algorithm [HAR 75]:

$$H(k) = \gamma(k)\frac{W(k) - W(k+1)}{W(k+1)}, \quad \gamma(k) = N - k - 1 \qquad [2.89]$$

denoting $W(k)$ the intra-cluster dispersion, defined as the total sum of square distances of the objects to their cluster centroids. The parameter γ is introduced to avoid an increasing monotony with increasing k. In this work, we use a small modification to the Hartigan metric, by treating the parameter $W(k)$ as the average intra-cluster distance.

According to Hartigan, the optimum number of clusters is the smallest k which produces $H(k) \leq \eta$ (typically $\eta = 10$). However, to allow a better alignment of the Hartigan index to other scores in the combination approach, we have introduced a correction of the index: $Hc(k) = H(k-1)$ and considered a modification of the optimum criterion by maximizing $Hc(k)$. In other words, the new criterion maximizes the relative improvement at k with respect to $k-1$, in terms of decreasing dispersion. This allows for a direct application of the corrected index $Hc(k)$ in the combination approach without resorting to a previous inversion of the scores.

2.8.2. *Davies Bouldin index*

The Davies Bouldin index [DAV 79] was proposed to find compact and well separated clusters. It is formulated as:

$$DB(k) = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\frac{\Delta(C_i)+\Delta(C_j)}{\delta(C_i,C_j)} \qquad [2.90]$$

where $\Delta(C_i)$ denotes the intra-cluster distance, calculated as the average distance of all the cluster objects C_i to the cluster medoid, whereas $\delta(C_i, C_j)$ denotes the distance between the clusters C_i and C_j (distance between the cluster medoids). The optimum number of clusters corresponds to the minimum value of $DB(k)$.

2.8.3. *Krzanowski and Lai index*

This metric belongs to the so-called "elbow models" [KRZ 85]. These approaches plot a certain quality function over all possible values for k and detect the optimum as the point where the plotted curves reach an elbow, i.e. the value from which the curve considerably decreases or increases. The Krzanowski and Lai index is defined as:

$$KL(k) = \left|\frac{\text{Diff}_k}{\text{Diff}_{k+1}}\right| \qquad [2.91]$$

$$\text{Diff}_k = (k-1)^{\frac{2}{m}}W_{k-1} - k^{\frac{2}{m}}W_k \qquad [2.92]$$

The parameter m represents the feature dimensionality of the input objects (number of attributes), and W_k is calculated as the within-group dispersion matrix of the clustered data:

$$W_k = \sum_{i=1}^{k}\sum_{x_j\in C_i}(x_j - c_i)(x_j - c_i)^T \qquad [2.93]$$

In this case, x_j represents an object assigned to the jth cluster, and c_i denotes the centroid or medoid of the ith cluster. The optimum k corresponds to the maximum of $KL(k)$.

2.8.4. *Silhouette*

This method is based on the *silhouette width*, an indicator for the quality of each object i [ROU 87]. The silhouette width is defined as:

$$sil(x_i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad [2.94]$$

where $a(i)$ denotes the average distance of the object i to all objects of the same cluster and $b(i)$ is the average distance of the object i to the objects of the closest cluster.

Based on the object silhouettes, we can extend the silhouette scores to validate each individual cluster using the average of the cluster object silhouettes:

$$sil(C_j) = \frac{1}{|C_j|} \sum_{x_i \in C_j} sil(x_i) \quad [2.95]$$

Finally, the silhouette score that validates the whole partition of the data is obtained by averaging the cluster silhouette widths:

$$sil(k) = \frac{1}{k} \sum_{r=1}^{k} sil(C_r) \quad [2.96]$$

The optimum k maximizes $sil(k)$.

2.8.5. *Gap statistic*

The idea behind the Gap statistic is to compare the validation results of the given dataset with an appropriate

reference dataset drawn from an *a priori* distribution [ROB 01]. Thereby, this formulation avoids the increasing or decreasing monotony of other validation scores with increasing number of clusters.

First, the intra-cluster distance is averaged over the k clusters:

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} \sum_{i,j \in C_r} D(i,j) \qquad [2.97]$$

where n_r denotes the number of elements of the cluster r. The Gap statistic is defined as:

$$Gap(k) = E(\log(W_k)) - \log(W_k) \qquad [2.98]$$

where $E(\log(W_k))$ is the expected logarithm of the average intra-cluster distance. In practice, this expectation is computed through a Monte-Carlo simulation on a number of sample realizations of a uniform distribution B.[2]

$$Gap(k) = \frac{1}{B} \sum_{b} (\log(W_{kb})) - \log(W_k) \qquad [2.99]$$

where W_{kb} denotes the average intra-cluster distance of the bth realization of the reference distribution using k clusters. The optimum number of clusters is the smallest value k such that $Gap(k) \geq Gap(k+1) - s_{k+1}$, where s_k is a factor that takes into account the standard deviation of the Monte-Carlo replicates W_{kb}.

Current research is focused on unsupervised term weighting scoring schemes, to be also applied in metrics based on words affinities. Furthermore, hierarchy and

2. It should be noted that the reference data drawn from this uniform distribution consist of a number N of objects identical to the dataset, with identical number of features m. The values of each feature in each object are assigned randomly in the original feature range.

classifications with dynamic features are also being analyzed, where features (e.g. words) of increasing relevance are sequentially incorporated as we descend to lower hierarchy levels of the hierarchy for refined matches or clusters divisions.

2.9. Semi-supervised learning

Semi-supervised classification is a framework of algorithms proposed to improve the performance of supervised algorithms through the use of both labeled and unlabeled data [DES 00]. One reported limitation of supervised techniques is their requisite of available training corpora of considerable dimensions to achieve accurate predictions on the test data. Furthermore, the high effort and cost associated with labeling large amount of training samples by hand—a typical example is the manual compilation of labeled text documents—is a second limiting factor. It led to the development of semi-supervised techniques. Many studies have shown how the knowledge learned from unlabeled data can significantly reduce the size of labeled data required to achieve appropriate classification performances [NIG 99, CAS 95].

Different approaches to semi-supervised classification have been proposed in the literature, including, among others, co-training [MAE 04], self-training [YAR 95], or generative models [NIG 00, NIG 99]. Two extensive surveys on semi-supervised learning are provided in [ZHU 06] and [SEE 01]. Unsupervised learning algorithms can be divided in several groups: *self-training*, *co-training* and, *generative models*.

2.9.1. *Self training*

In self-training, a single classifier is iteratively trained with a growing set of labeled data, stating from a small

initial seed of labeled samples. Commonly, an iteration of the algorithm entails the following steps: 1) training on the labeled data available from previous iterations, 2) applying the model learned from labeled data to predict the unlabeled data, and 3) sorting the predicted samples according to their confidence scores and adding the top most confident ones with their predicted labels to the labeled set, which implies removing them from the unlabeled dataset.

One example of self-training is the work by Yarowski [YAR 95]. A self-training approach was applied to word sense disambiguation. The basic problem was to classify a word and its context into the possible word senses in a polysemic corpus. The algorithm was supported by two important constraints for the augmentation of the labeled senses: (1) the collocation constraint, according to which a word's sense is unaltered if the word co-occurs with the same words in the same position (collocation) and (2) the one sense per discourse, according to which a word sense is unaltered in the discourse where the word appears, e.g. within a document. The algorithm was started by a tagged seed for each possible sense of the word, including important seed collocates for each sense. The sense labels were then iteratively augmented according to self-training approaches. In this case, the *one sense per discourse* criterion was also applied to achieve more augmentation with samples within the documents.

2.9.2. *Co-training*

In a similar way as self training, co-training approaches are based on an incremental augmentation of the labeled sets by iteratively classifying the unlabeled sets and attaching the most confident predicted samples to the labeled set. However, in contrast to self-training, two complementary classifiers are simultaneously applied, fed with two different "views" of the feature set. The prediction of the first classifier is used to

augment the labeled set available to the second classifier and vice-versa. To obtain a maximum benefit from this "synergy" of classifiers, two important assumptions should be fulfilled:

– *Compatibility of classifiers:* The classification models should mostly "agree" in their predictions, i.e. if a sample is classified to class y_j by the first classifier, it should be probably classified to the same class by the second classifier.

– *Conditional independency of feature subsets:* No conditional dependency should be observed between the two feature subsets applied to the classifiers.

In [MAE 04], a co-training strategy was applied to predict the emotional/non-emotional character of a corpus of student utterances collected within the ITSPOKE project (Intelligent Tutoring Spoken dialog system). As the conditional independency between the different feature sets could not be proved, the authors selected two *high-precision* classifiers. The first one was trained to recognize the emotional status of an utterance (e.g. "1" emotional vs "0" for non-emotional), whereas the second one predicted its non-emotional status ("1" non-emotional vs "0" emotional). The labeled set was iteratively increased by attaching the top most confident predicted samples to the labeled set from previous iterations. Furthermore, the feature subsets applied to each classifier were optimized according to two evaluation criteria, using a greedy search algorithm.

2.9.3. *Generative models*

Denoting $\mathcal{X}$, the set of data points, in $\mathcal{R}^D$, and Y the set of class labels corresponding to the dataset, a generative model assumes an underlying model of mixtures $p(x|y)$, which should be identifiable by using certain tools such as the EM algorithm or clustering methods.

In [NIG 99], the EM algorithm was used for the semi-supervised classification of texts. The model parameters, θ, to be inferred by the algorithm, were defined as the set the word/class probabilities and the class prior probabilities.

Other strategies attempt to derive the mixture model by means of clustering. These approaches are commonly referred to as *cluster-and-label*. For example, in [DEM 99] a *genetic* k-means clustering was implemented using a genetic algorithm (GA) (see section for more details about GAs). The goal of the algorithm was to find a set of k cluster centers that simultaneously optimised an internal quality objective (e.g. minimum cluster dispersion) and an external criterion based on the available labels (e.g. minimum cluster entropy). Thus, a real value chromosome representation was selected, with a chromosome length CL $= Dxk$, where D is the number of features in the dataset. Within each iteration (generation) of the GA, the clusters where built by connecting each point to the closest center (k-means). The labels where expanded to all patterns by labeling each cluster using a majority voting strategy. This way, the total cluster entropy could be calculated. As aforementioned, the simultaneous optimization of both internal and external criteria was attained through the formulation of a new objective as a linear combination of both

$$\text{minimize } \mathcal{O} : \{\alpha * \text{Dispersion} + \beta * \text{Entropy}\} \qquad [2.100]$$

In [DAR 02], a SOM was applied to cluster unlabeled data. The SOM was first trained using the labeled data seed. If all the labeled samples that shared an identical winning node also had an identical label, say, l_i, that node was labeled as l_i. Otherwise the node was considered as "non-labeling." In a subsequent clustering phase, the unlabeled data where "clustered" to their closest units in the map (winning nodes). During the clustering process, all unlabeled data clustered to a particular node were also implicitely labeled with the

node's label, in case the node had been assigned a label in the training phase.

2.10. Summary

In this chapter, some popular algorithms in the areas of unsupervised and semi-supervised machine learning have been discussed. In the first part, an overview of the different clustering strategies has been presented (hierarchical algorithms, competititive models, model-based algorithms, density-based clustering, and graph clustering). The clustering techniques that require the number of clusters k as an input argument include the hierarchical clustering, k-means, neural gas, the SOM (in such cases when the algorithm is to be used directly for clustering and the PAM algorithms). The techniques which do not require k as an input argument are the DBSCAN algorithm, the SOM for visualization, and the PoBOC algorithm. However, DBSCAN requires input density parameters. The SOM requires human inspection of the U-Matrix to decide k and its solution depends on other variables, such as the neighborhood function and the number of nodes in the kohonen layer. Thus, the PobOC clustering algorithm is the unique strategy which requires neither k nor other corresponding *a priori* information from the user. This first part of the chapter has been concluded with an overview of the clustering application areas and examples, and some cluster internal evaluation techniques. These latter metrics can also be used to detect the number of clusters k, in combination with clustering approaches which accept k as input (see above). In this case, the metrics are also referred to as *cluster validation*.

In the following chapters, some of the aforementioned clustering approaches have been selected for different purposes. For example, in the second part of the book (Chapters 3 and 4), which is devoted to semi-supervised

classification, clustering is used as an essential part of the proposed semi-supervised algorithms. As the number of clusters k is asumed to be known (equal to the number of classes), algorithms which require k are selected. Due to the disadvantages of $k-$means (in particular, the sensitivity to the initialization) and neural gas (more robust w.r.t the initialization than k-means but also sensitive to the choice of other parameters), hierarchical approaches (Chapter 3) and the PAM algorithm (Chapter 4) have been preferred for semi-supervised classification. As aforementioned, the PoBOC algorithm is the only strategy, to the author's knowledge, which is able to infer the number of clusters. This algorithm has been used in the third part of the book (Chapter 5), as the baseline of a new hierarchical version (HPoBOC), which addresses a limitation of PoBOC to detect clusters in hierarchies. Finally, the cluster validation techniques are also addressed in Chapter 6, in which a combination approach is proposed to solve the "individual" performances of these kinds of approach.

In the second part of this chapter, different semi-supervised approaches have been described (self-training, co-training, and generative models). These approaches differ in their definitions of the objective functions. However, as described above, the objective is formulated in all cases as a global function which takes into consideration labeled and unlabeled data simultaneously. In contrast, the semi-supervised approaches developed in the book (Chapters 3 and 4) separate this global objective into two different optimization functions: a first one optimizing a clustering (first step) and a second one optimizing the cluster labeling given to the labeled data. In defining these separate objectives, the new approaches may reduce the influence of potential labeling errors in the cluster solutions.

PART 2

Approaches to Semi-Supervised Classification

Chapter 3

Semi-Supervised Classification Using Prior Word Clustering

3.1. Introduction

In the first part of this book, two semi-supervised approaches have been developed, which exploit the availability of unlabeled data by means of unsupervised clustering.

As stated in Chapter 1, cluster and label approaches in the machine learning literature often merge the cluster and label steps as a global optimization problem in which both tasks are simultaneously solved. The approaches developed in this book are intended to avoid the influence of the labeled seeds on the cluster solution, which can induce wrong clustering if potential labeling errors are present in the labeled sets.

In particular, the algorithm described in this chapter is based on the *synonymy assumption*: under minimal class labels, the underlying classes can be approximately recovered by extracting semantic similarities from the data. The synonymy assumption has been applied to the feature

Semi-Supervised and Unsupervised Machine Learning: Novel Strategies Amparo Albalate and Wolfgang Minker

space in which a clustering algorithm is used to extract groups of synonym words. In previous research [LI 98], word clustering has been typically investigated as a feature clustering strategy for *supervised* text classification. By applying word clustering for supervised classification, a degradation in the classification accuracy was reported. However, in the experiments described in this chapter, it is shown that a classification algorithm using minimum labeled seeds can benefit from word clustering on unlabeled data (semi-supervised classification).

Furthermore, besides the use of unlabeled data for word clustering, unlabeled data has also been used for unsupervised disambiguation of ambiguous utterances.

3.2. Dataset

The approaches described in the following have been applied to a collection of utterance transcripts. In particular, the utterance dataset from the video troubleshooting domain has been selected (see section 1.1 in Chapter 1).

3.3. Utterance classification scheme

The supervised utterance classification scheme is depicted in Figure 3.1. First, each utterance is pre-processed into an utterance vector. The classification algorithm then maps this vector into one of the categories from the list of symptoms. The pre-processing step and classification algorithm are described in detail in sections 3.3.1 and 3.3.2.

3.3.1. *Pre-processing*

As mentioned earlier, the pre-processing step transforms each input utterance transcript u_i into an utterance vector

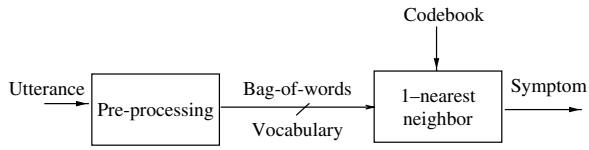


Figure 3.1. *Supervised utterance categorization scheme using the NN algorithm*

x_i, which is the utterance form used by the classification algorithm. The first pre-processing step extracts the surface form of the utterance words. As introduced in Chapter 1, a possible surface form of a word is the word's stem. However, after word stemming, the lexical category of the original word is not preserved. For example, the following utterances

"need a worker to install Internet" [Appointment]
"have installed Internet but it does not work" [Internet]

would lead, after stop-word removal, to very similar bag-of-words vectors if the word *worker* were reduced to its stem form *work*. However, the utterances have been annotated with different categories ([appointment] and [Internet], respectively). For this reason, a lemmatizer [TOU 00] has been selected instead (note that the lemma form of *worker* remains *worker*).

Following the lemmatizing step, stop-word filtering has been applied. It discards ubiquitous terms deemed irrelevant for the classification task as they do not contribute any semantic content to the utterances. Examples are the lemmas *a, the, be, and for*. In this book, the SMART stop list has been used [SAL 71] with small modifications. In particular, confirmation terms (*yes*, *no*) have been deleted from the stop-word list, whereas words typical for the spontaneous speech (*eh, ehm, uh, ...*) have been introduced as stop words.

As an example of the transformation of utterances and words through the pre-processing steps, the input utterance

My remote control is not turning on the television is transformed as follows:

> Utterance (u_i): "My remote control is not turning on the television"
> + Lemmatizing ($u_i^{(l)}$): "My remote control be not turn on the television"
> + Stop-word filtering ($u_i^{(l+s)}$): Remote control not turn television

where the notations u_i, $u_i^{(l)}$, and $u_i^{(l+s)}$ refer to the initial utterance transcript, the utterance after lemmatizing (l), and the utterance after lemmatizing and stop-word filtering $(l+s)$.

3.3.1.1. *Utterance vector representation*

The salient vocabulary or lexicon is obtained after pre-processing all utterances in the training corpus, $\mathcal{X}_T$. The lexicon $\mathcal{W} = \{w_1, w_2, \ldots, w_D\}$ is then defined as the set of different lemmas found in the pre-processing training set. In the following, the lexicon elements are also referred to as terms. The number of terms extracted from the Internet corpus is $D = 1{,}614$. Finally, a binary vector model ("bag-of-words") has been selected to represent the pre-processed utterances,

$$x_i = [x_{i_1}, x_{i_2}, \ldots, x_{i_D}] \qquad [3.1]$$

where x_i denotes an utterance vector. As already explained for texts, each binary element $x_i^{(j)}$ indicates the presence or absence of the term w_i in the pre-processed utterance transcript, $u_i^{(l+s)}$:

$$x_i^{(j)} = \begin{cases} 1, & if\ w_j \in u_i^{(l+s)} \\ 0, & if\ w_j \notin u_i^{(l+s)} \end{cases}$$

3.3.2. *Utterance classification*

A simple yet robust categorization approach is the nearest neighbor (NN) algorithm (j-NN). Let $\{X_T^{(l)}, Y_T^{(l)}\}$ denote the

labeled seed of prototype vectors and $\mathcal{K}$ the set of k possible class labels. The j-NN algorithm compares each input utterance vector x_i to its closest j prototype vectors (nearest neighbors) and decides the most frequent category in the NNs according to a majority voting criterion. In this work, the labeled seed is composed of one prototype per category ($n = 1$). Thus, the j-NN algorithm derives into a simple 1-NN rule. The NN classification rule, θ_{NN}, decides for each input utterance vector in a test set, $x_i \in \mathcal{X}^{(\mathrm{test})}$, the symptom category of the closest prototype vector:

$$\theta_{\mathrm{NN}}(x_i) \doteq \hat{y}_i = y_k \in \mathcal{Y}_T^{(l)}, \qquad x_k = \underset{(x_j \in \mathcal{X}_T^{(l)})}{\operatorname{argmax}} \left(d(x_j, x_i)\right) \qquad [3.2]$$

In this work, the overlap distance has been applied to compute the dissimilarities between utterance vectors. First, the overlap similarity between two utterance vectors is the number of words that overlap in the pre-processed utterances. This metric of similarity is equivalent to the dot product between utterance vectors:

$$s_{\mathrm{overlap}}(x_i, x_j) =< x_i, x_j > \qquad [3.3]$$

The overlap distance can be calculated as the inverse of the overlap dissimilarity:

$$d_{\mathrm{overlap}}(x_i, x_j) = N - s_{\mathrm{overlap}}(x_i, x_j) \qquad [3.4]$$

Note that the prototype vectors applied to the NN algorithm have been manually selected. In this work, the basic criterion for the selection of the prototype set $\{\mathcal{X}_T^{(l)}, \mathcal{Y}_T^{(l)}\}$ is the minimum possible term overlap between different prototypes in the codebook.

3.4. Semi-supervised approach based on term clustering

In this section, a semi-supervised version of the utterance classification scheme in Figure 1.1 is introduced. By applying a unique labeled utterance per category, a poor classification performance is to be expected, in comparison with other classifiers with large labeled prototype seeds. One reason for this poor performance can be explained by the prevalence of certain language effects such as *synonymy* and *polysemy*, which are not sufficiently covered in the minimum labeled seed. Synonymy refers to the capability of language to describe a single semantic concept with different words (synonym). Polysemy, however, is the use of a unique word form with multiple meanings.

The synonymy effect is patent in the effective vocabulary available to the classifier, which is reduced to less than 5% of the total vocabulary dimension in $\mathcal{W}$. This results in a large amount of utterances mapped to a [*nomatch*] class, given the existence of out-of-vocabulary terms in the test utterances.

As an example, one of the symptom categories, [NoSound], represents the problems related to sound. A prototypical caller utterance reporting this problem is "*No sound.*" However, the user may speak other alternatives, such as "*problem with volume*" or "*lost audio,*" which are not matched against the mentioned prototype (no sound) due to the orthogonality between their vectors. In other words, there are no overlapping terms between the input utterances and the prototype utterance. This problem can be partially solved using the observation that *sound* has a similar meaning as that of *audio* or *volume*.

Given these statements, a feature extraction technique that identifies groups of terms on the basis of the semantic affinities among them has been developed in this work.

Synonym terms are identified from a training set composed of both labeled and unlabeled data:

$$\mathcal{X}_T = \mathcal{X}_T^{(l)} \cup \mathcal{X}_T^{(u)} \quad [3.5]$$

where $\mathcal{X}_T^{(l)}$ refers to the labeled seed (prototypes) and $\mathcal{X}_T^{(u)}$ indicates the unlabeled portion of $\mathcal{X}_T$. Thus, the new algorithm may be considered as a semi-supervised version of the basic classification scheme as shown in Figure 3.1. A similar approach was made in the literature [DEP 00], in which a list of manually selected keywords and their synonyms was used to trigger an unsupervised classification of texts. In this work, a method to automatically extract classes of semantically related words from the corpus of utterance transcripts has been developed.

Although a number of manually compiled word thesauri are available nowadays, such as Wordnet [WOR 98], the automatic identification of related words provides certain advantages over manually constructed thesauri. The most important one is that automatic methods can be better adapted to a specific domain, as they are statistically "trained" on specific data. As an example, the words *confirm* and *schedule* are not apparently synonym terms. However, on the Internet corpus, both terms are strongly related to the category [Appointment]. Typical user utterances for this category are, for example, "I want to confirm an appointment" and "I want to schedule an appointment." Furthermore, both terms have barely any occurrences outside the [Appointment] category.

The block diagram of the new semi-supervised algorithm with term clustering can be seen in Figure 3.4.

3.4.1. *Term clustering*

The automatic identification of semantic term classes is performed in two main steps:

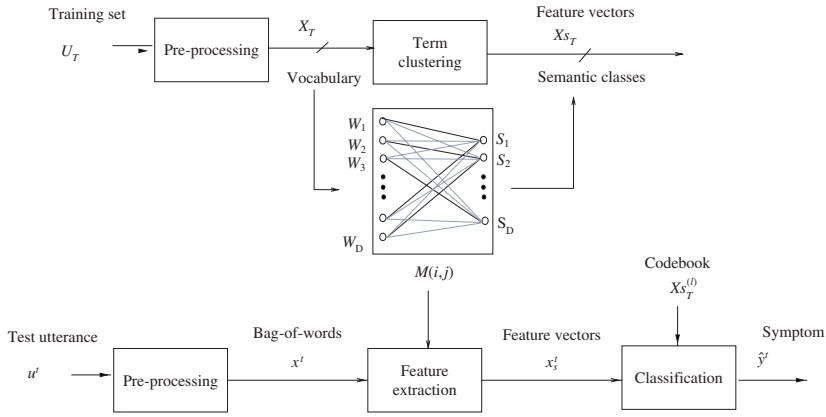


Figure 3.2. *Utterance categorization using the NN algorithm and a term clustering scheme as feature extraction. To address the synonymy problem, a hard clustering scheme can be used. It performs a unique mapping of each term in the vocabulary into a single semantic class. Another possibility is to use fuzzy algorithms to provide a fuzzy or soft association of each term pattern to the output semantic classes through a membership matrix. The association of hard clustering is represented in the figure as bold traces, whereas a fuzzy association is denoted as thin lines*

– the definition of distance metrics that allow us to capture semantic dissimilarities between the terms; and

– the application of a clustering scheme to the matrix of dissimilarities obtained from Step 1.

Both tasks are described in detail in sections 3.4.2 and 3.4.4.

3.4.2. *Semantic term dissimilarity*

As mentioned earlier, the first task related to the extraction of synonym terms through clustering approaches is the calculation of dissimilarity matrices that reflect the semantic differences between the terms.

For that purpose, in a similar way as utterances have been represented in the bag-of-words vectors, an encoding of terms in vector structures has been applied. This allows for the comparison of terms in a vector space.

3.4.2.1. *Term vector of lexical co-occurrences*

As an attempt to explain the main characteristics of semantically related terms two basic criteria have been proposed in the literature:

– *First-order co-occurrence*: two words are semantically similar to the degree that they show patterns of co-occurrence and co-absence in the same texts [WUL 97].

– *Second-order co-occurrence*: two words are semantically similar to the degree that they occur in the neighborhoods of similar words [PIC 99].

The first-order co-occurrence criterion is adequate for text documents, as synonym words are expected to co-occur in the same document. In contrast, semantically related words are not expected to co-occur inside an utterance transcript or sentence, as the semantic variability of a sentence is substantially reduced in comparison to text documents. This fact is also shown in the Internet corpus of utterances. For example, in the category [Operator], the synonym terms *speak* and *talk* are typically "exclusive" inside utterances, such as "*I want to talk to an operator*" or "*I want to speak to an operator.*" However, the second-order co-occurrence criterion is adequate for capturing their semantic connection, as the terms usually occur with similar contexts [1] (e.g. the terms operator, representative, and agent).

1. In this book, the context of a word w_i is defined at the utterance level, i.e. the words that co-occur with w_i in the same utterance.

Consequently, each term in the vocabulary, $w_i \in \mathcal{W}$, has been defined as a D-dimensional vector:

$$w_i = [w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(D)}] \quad [3.6]$$

whose components indicate the relative frequency of occurrence of the term w_i in the same utterances with the other D–1 terms in the vocabulary:

$$w_i^{(j)} = \frac{N(w_i, w_j)}{\sum_{k \neq i} N(w_i, w_k)} \quad [3.7]$$

where $N(w_i, w_j)$ denotes the total number of times that w_i and w_j co-occur.

As an example, the co-occurrence vector for the word *operator* (for occurrence frequencies larger than 0) is the following:

> *C[operator](%)*: *need*(15%), *talk*(11.2%), *speak*(10.3%), *please*(9.9%), *want*(5.1%), *give*(3.8%), *get*(2.5%), *date*(1.7%), *help*(1.7%), *agent*(1.7%), *Internet*(1.7%), *human*(1.2%), *real*(1.2%), *let*(1.2%), *service*(1.2%), *not*(1.2%), *id*(0.8%), *technical*(0.8%), *line*(0.8%).

3.4.2.2. *Metric of dissimilarity*

Given the term vector representation, one of the popular distance functions which can be used to compute the dissimilarities between term vectors is the Euclidean distance:

$$d_{\text{euc}}(w_i, w_j) = ||w_i - w_j|| = \sqrt{\sum_{k=1}^{D} \left(w_i^{(k)} - w_j^{(k)}\right)^2} \quad [3.8]$$

However, one important observation using the Euclidean distance is that relevant terms, occurring in the same contexts of the terms that are to be compared, contribute more

significantly to the distance between the terms. This fact can be illustrated with the utterance vectors for the terms *talk* and *speak* (Table 3.1).

	Agent	*Cable*	*Give*	*Red*	...
w_i: *talk*	0.2	0.08	0.005	0.003	...
w_j: *speak*	0.18	0.07	0.006	0.001	...

Table 3.1. *Comparison of the terms w_i (talk) and w_j (speak) - table rows. The term that occurs more often in their neighborhoods is agent (first column). However, by applying the Euclidean distance, this term contributes a significant increment to the total distance between talk and speak in comparison with other terms in the table columns*

As can be observed, the word with the highest co-occurrence patterns with the terms *talk* and *speak* is *agent*. However, the contribution of this term to the total distance $d_{\text{euc}}(talk, speak)$ is $((0.2 - 0.18)^2 = 0.0004)$. This value is substantially larger than the contributions of other terms with occasional co-occurrences in the contexts of *talk* and *speak*. For example, the contribution due to the term *give* is $(0.003 - 0.001)^2 = 0.000004$.

Therefore, a normalization of the Euclidean distance has been defined, according to the following equation:

$$\overline{d_{\text{euc}}}(w_i, w_j) = \sqrt{\sum_{k=1}^{D} \left(\frac{w_i^{(k)} - w_j^{(k)}}{1 + \min(w_i^{(k)}, w_j^{(k)})} \right)^2} \qquad [3.9]$$

The larger the value of the minimum $\min(w_i^{(k)}, w_j^{(k)})$, the more important is the term w_k to explain the similarities between w_i and w_j. Thus, a higher normalization score in this case decreases the distance between the terms w_i and w_j.

3.4.3. *Term vector truncation*

An extension of the term vector definition in equation [3.6] was motivated by the different patterns of co-occurrences and their subsequent effects on the semantic term dissimilarities. As suggested in section 3.4.2, while a few terms "explain" the semantic behavior and affinity of two terms w_i and w_j being compared, many other terms show occasional occurrences with these terms. This observation in the term (co-occurrence) vectors can be interpreted as an underlying noise pattern with a certain impact in the calculation of distances.

In consequence, a new term co-occurrence vector has been defined,

$$w_i' = [w_i'^{(1)}, w_i'^{(2)}, \ldots, w_i'^{(D)}] \qquad [3.10]$$

by setting a minimum co-occurrence threshold $N_{\text{th}} = 1\%$. Such words, w_k, whose co-occurrence value in the word vector w_i do not exceed this minimum co-occurrence threshold have been considered as "noise words" and were discarded from the original term vector.

$$w_i'^{(j)} = \begin{cases} w_i^{(j)}, & \text{if } w_i^{(j)} > 0.01 \\ 0, & \text{otherwise} \end{cases}$$

Then, the new co-occurrence values have been then recalculated so that the co-occurrence values w_i' build another random variable, i.e. $\sum_{k \neq i} w_i'^{(k)} = 1$. The new vector can be formulated as follows:

$$w_i'^{(j)} = \frac{w_i'^{(j)}}{\sum_{k=1}^{D} w_i'^{(k)}} \qquad [3.11]$$

As can be observed, the important co-occurrences in w_i are placed with higher emphasis with respect to the original vector. As an example, the co-occurrence vector for the term *Operator* is modified as follows:

C[operator](%): *need*(15%), *talk*(11.2%), *speak*(10.3%), *please*(9.9%), *want*(5.1%), *give*(3.8%), *get*(2.5%), *date* (1.7%), *help*(1.7%), *agent*(1.7%), *Internet*(1.7%), *human*(1.2%), *real*(1.2%), *let*(1.2%), *service*(1.2%), *not*(1.2%), *id*(0.8%), *technical*(0.8%), *line*(0.8%), ...

after term vector truncation:

C[operator](%): *need*(21.24%), *talk*(15.86%), *speak* (14.59%), *please*(14.02%), *want*(7.22%), *give*(5.38%), *get*(3.54%), *date*(2.41%), *help*(2.41%), *agent*(2.41%), *Internet*(2.41%), *human*(1.69%), *real*(1.69%), *let*(1.69%), *service*(1.69%), *not*(1.69%).

3.4.4. *Term clustering*

Finally, a clustering algorithm has been applied to the matrix of term dissimilarities (equation [3.9]) to extract clusters of similar terms. By denoting $\mathcal{S} = \{S_i\}_{i=1}^{D'}$, the set of synonym term clusters, a "hard" clustering algorithm is equivalent to an implicit mapping function Fc that uniquely maps each lexicon term to one of the clusters in $\mathcal{S}$:

$$Fc : \mathcal{W} = \{w_i\}_{i=1}^{D} \rightarrow \mathcal{S} = \{S_i\}_{i=1}^{D'}, \qquad D' < D \qquad [3.12]$$

In this work, the clustering algorithm that extracts semantic clusters is based on the complete link criterion for merging a pair of candidate clusters. As described in Chapter 1, the complete link clustering algorithm requires all elements in a cluster to be less distant from each other than the threshold distance specified by the user. The reason for the selection of the complete link merging criterion is that all (synonym) words inside a cluster should be strictly related to each other. However, the implemented clustering algorithm differs from the complete linkage clustering in the selection for the next pair of clusters to be merged at the next step. This

is based on a single link criterion (the clusters with the closest patterns are selected instead). The algorithm is described in the following steps:

1. **Input:** Matrix of dissimilarities between the terms.

2. **Output:** Set of word clusters of semantically related words, $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$.

3. **Initialization:** Place each term w_i in its own cluster: $S_i = \{w_i\}$.

4. Find the terms w_i and w_j with minimum distance, $d_{\min}$ in the similarity matrix.

4.1. **Merging step:** If the terms w_i and w_j, which are found in Step 4, belong to different clusters, $c_a^{(i)}$ and $c_b^{(j)}$, and the distance of the farthest elements in c_a and c_b is lower than the specified threshold distance, $d\text{th}$ ($d_{\max}(c_a, c_b) \leq D\text{th}$) merge clusters c_a and c_b.

4.2. Update dissimilarity matrix, so that $d_{ij} = d_{ji} = \infty$.

5. Repeat from Step 4 until all terms remain assigned to a single cluster or no pair of terms, w_i and w_j, can be found whose clusters are less distant than $D\text{th}$.

As shown in Figure 3.2, the clustering solution can be represented as a binary membership matrix, $M_{[DxD']}$.

$$M[i,j] = \begin{cases} 1, & \text{if } w_i \in S_j \\ 0, & \text{otherwise} \end{cases}$$

where each binary element $M[i,j]$ denotes the "hard" membership of the input lexicon term w_i in the output semantic cluster S_j.

Tables 3.2 and 3.3 show two examples of the semantic clusters (with more than one member) obtained by the term clustering algorithm with $d\text{th} = 0.3$ and 0.5. An observation from these tables explains the relationships of the terms inside each extracted cluster. The major objective of

Semantic classes (example with threshold distance 0.3)
talk speak
agent customer representative support tech
operator human person
someone somebody
security suite
user name
email mail outlook
website site
virus protection
cancel schedule date
ping quality
picture sound
software c-d
parental control
i-p d-n-s
box converter
error message
cancel schedule date
connect run
modem router trouble network system p-c figure
speed company installation pipeline
d-s-l kit
cable t-v line
receive check
computer information
channel homepage
turn see
service phone
hook put
say keep
signal response

Table 3.2. *Example of classes of semantically related terms extracted with a clustering threshold distance* $d\mathrm{th} = 0.3$

term clustering was to identify semantically related terms, which co-occur with similar contexts due to their semantic equivalence. Thus, they can replace each other in an utterance with no semantic change. Good examples of synonyms successfully identified by the algorithm are the pairs *(speak, talk), (somebody, someone), (mail, Email),* and *(website, site)*.

Semantic classes (example with threshold distance 0.5)
talk speak
operator human
agent customer representative support person tech
website site
email mail outlook signal
parental control
security suite virus protection
page webpage
error message
program software c-d antivirus
someone technical somebody
installation assistance pipeline
channel homepage
cancel schedule date make confirm
help setup
buy purchase
question user name
i-p ip d-n-s
receive send check
box converter Microsoft
d-s-l kit
router laptop Ethernet
go tell
install call
say keep use
put hookup
set add
connect hook run start
firewall domain
screen light
know reconnect
mailbox configuration
service phone day
turn see
download scan
come disconnect leave
something card yahoo
game m-s-n
house room
net code
disk communication
web a-o-l

Table 3.3. *Example of classes of semantically related terms extracted with a clustering threshold distance* $d\mathrm{th} = 0.5$

However, not only synonym terms have been identified by the algorithm. Other meaningful components that have been also extracted as a "side product" are frequent term bi-grams. In contrast to uni-grams, bi-grams are pairs of consecutive words (w_1, w_2) whose high co-occurrence pattern evidences significant conditional statistical dependencies, i.e. $P(w_2|w_1) \geq P(w_2)$. Some examples are the pairs *(technical, support), (antivirus, security), (parental, control), (error, message)*, and *(user, name)*. This observation is related to the definition of words in term vectors of term co-occurrences. In fact, a bi-gram pair of terms that co-occur frequently in the training corpus should have close co-occurrence vectors. This happens because the words that occur in the contexts of both bi-gram terms are also similar.

3.4.5. *Feature extraction and utterance feature vector*

By applying clustering of semantic classes, the utterance vector needs to be redefined with the new extracted features (semantic classes). Given $\mathcal{S}$, the set of semantic clusters extracted through the term clustering scheme, and M, the cluster membership matrix that encodes the term clustering output, the feature extraction that transforms the input utterance vector x_i into a vector of semantic features x_i^s is defined as follows:

$$x_i^s = [x_i^{s_1}, x_i^{s_2}, \ldots, x_i^{s'_D}] = x_i \cdot M \qquad [3.13]$$

In other words, each binary component of the utterance feature vector $x_{i_j}^{(s)}$ denotes the presence/absence of at least one member of the semantic cluster S_j in the pre-processed utterance $u_i^{(l+p)}$.

3.4.6. *Evaluation*

The symptom categorization of user utterances has been evaluated on an accuracy basis. The accuracy achieved by the classification algorithm is calculated by comparing the automatic symptoms (labels) assigned by the algorithms to the input test utterances with the corresponding manual labels, which are available for test purposes. The accuracy is then defined as the ratio of correctly classified utterances over the total number of utterances in the test set:

$$\text{Accuracy} = \frac{\#\text{correctly classified utterances}}{\#\text{utterances in test set}} \qquad [3.14]$$

Although typical evaluation criteria used for the categorization of texts are the microaverage F metrics, but in this work, the accuracy has been preferred to the microaverage metric. In fact, accuracy scores have been considered as a better indicator of the percentage of correctly routed calls due to the symptom categorization module of the Spoken Language Dialogue Systems (SLDS).

Note that the microaverage F1 corresponds to the accuracy of uniformly distributed classes. However, for skewed distributions, a classifier may produce high accuracy values but low F1 scores. This occurs when a high ratio of the correct classifier predictions are concentrated in the most frequent classes. In this work, since a frequent class indicates a frequent reason for calls, high accuracy values are still good indicators for the average performance of the SLDS. First, the performance of the supervised classification algorithm (Figure 3.1) has been compared with the semi-supervised classification using term clustering. Some examples of the accuracy results obtained with six different labeled seeds initializations are shown in Figures 3.3–3.5. The vertical axes refer to the accuracy scores, whereas the horizontal

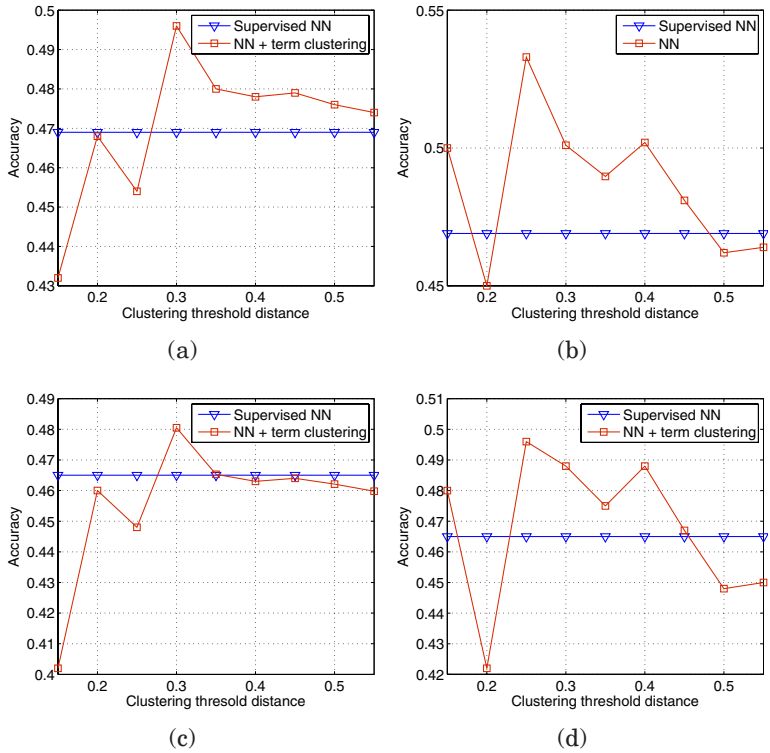


Figure 3.3. *Comparison of accuracy scores of the supervised classifier versus the semi-supervised approach with term clustering, using two different initializations*

axes indicate different distance thresholds applied to the term clustering algorithm for the semi-supervised approach.

Each pair of plots 3.3(a,b), 3.3(c,d), 3.4(a,b), 3.4(c,d), 3.5(a,b), and 3.5(c,d) are referred to identical labeled seeds. The difference in the left plots with respect to the right plots is the term vector representation applied to the term clustering in the semi-supervised approach. The left plots are referred to the basic term vector definition in equation [3.11], whereas the right plots are referred to the term vector definition after truncation (equation [3.10]).

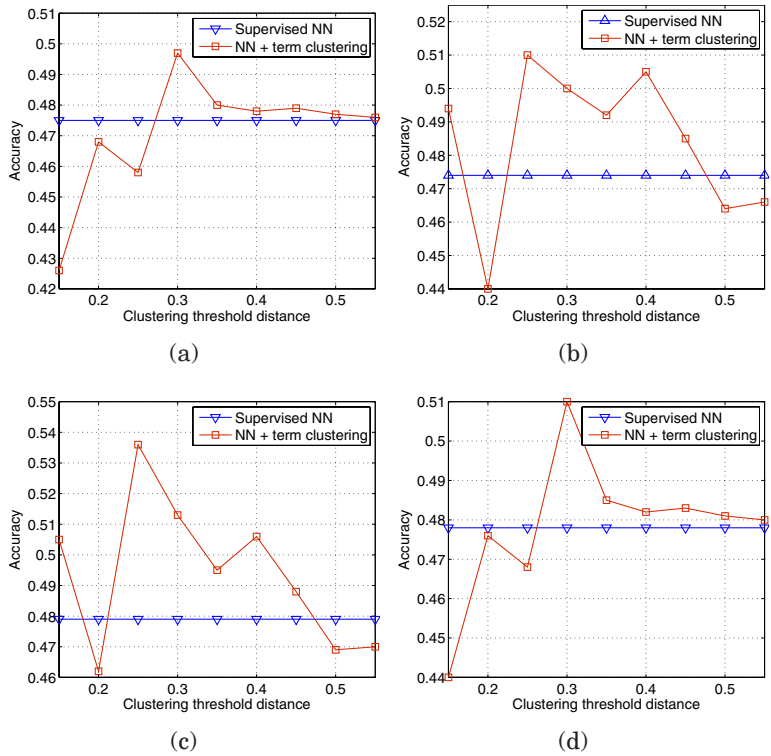


Figure 3.4. *Comparison of accuracy scores of the supervised classifier versus the semi-supervised approach with term clustering, using two different initializations*

Figure 3.6 shows the mean accuracy values of the supervised and semi-supervised classification schemes over 20 different prototype labeled seeds initializations. Standard deviations are indicated as shadowed areas around their respective mean curves. Figure 3.6a shows the classification accuracy of the supervised scheme and the semi-supervised approach using the term vector definition of equation [3.11]. Figure 3.6b refers to the term vector definition in equation [3.10]. Although the performance of the semi-supervised classifier depends on the threshold distance $d\text{th}$ applied to

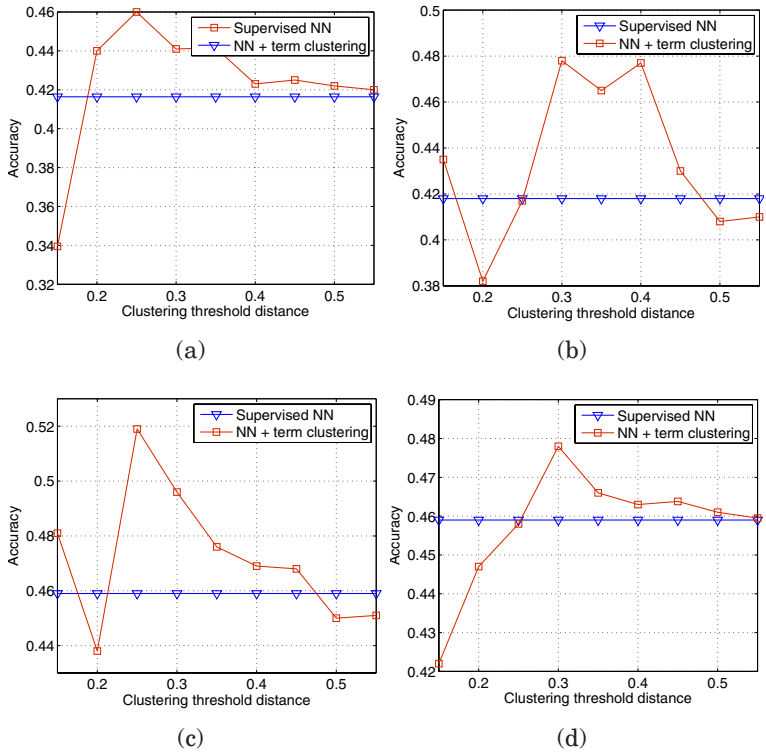


Figure 3.5. *Comparison of accuracy scores of the supervised classifier vs. the semi-supervised approach with term clustering, using two different initializations*

term clustering, some improvements can be observed for different values dth, up two 2%, using the definition of a term co-occurrence vector in equation [3.11] and 5.5% if word vector truncation is applied.

3.5. Disambiguation

If applied to utterance bag-of-words or feature vectors, the NN classifier rejects a considerable number of ambiguous utterances for which several candidate prototypes are found.

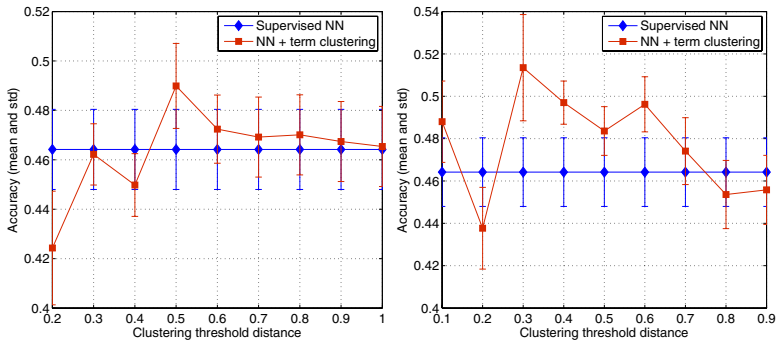


Figure 3.6. *Mean accuracy values over 20 labeled seeds initializations achieved by the supervised NN classifier vs. the semi-supervised approach with term clustering. Standard deviations are indicated as shadowed areas around their respective mean curves. (a): Supervised NN based on bag of words features vs. semi-supervised using semantic class features. (b): Supervised vs. semi-supervised by incorporating word vector truncation*

Candidate prototypes are labeled utterance vectors in $\mathcal{X}_T^{(l)}$, which share maximum proximity to the input utterance. This happens especially when the similarity metric between the vectors results in integer values, as the overlap distance in equation [3.4]. A disambiguation module (Figure 3.7) has therefore been devised to resolve the mentioned ambiguities and to map an ambiguous utterance to one of the output categories.

First, the terms that cause the ambiguity are identified and stored in a list of competing terms. As an example, consider the utterance *I want to get the virus off my computer*. After pre-processing and hard term clustering, this utterance results in the feature set *computer get off virus*. The feature vector has minimum distance to the prototypes *computer freeze* [CrashFrozenComputer] and *install protection virus* [Security]. The competing terms that produce the ambiguity are in this case the words *computer* and *virus*. Therefore, in this study, the disambiguation among prototypes

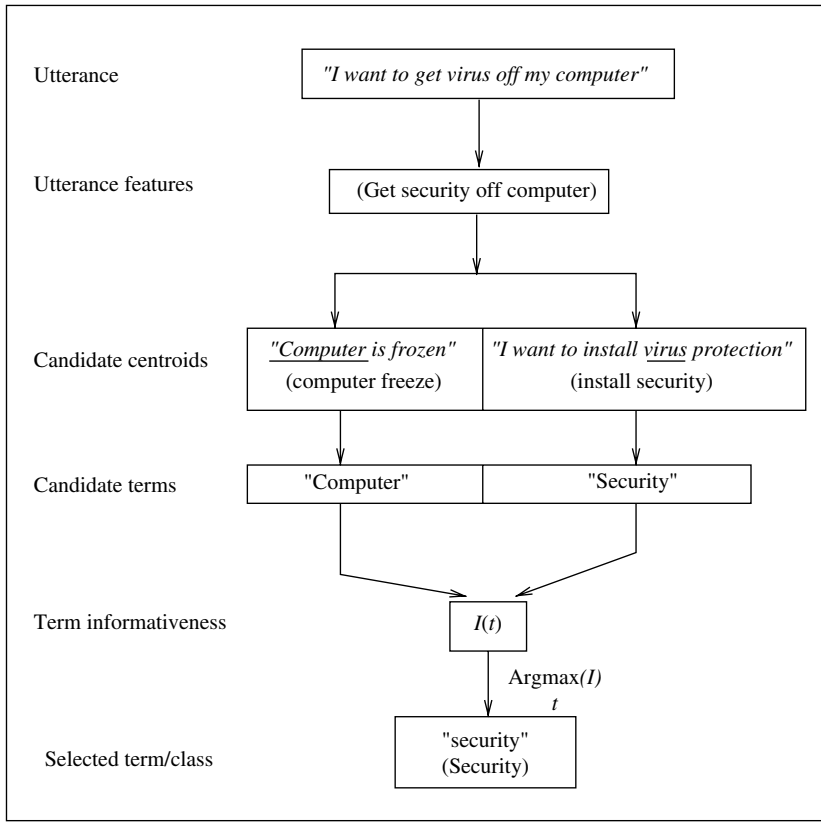


Figure 3.7. *Disambiguation scheme used for utterance categorization*

(or categories) is equivalent to a disambiguation among competing terms. For this reason, as a further means of disambiguation, the *informativeness* of a term w_i has been estimated as shown in equation [3.15]:

$$I(w_i) = -\left(\log(Pr(w_i)) + \alpha \cdot \log\left(\sum_{\substack{j \\ L_j=N}} N(w_i, w_j) Pr(w_j)\right)\right) \quad [3.15]$$

where $Pr(w_i)$ denotes the maximum-likelihood estimation for the probability of the term w_i in the training corpus, and L_j refers to the part of speech (POS) tag of w_j, where N refers to nouns. POS tags have been extracted by means of the Standford POS tagger [TOU 00].

As can be inferred from equation [3.15], two main factors are taken into account to estimate the relevance of a word for the disambiguation:

– the word probability and

– the term co-occurrence with frequent nouns in the corpus.

The underlying assumption that justifies this second factor is that words representative of problem categories are mostly nouns and appear in the corpus with moderate frequencies. The parameter α is intended to control the trade-off between the two factors. Reasonable values are in the range of ($\alpha \in [1, 2]$) placing some emphasis on the second factor; a value of $\alpha = 1.6$ has been selected in the experiments, although other values in the mentioned range may yield a similar performance.

Finally, the term with the highest informativeness is selected among the competitors, and the ambiguous utterance vector is matched to the corresponding prototype or category.

3.5.1. *Evaluation*

The disambiguation scheme has been also individually evaluated. For this purpose, two cases of ambiguous utterances need to be distinguished:

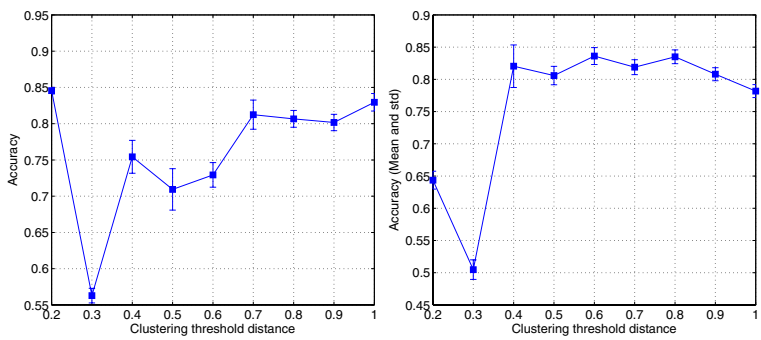
(a) *resolvable ambiguities*: ambiguous utterances that can be resolved using the disambiguation scheme because one of their "candidate prototypes" corresponds to the true (manual) category of the input utterance and

(b) *unresolvable ambiguities*: ambiguous utterances that cannot be resolved, because no candidate prototype corresponds to the category of the utterance; thus, no disambiguation strategy can be applied in this case.

To assess the disambiguation performance of the disambiguation scheme, only resolvable ambiguities have been considered. A resolvable ambiguity is said to be "correctly resolved," if the *winning prototype* is the one with the same category as the manual category of the utterance. The disambiguation accuracy is then formulated as follows:

$$\text{Disambiguation accuracy} = \frac{\#\text{Resolvable ambiguities correctly resolved}}{\#\text{resolvable ambiguities}} \quad [3.16]$$

Figure 3.8 shows the mean disambiguation accuracy of ambiguous utterance vectors over 20 different prototype initializations, with respect to the variable $d\text{th}$ used in the term clustering algorithm. Figure 3.8a refers to the basic term



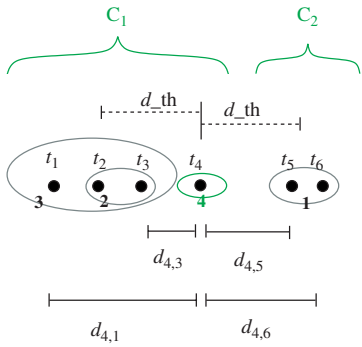
(a) (b)

Figure 3.8. *Disambiguation accuracies obtained by the disambiguation scheme. (a): using term Clustering, (b): term clustering and term vector truncation*

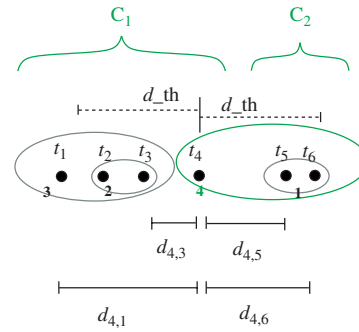
clustering strategy without vector truncation. Figure 3.8b corresponds to term clustering and term vector truncation.

As can be observed, the disambiguation performance varies with the values of $d\text{th}$ used in the term clustering algorithm. It reaches a minimum accuracy score at $d\text{th} = 0.3$, both with and without vector truncation. From this point, it shows an increasing trend with the $d\text{th}$ variable, which is more abrupt if vector truncation is applied before term clustering. This trend in the disambiguation performance is strongly associated with the clustering scheme (see Figure 3.9).

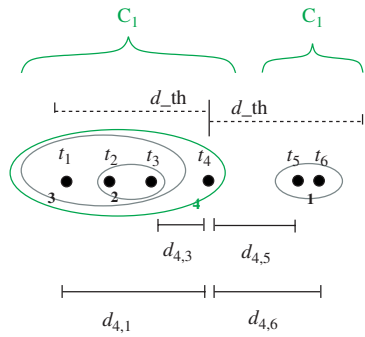
Ideally, each extracted cluster should be composed of terms that are mostly relevant for a topic category. However, as illustrated in Figure 3.9c (case 2), certain values of $d\text{th}$ produce clusters of terms descriptive for different categories. For example, terms $t_1 - t_4$ in the figure are assumed to be relevant for a certain topic category, C_1. Likewise, terms t_5 and t_6 are assumed to be representative for a second category, C_2. The order in which clusters are extracted by the algorithm is denoted by the labels (1–4). Terms t_1, t_2, t_3, t_5 and t_6 are allocated in clusters 3 and 4 in all three cases. These allocations correspond to their relevant categories, C_1 and C_2. In contrast, different cluster allocations can be found for the term t_4, depending on $d\text{th}$. If the value of d_{th} is too small (case 1), the complete link criterion is not fulfilled for any of the two existing clusters and t_4 composes an individual cluster. A small increase in $d\text{th}$ such that t_4 *only* satisfies the merging criterion for cluster 1 causes the term t_4 to be attached to such cluster (case 2). This situation can lead to an increment in ambiguous utterances due to a possible overlapping of terms in some of the labeled prototypes. A further increment in $d\text{th}$, such that $d_{4,1} \leq d\text{th}$ (the merging condition for both clusters 1 and 3 is fulfilled) allows the attachment of t_4 to cluster 1. t_4 is not assigned to cluster 1 because the single link search criterion of this cluster algorithm gives the priority to cluster



(a) Case 1: $d_{th} < d_{4,1}$ and $d_{th} < d_{4,6}$. Term $t4$ cannot be assigned to any of the two existing clusters. Thus, it composes an individual cluster



(b) Case 2: $d_{th} < d_{4,1}$ but $d_{th} > d_{4,6}$. Term $t4$ cannot be merged with cluster 3 but it is attached to cluster 1



(c) Case 3: $d_{th} > d_{4,1}$ and $d_{th} > d_{4,6}$. Because $d_{4,3} < d_{4,5}$, term $t4$ is attached to cluster 3, which contains $t3$

Figure 3.9. *Illustration of term clustering with different values of the threshold distance with an hypothetical uni-dimensional example*

3 (as $d_{4,3} < d_{4,5}$). In this situation, the cluster configuration is consistent with the (real) sets of descriptive terms for the categories C_1 and C_2.

The previous explanation can also be associated with the true semantic clusters extracted by the term clustering algorithm. Case 2 in Figure 3.9 is equivalent to the cluster output for $d\text{th} = 0.3$ (Table 3.2). The terms *router* and *modem* are allocated into the same cluster by the term clustering algorithm. However, these terms commonly occur with different topic categories, [Homenetwork] and [Modem], respectively. Typical utterances for the mentioned categories are "wireless router" and "modem." They are often applied as the labeled prototypes for these categories. Since *modem* is the most frequent term in the cluster, it is selected as the cluster's representative term. Thus, the prototypes after term clustering are (wireless modem) and (modem). Any input utterance whose bag-of-words vector only overlaps *modem* or *router* with the prototypes results in an ambiguous vector, as both terms derive into a single cluster representative (modem). Since the categories [modem]

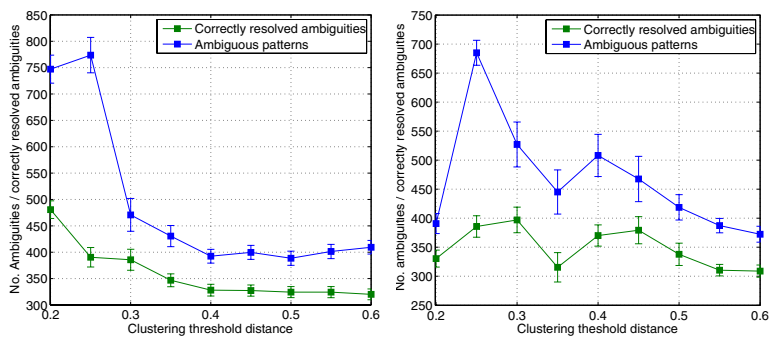


Figure 3.10. *Number of ambiguous patterns found by the utterance classification scheme. (a): with term clustering, (b): with term clustering and term vector truncation*

and [homenetworks] occur with relatively high frequencies, this "error" in term clustering yields a significant amount of ambiguities, as shown in Figure 3.10. Furthermore, although such an ambiguous utterance may be a resolvable ambiguity as defined earlier, it cannot be resolved by the disambiguation scheme since the conflictive terms are "identical" (modem). In other words, the term's informativeness criterion is not sufficient to discriminate among prototypes. In these cases, the decision for the winning category is randomly performed. As can be observed in Figure 3.10, while the number of ambiguities considerably increases for $d\text{th} = 0.3$, only a small

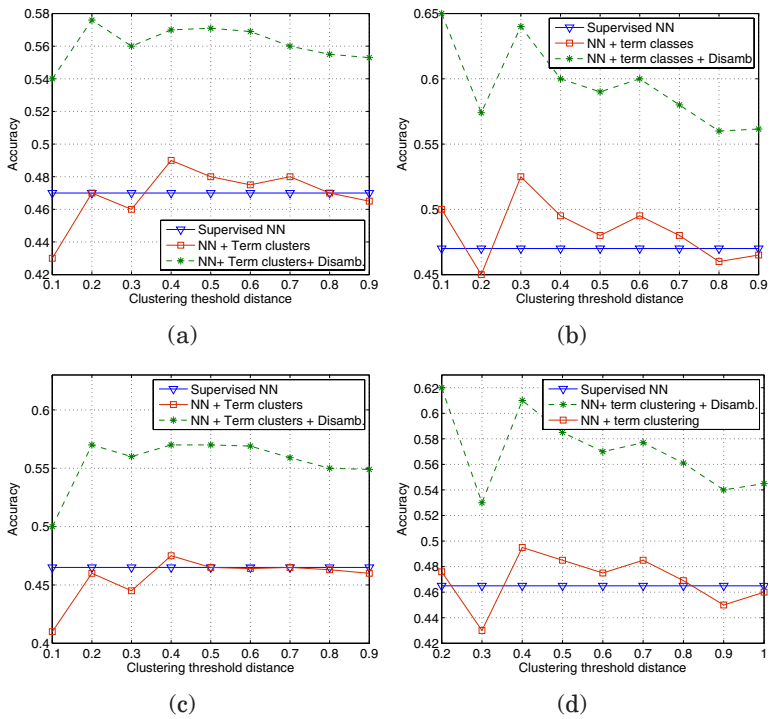


Figure 3.11. *Comparison of accuracy scores of the supervised classifier vs. the semi-supervised approach before and after disambiguation using two different prototype initializations*

increment is attained on the number of ambiguities correctly resolved.

Final accuracy scores of the semi-supervised classification before and after disambiguation versus the supervised classification can be observed in Figures 3.11–3.13. As in section 3.4.6, the semi-supervised evaluation in these figures is referred to six different labeled seed initializations. The left plots in these figures are referred to the basic term vector definition in equation [3.11], whereas the right plots are referred to the term vector definition after truncation (equation [3.10]).

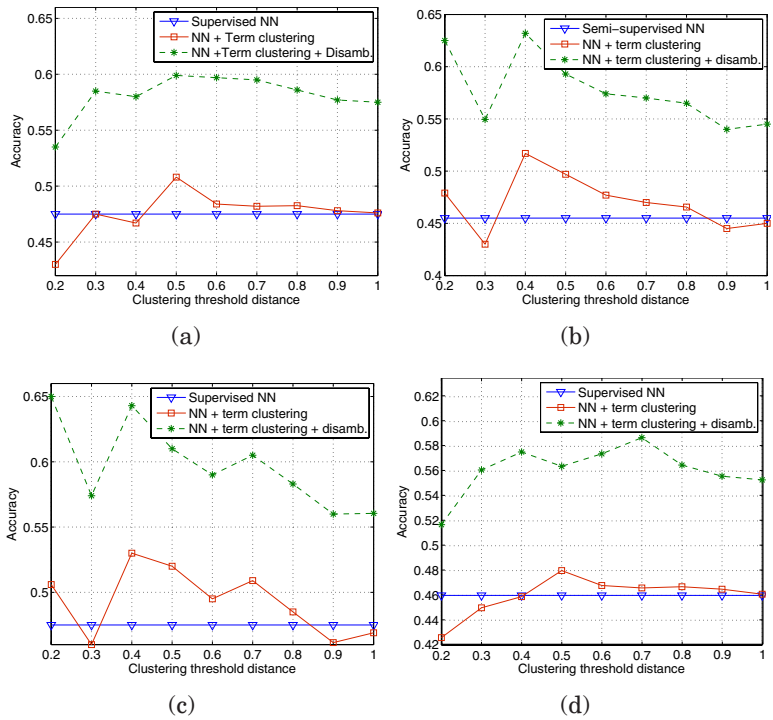


Figure 3.12. *Comparison of accuracy scores of the supervised classifier vs. the semi-supervised approach before and after disambiguation using two different prototype initializations*

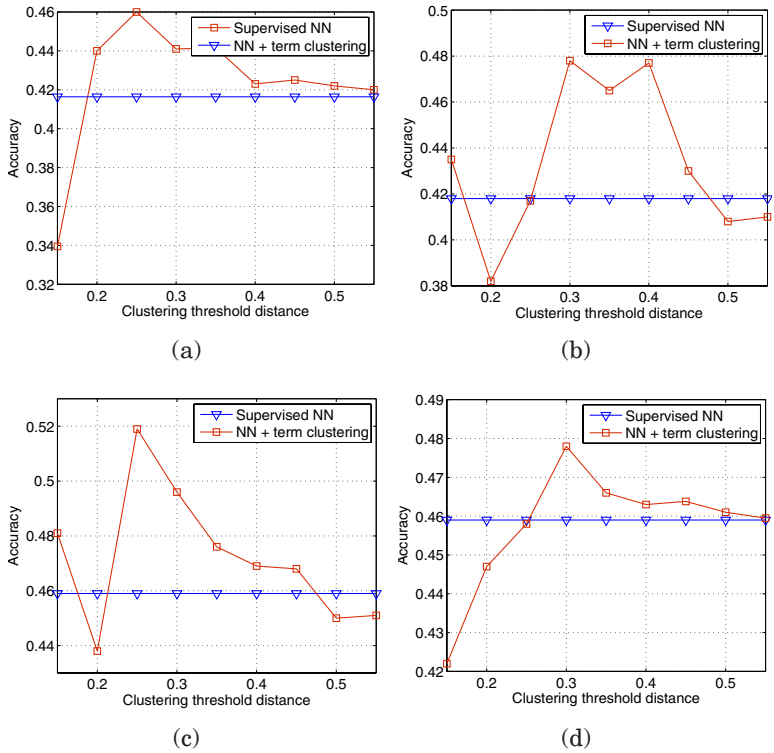
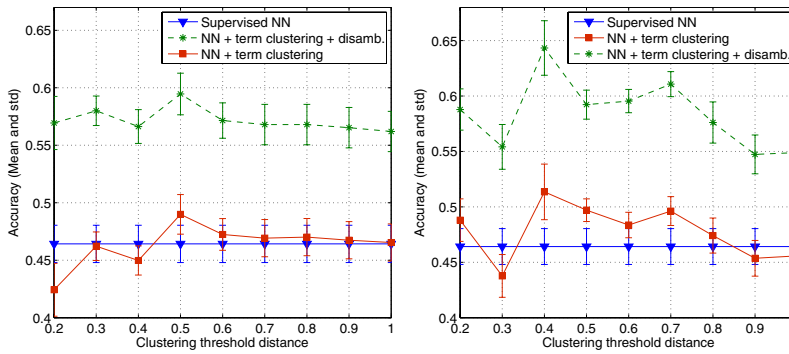


Figure 3.13. *Comparison of accuracy scores of the supervised classifier vs. the semi-supervised approach before and after disambiguation using two different prototype initializations*

Figure 3.14 shows the mean accuracy scores obtained by the supervised algorithm versus the semi-supervised approach (before and after disambiguation) over 20 different initializations.

Standard deviations are indicated as shadowed areas around their respective mean curves. Figure 3.14a shows the classification accuracy of the supervised scheme and the semi-supervised approach with disambiguation, using the term vector definition of equation [3.11]. Figure 3.14b refers to the term vector definition in equation [3.10]. As can



(a) (b)

Figure 3.14. *Mean accuracy values achieved by the supervised classifier vs. the semi-supervised approach before and after disambiguation. Standard deviations are indicated as shadowed areas around their respective mean curves. (a): Supervised NN based on bag of words features vs. semi-supervised using semantic class features. (b): Supervised vs. semi-supervised by incorporating word vector truncation*

be observed, by incorporating the disambiguation scheme, the semi-supervised approach achieves notable improvements with respect to the supervised classifier, up to 65% accuracy.

3.6. Summary

In this chapter, a semi-supervised classification scheme has been described. It takes advantage of the availability of unlabeled data by means of unsupervised clustering applied to the feature space of words. This approach is based on the hypothesis that synonymy is one important source of variability in the utterances for a single symptom. Thus, by identifying clusters of semantically related terms, the vocabulary available to the classifier can be "augmented." One of the main contributions in this work is the novel definition of a word in a vector space model so that any distance metric applied to word vectors can capture the degree

of *semantic dissimilarity* between the words. When using a term vector model with term truncation (so that only the most important words in the context of another word w_i account for the definition of the ith word vector), the semi-supervised approach has shown average accuracy improvements up to 5% with respect to the basic supervised classifier.

This has been motivated by the observation that a significant amount of ambiguities resulted from the proposed classification schemes. Ambiguities are basically due to the co-occurrences of words representative for different categories.

Chapter 4

Semi-Supervised Classification Using Pattern Clustering

4.1. Introduction

In the previous chapter, a semi-supervised approach has been described, which gained advantage of unlabeled data by means of clustering. A minimum labeled seed was fed to a supervised classifier. Then, the lexicon features in these initial labeled seeds were automatically expanded through a set of synonym groups found by the clustering algorithm.

In this chapter, a new alternative to semi-supervised algorithm is introduced. In a similar way as the approach described in the previous chapter, clustering is also used to "augment" the small labeled seeds. However, in contrast to the previous approach, the cluster assumption is now applied to obtain groups of data instances instead of features. This *cluster principal assumption* – underlying class labels should naturally fall into clusters – has been frequently applied to other works in the semi-supervised machine learning (ML) literature [BLU 01].

Semi-Supervised and Unsupervised Machine Learning: Novel Strategies Amparo Albalate and Wolfgang Minker

Following the clustering step, the labeled seeds have been used to tag the clusters in such a way that the initial labels are augmented to the complete clustered data. In the previous chapter, an explicit labeling step was absent. However, by assuming no overlap of the terms from different categories inside each extracted group, an implicit labeling of semantic clusters was performed. The clusters that overlapped at least one term with one of the labeled prototypes remained labeled with the class label of the prototype. In this chapter, the labeling task has been comprehensively defined and solved as a separated optimization problem.

Finally, because clustering is performed on a data space rather than on a feature space, the algorithm can be generalized to any type of feature. For this reason, besides a dataset of utterances, the strategy has been tested on a number of real and simulated datasets, most of them from the University of California Irvine (UCI) dataset repository.

4.2. New semi-supervised algorithm using the cluster and label strategy

In essence, the semi-supervised classification described in this chapter differs from previous works in which the clustering and labeling tasks are clearly distinguished as two independent optimization problems. First, a clustering algorithm extracts the cluster partition that maximizes an *internal* – data driven – quality objective. Then, an optimum cluster labeling, given the labeled seed and the cluster partition of the data, is formulated as an optimum assignment problem, which has been solved using the Hungarian algorithm.

4.2.1. *Block diagram*

The block diagram of the semi-supervised approach is illustrated in Figure 4.1. It consists of two main parts: *cluster and label* and *supervised classification*.

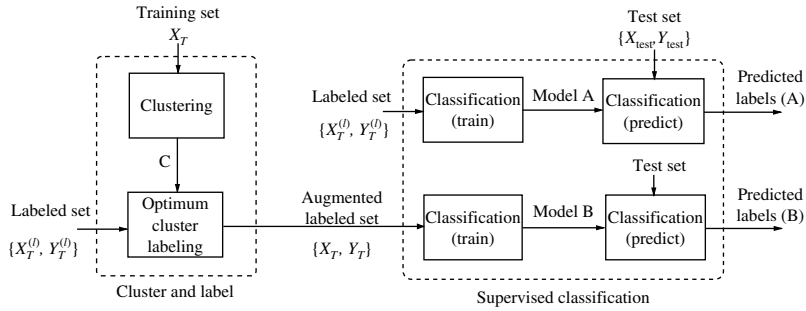


Figure 4.1. *Block diagram of the semi-supervised algorithm*

4.2.1.1. *Dataset*

First, the data are divided into a test set ($\sim 10\%$) and a training set ($\sim 90\%$). Let

$$\mathcal{X}_T = \{x_i\}_{i=1}^{l+u}, \qquad \forall x_i \in \mathcal{R}^N$$

denote the training data, where l refers to the labeled seed size and u the size of the unlabeled seed. The total training size considering both labeled and unlabeled portions is thus $l + u$. This training set can be divided into two disjoint subsets:

$$\mathcal{X}_T = \mathcal{X}_T^{(l)} \cup \mathcal{X}_T^{(u)}, \qquad \mathcal{X}_T^{(l)} = \{x_i\}_{i=1}^{l}, \qquad \mathcal{X}_T^{(u)} = \{x_i\}_{i=l+1}^{l+u}$$

where $\mathcal{X}_T^{(u)}$ denotes the subset of unlabeled patterns in $\mathcal{X}_T$ and $\mathcal{X}_T^{(l)}$ the labeled portion of $\mathcal{X}_T$ for which the corresponding set of labels $\mathcal{Y}_T^{(l)}$ is (assumed to be) known. Hence, the initial labeled set can be expressed as the set of pairs:

$$\{\mathcal{X}_T^{(l)}, \mathcal{Y}_T^{l}\} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\}$$

with the set of (manual) labels, $\mathcal{Y}_T^{(l)} = \{y_i\}_{i=1}^{l}$. The block diagram of the algorithm is shown in Figure 4.1. The two main parts in the semi-supervised strategy are the *cluster and label* block, and the *supervised classification* in which a supervised model is trained with the augmented dataset from

the *cluster and label* block. The test set is denoted as the pairs $\{\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}}\}$ of test instances and labels. A fully supervised version is also depicted in the upper branch of the supervised classification block. In this case, the supervised classifier is trained with the initial labeled seed: $\{\mathcal{X}_T^{(l)}, \mathcal{Y}_T^l\}$.

4.2.1.2. *Clustering*

The first step in the semi-supervised approach is to find a cluster partition $\mathcal{C}$ of the training data X_T into a set of k disjoint clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, where k is the number of classes. In this work, the partitioning around medoids (PAM) algorithm has been applied. As explained in Chapter 2, the PAM algorithm provides the optimum cluster partition with the minimum *sum of distances* to the cluster medoids. The clustering task can be expressed as an implicit mapping function, F_C, that assigns each training instance into one of the clusters in $\mathcal{C}$:

4.2.1.3. *Optimum cluster labeling*

The labeling block performs a crucial task in the semi-supervised algorithm. Given the set of clusters $\mathcal{C}$ in which the training data are divided, the objective of this block is to find an optimum objective mapping of labels to clusters,

$$\mathcal{L} : \mathcal{C} \rightarrow \mathcal{K}, \quad \mathcal{K} = \{1, 2, 3, \cdots, k\}$$

so that an optimum criterion is fulfilled. Each cluster has been assigned exactly one class label in $\mathcal{K}$. Assuming a fixed ordering of the clusters extracted by the PAM algorithm, $\{C_1, C_2, \ldots, C_k\}$, the objective of the cluster labeling block is to find a *permutation* of the class labels corresponding to the ordered set of clusters. By labeling each cluster with one of the class labels in $\mathcal{K}$, all data instances in the training set $\mathcal{X}_T$ remain implicitly labeled with the class label of their respective clusters. In mathematical terms, the optimum

cluster labeling provides an estimation $\hat{y}_i$ of the optimum class label for each observation x_i in the training set:

$$\hat{y}_i = L(x_i) = L(F_C(x_i)), \quad \forall x_i \in \mathcal{X}_T$$

As a result of cluster labeling, the initial labeled seed $\{\mathcal{X}_T^{(l)}, \mathcal{Y}_T^{(l)}\}$ is extended to the complete training data. This augmented labeled set can be expressed as

$$\{\mathcal{X}_T, \mathcal{Y}_T\} = \{x_i, y_i\}_{i=1}^{l} \cup \{x_i, \hat{y}_i\}_{i=l+1}^{u}$$

where $\mathcal{Y}_T$ denotes the set of augmented labels corresponding to the observations in $\mathcal{X}_T$. For training instances in the labeled seed, the available manual labels are selected. For unlabeled patterns, $x_i \in \mathcal{X}_T^u$, the augmented label is the class label estimation $\hat{y}_i$ obtained by the optimum cluster labeling.

The optimum cluster labeling block is explained in more detail in section 4.3.

4.2.1.4. *Classification*

Finally, a supervised classifier is trained with the augmented labeled set $(\mathcal{X}_T, \mathcal{Y}_T)$ obtained after cluster labeling. The learned model is then applied to predict the labels for the test set.

Simultaneously, a fully supervised classification scheme has been compared with the semi-supervised algorithm. In this case, the classifier is directly trained with the initial labeled seed ($\mathcal{X}^{(l)}$, $\mathcal{Y}^{(l)}$). The (supervised) model (denoted as model A in Figure 4.1) is again applied to predict the labels of the test data.

Both semi-supervised and supervised strategies have been evaluated in terms of accuracy, by comparing the predicted labels of the test patterns with their respective manual labels. The evaluation results are discussed in section 4.6.

4.3. Optimum cluster labeling

This section describes the objective criterion applied to the *optimum cluster labeling* and the algorithms used to achieve this optimum.

4.3.1. *Problem definition*

Given the training data, $\mathcal{X}_T = \mathcal{X}_T^{(l)} \cup \mathcal{X}_T^{(u)}$, the set $\mathcal{Y}_T^{(l)}$ of labels associated with the portion $\mathcal{X}_T^{(l)}$ of the training set, the set $\mathcal{K}$ of labels for the k existing classes,[1] and a cluster partition $\mathcal{C}$ of $\mathcal{X}_T$ into disjoint clusters, the optimum cluster labeling problem is to find an objective mapping function, L:

$$L : \mathcal{C} \rightarrow \mathcal{K}, \quad \mathcal{K} = \{1, 2, 3, \cdots, k\}$$

which assigns each cluster in $\mathcal{C}$ to a class label in $\mathcal{K}$, while minimizing the total labeling cost. This cost is defined in terms of the labeled seed $(\mathcal{X}_T^{(l)}, \mathcal{Y}_T^{(l)})$ and the set of clusters $\mathcal{C}$. Consider the following matrix of overlapping products O:

$$O = \begin{pmatrix} n_{i1} & n_{i2} & \cdots & n_{ik} \\ n_{21} & n_{22} & \cdots & n_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ n_{k1} & n_{k2} & \cdots & n_{kk} \end{pmatrix} \qquad [4.1]$$

with constituents n_{ij}, denoting the number of labeled patterns from $\mathcal{X}_T^{(l)}$ with class label $y = i$ that fall into cluster C_j. The labeling objective is to minimize the global cost of the cluster labeling denoted by L:

$$\text{Total cost}(L) = \sum_{C_i \in \mathcal{C}} w_i \cdot \text{cost}\big(L(C_i)\big) \qquad [4.2]$$

1. Although class labels can take any arbitrary value, either numeric or nominal, for simplicity in the formulation and implementation of the cluster labeling problem the k class labels are transformed into integer values ($[1, \ldots, k]$).

where $W = (w_1, \cdots, w_k)$ is a vector of weights for the different clusters. These weights may be used if significant differences in the cluster sizes are observed. In this book, the weights are assumed to be equal for all clusters, so that $w_i = 1, \forall i \in 1 \cdots k$.

The individual of labeling a cluster C_i with class j is defined as the number of samples from class j (in the labeled seed) which fall outside the cluster C_i, i.e.,

$$\text{Cost}\big(L(C_i) = j\big) = \sum_{C_k \neq C_i} n_{j,k} \qquad [4.3]$$

by applying equation [4.3] into the total cost definition of equation [4.2], yields

$$\text{Total cost}(L) = \sum_{C_i \in \mathcal{C}} \sum_{C_k \neq C_i} n_{L(C_i),k} \qquad [4.4]$$

Using a greedy search algorithm, the cost minimization of equation [4.2] requires $k!$ operations (where k denotes the number of clusters/classes). Such a complexity becomes computationally intractable for $k \geq 10$. However, larger number of classes are often involved in real classification problems. In this book, two popular optimization algorithms have been applied to achieve the optimum cluster labeling with substantially lower complexities:

– The Hungarian algorithm by Huhn and Monkres and

– the genetic algorithms (GAs).

Both algorithms require the definition of a cost matrix $\mathbf{C}_{[kxk]}$, whose rows denote the clusters and the columns are referred to class labels in $\mathcal{K}$. The elements $\mathbf{C}_{ij}$ denote the individual costs of assigning the cluster C_i to class label j, i.e. $\mathbf{C}_{ij} = cost(L(C_i) = j)$.

More details about the Hungarian algorithm and GA applied to solve the optimum cluster labeling problem are described in sections 4.3.2 and 4.3.3.

4.3.2. *The Hungarian algorithm*

The Hungarian method is a linear programing approach which solves the assignment problem in $\mathcal{O}(N)^3$ operations [KUH 55, GOL 03]. The algorithm was devised by Huhn in 1955. Its name, "Hungarian," was given after two Hungarian scientists who had previously established a large part of the algorithm's mathematical background.

Although the algorithm's complexity was originally $\mathcal{O}(N^4)$, a more recent version with complexity $\mathcal{O}(N^3)$ was developed by Huhn and Monkres, thus referred to as the *Huhn–Monkres* algorithm.

The algorithm states the assignment problem in terms of *bipartite graphs*. In the following paragraphs, some important notions from graph theory are introduced .

4.3.2.1. *Weighted complete bipartite graph*

DEFINITION 4.1.– Bipartite graph*: a bipartite graph is a graph, $G(\mathcal{V}, \mathcal{E})$, with set of vertices $\mathcal{V}$ and set of edges $\mathcal{E}$, in which two disjoint subsets of $\mathcal{V}$ can be found, $\mathcal{X}$ and $\mathcal{Y}$, with no internal edges connecting two vertices within a subset. Thus, the set of edges $\mathcal{E}$ only connects any vertex in $\mathcal{X}$ with any of the vertices of $\mathcal{Y}$. In the following, a bipartite graph is also denoted $G(\mathcal{X}, \mathcal{Y}, \mathcal{E})$.*

DEFINITION 4.2.– (Complete weighted bipartite graph)*: a complete bipartite graph is a bipartite graph in which, for all pairs (x, y) of vertices from $\mathcal{X}$ and $\mathcal{Y}$, there exists an edge, $e_{xy} \in \mathcal{E}$, that connects x with y.*

DEFINITION 4.3.– (weighted complete bipartite graph)*: a weighted complete bipartite graph is a complete bipartite graph in which each edge, $e_{x,y}$, is assigned to a certain weight $w(x, y)$. Note that a value of $w(x, y) = 0$ is also possible, and the weighted bipartite graph is still complete.*

Figure 4.2 shows three examples of the bipartite graph, the complete bipartite graph, and the weighted complete bipartite graph, respectively.

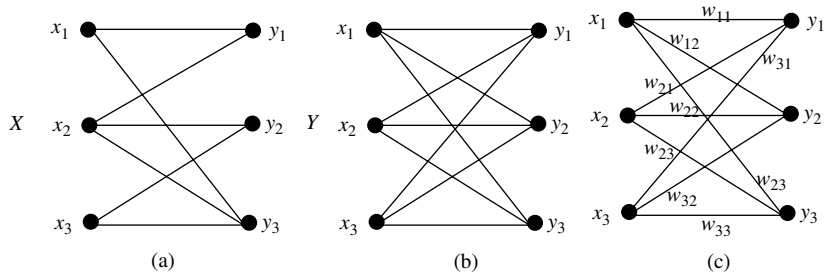


Figure 4.2. *Example of bipartite graphs: (a) bipartite graph with two disjoint subsets, (b) complete bipartite graph, and (c) completed weighted bipartite graph*

4.3.2.2. *Matching, perfect matching and maximum weight matching*

DEFINITION 4.4.– (Matching)*: a matching M, defined on a graph $G(\mathcal{X}, \mathcal{Y}, \mathcal{E})$, is any subset of edges, $M \in \mathcal{E}$, such that no common vertex is shared between different edges in M (Figure 4.3(a)).*

DEFINITION 4.5.– (Perfect matching)*: a perfect matching is a matching M in which each vertex x is connected to a vertex y by an edge in M (Figure 4.3(b)).*

DEFINITION 4.6.– (Maximum weight matching)*: denoting $\mathcal{M}$, the set of all possible perfect matchings in $G(X;Y;E)$, the maximum weight matching is the perfect matching $M' \in \mathcal{M}$ in which the sum of edge weights is maximum:*

$$M' = \underset{M_i \in \mathcal{M}}{\operatorname{argmax}} \Big(\sum_{e_{xy} \in M_i} w_{xy} \Big) \qquad [4.5]$$

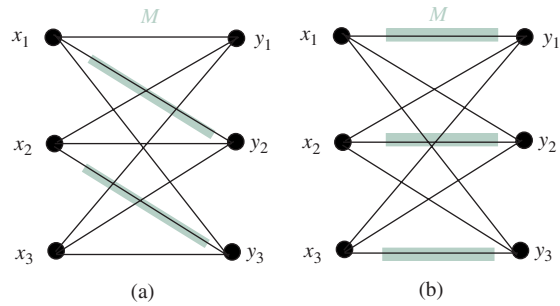


Figure 4.3. *Example of matching in bipartite graphs. (a) Arbitrary matching, M and (b) Complete bipartite graph*

4.3.2.3. *Objective of Hungarian method*

Given the above introductory definitions regarding bipartite graphs, the objective of the Hungarian algorithm is to find the *maximum* weight matching M' in a complete bipartite graph, $G(X, Y, E)$.

Two fundamental notions that allow us to significantly reduce the search space of perfect matchings $\mathcal{M}$ to find the maximum weight matching are the concepts of *vertex labeling* and *equality graphs*.

DEFINITION 4.7.– (Feasible vertex labeling)*: a feasible vertex labeling upon a weighted bipartite graph* $G(X, Y, E)$ *is a function that assigns a label,* $l \in \mathcal{Z}$ *to each vertex in* V*, such that the sum of labels of any pair of vertices connected by an edge* $e_{x,y}$ *is greater than equal to the edge weight* w_{xy}*,*

$$l : V \rightarrow \mathcal{Z} \quad | \quad l(x) + l(y) \geq w_{xy}, \quad \forall x \in X, y \in X$$

DEFINITION 4.8.– (Equality (subgraph):) *given a feasible vertex labeling,* l*, the equality subgraph of a complete bipartite graph is the graph* $G_l(X, Y, E_l)$ *defined by the vertices in* X *and*

Y and a subset of edges $E_l \in E$, whose weights w_{xy} are strictly equal to the sum of vertex labels $l(x) + l(y)$:

$$E_l = \{(x, y)\} \quad | \quad l(x) + l(y) = w_{xy}$$

Given the vertex labeling and equality subgraph definitions, the basis of the Hungarian algorithm is the Huhn – Monkres theorem.

Theorem 4.1. (Huhn – Monkres theorem)*: if a perfect matching is found in an equality subgraph of $G(X, Y, E)$, this matching is the maximum weight matching.*

Proof 1. *From the definition of feasible labeling, any edge $(x, y) \in E$ satisfies that*

$$w(x, y) \leq l(x) + l(y) \qquad [4.6]$$

For a perfect matching M, each vertex is only adjacent to one edge, thus,

$$\sum_{e_{x,y} \in M} w(x, y) \leq \Big[\sum_{(e_{x,y}) \in M} l(x) + l(y) = \sum_{x \in X} l(x) + \sum_{y \in Y} l(y) \Big] \qquad [4.7]$$

Now, any edge $e_{x,y}$ in an equality graph satisfies

$$w(x, y) = l(x) + l(y) \qquad [4.8]$$

Hence, for any perfect matching over the equality graph, M_l, it yields

$$\sum_{(x,y \in M')} w(x, y) = \sum_{x \in X} l(x) + \sum_{y \in Y} l(y) \qquad [4.9]$$

Finally, by merging equations [4.7] and [4.10]

$$\sum_{(x,y \in M')} w(x, y) = \sum_{x \in X} l(x) + \sum_{y \in Y} l(y) \geq \sum_{(x,y \in M), M \in \mathcal{M}} w(x, y) \qquad [4.10]$$

Thus, the problem of finding a maximum weighted matching is transformed into the one finding a feasible vertex labeling with a perfect matching in the associated equality subgraph. This is essentially achieved by selecting an initial vertex labeling as well as a matching M, of size $|M|$, in the equality graph, and iteratively growing M until it becomes a perfect matching ($|M| = k$). In each iteration, the size of M is increased by one edge after an *augmented path* is found.

DEFINITION 4.9.– (Path): *a path over a graph $G(V, E)$ is defined as a sequence of vertices $\{v^1, v^2, v^3, v^4, \ldots, v^p\}$, such that there exists an edge connecting each pair of vertices (v^i, v^{i+1}). Note that the superscripts $1, 2, \ldots, p$ are not referred to the indices of the vertices in the graph, but to their order in the path sequence.*

DEFINITION 4.10.– (Augmented path): *given a matching M in the equality graph, an augmented path is a path (1) whose edges alternate between M and $\bar{M}$ (alternating path), and (2) whose start and end vertices, v^1 and v^p, are unmatched, i.e., $\{(v^1, v^2) \in \bar{M}, (v^2, v^3) \in M, (v^3, v^4) \in \bar{M}, \ldots, (v^i, v^{i+1}) \in \bar{M}\}$.*

Obviously, if an augmented path is found, the size of M can be increased by one edge by inverting the edges in the path from $\bar{M} \rightarrow M$ and $M \rightarrow \bar{M}$ so that the new path can be expressed as (Figure 4.4(a)) $\{(v^1, v^2) \in M, (v^2, v^3) \in \bar{M}, \ldots, (v^i, v^{i+1}) \in M\}$ (Figure 4.4(b)).

As mentioned earlier, the Hungarian algorithm starts by an arbitrary vertex labeling. Typically, the labels for the vertices in Y are set to 0, while each vertex $x_i \in X$ is labeled with the maximum of its incident edges,

$$l(y_i) = 0 \qquad [4.11]$$

$$l(x_i) = \max_{y_i \in Y}(w(x_i, y_i)) \qquad [4.12]$$

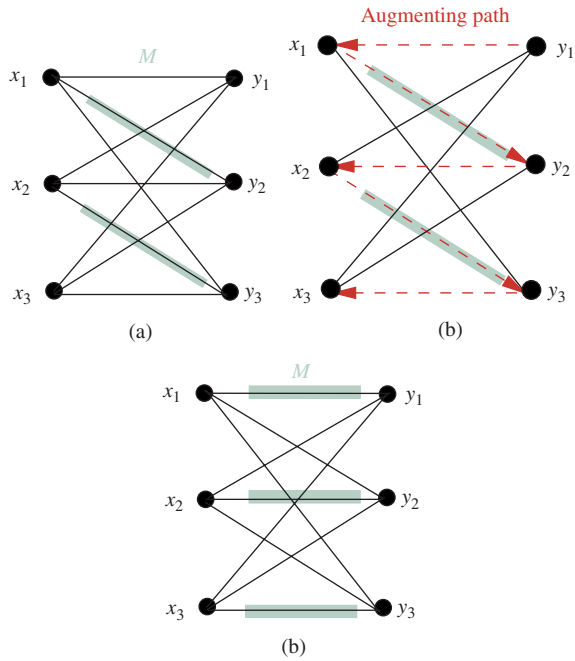


Figure 4.4. *Illustration of augmented paths. (a) Bipartite graph with two disjoint subsets and an arbitrary matching M; (b) Alternating path with start at y_1 and end at $x3$. The vertex order is indicated by the arrows. (c) The alternating path allows us to increase the matching M, resulting in a perfect matching*

Then, a matching in the equality graph, E_l, associated with the vertex labeling is selected. If $|M| = k$, this matching is already perfect and the optimum is found. If the matching is not perfect, the size of M needs to be gradually incremented in a number of iterations. By definition 4.10, $|M|$ can be increased by one edge if an augmenting path is found. Thus, each iteration step is directed toward the search for an augmented path.

As M is not yet perfect, there must be some unmatched vertex $x \in \mathcal{X}$ connected to a matched vertex y. This seems

to be obvious, since otherwise the vertex x_i would be already matched in M. Let $N_l(x)$ denote the subset of vertices in $\mathcal{Y}$ which are connected to x (the neighbors of x). Also, let X' denote the subset of vertices $X' \in \mathcal{X} - \{x\}$ matched to any vertex in $N_l(x)$. Thus, x is "competing" with X' for an edge in M. The path $\{x, y, x' \in X'\}$ can be thought of as a section of an augmented path. Now, if an unmatched vertex y' is found, connected to any vertex in $S = \{X' \cup x_i\}$ in a (new) equality graph, two situations may occur:

1. y' is connected to x. Then, M can be increased by adding a new edge (x, y'). The new matching can be expressed as $M' = M \cup (x, y')$,

2. y' is connected to a vertex in X'. Let this vertex be denoted as x_2, and y_2, the vertex in $\mathcal{Y}$ matched to x_2. Then, an augmented path can be found in the form x, y_2, x_2, y', and M can be incremented by inverting the path edges from M to $\bar{M}$ and vice versa.

Now, assuming that a maximum size matching M has been previously selected, the required unmatched vertex y' does not yet exist in E_l. Otherwise, this vertex would already be included in M. Therefore, the equality graph E_l must be expanded to find new potential vertices in Y to augment M. Obviously, the expansion of E_l requires a vertex (re)labeling l'. It is formulated as

$$\delta_l = \min_{x \in S, y \in N_l(S)} (l(x) + l(y) - w(x, y)) \qquad [4.13]$$

$$l'(v) = \begin{cases} l(v) - \delta_l, & v \in S \\ l(v) + \delta_l, & v \in N_l(S) \\ l(v), & \text{otherwise} \end{cases} \qquad [4.14]$$

The relabeling function l' ensures that a new equality graph is found, E'_l, such that $E'_l \in E_l$ and some new edge $(x_i \in S, y \notin N_l(S))$ exists. In other words, the new set of S neighbors in E'_l is $N'_l(S) = N_l(S) \cup \{y\}$.

Consider the new edges ($x \in S, y \notin N_l(S)$) in E'_l. With reference to the vertex y, two possible cases are to be considered:

– $\nexists x_i : (x_i, y) \in M$ (y is not matched). Thus, an augmented path can be found and $|M'| = |M| + 1$.

– $\exists x_i : (x_i, y) \in M$ (y is matched). The new edge can be expressed as $(x_i \in S, y \notin N_l(S) \in M)$. Since y is not an unmatched vertex, the path cannot be augmented. In this case, the vertex x' matched to y is attached to S so that y belongs to $N'_l(S)$. Then, a new vertex relabeling is required, forcing new edges in E'_l connecting vertices from S to $Y - N'_l(S)$. Such vertex labeling is iterated, adding vertices to S and $N_l(S)$ until an unmatched vertex in Y is found.

Each time $|M|$ is incremented, the previous steps are repeated, starting with another free vertex in X. This process is iterated until all free vertices in X are explored, in which case $|M| = k$ and a perfect matching (the maximum weight matching) is achieved.

4.3.2.4. *Complexity considerations*

In each phase of the Hungarian method, the size of the matching $|M|$ is incremented by one edge. Thus, the perfect matching is attained in a maximum of k phases. As explained earlier, within each phase, a vertex relabeling is required to find a free vertex $y \notin N_l(S)$. If the size of the matching at phase i is $|M^i|$, a maximum number of relabeling steps $|M^i| - 1$ is required in the worst case. Thus, each phase requires at most $|M^i| - 1$ iterations. Typically, the upper bound for such complexity is stated as $O(k)$. Finally, each relabeling requires, in principle, a total number of k^2 operations. However, with an adequate implementation this complexity can be reduced to $O(k)$. Thus, the total complexity of the Hungarian method is $O(k)$ phases $\cdot O(k)$ relabelings $\cdot O(k) = O(k^3)$.

4.3.3. *Genetic algorithms*

In the early 1970s, John Holland invented an algorithmic paradigm which attempted to solve engineering problems by closely observing nature. Such algorithms showed an extraordinary potential to solve a variety of complex problems, outperforming traditional approaches.

Inspired by Darwin's discoveries – the laws of natural selection, inheritance, genetics and evolution – this kind of algorithm was named GA [DAV 91, GOL 89].

From evolution theory, natural selection is governed by the "survival of the fittest" law. According to this principle, those living structures that perform well under certain environmental conditions tend to survive, unlike those that fail to adapt to the environment.

This evolutionary principle forms the basis for GAs. The environment is modeled by the optimization criterion (objective function), while the individuals are represented by their chromosomes. *Natural selection* ensures that only the genetic material of the best individuals is preserved in the next generations. In terms of GAs, the former is achieved by an appropriate *parent selection* strategy. Also, the evolution from one generation to the next one is present in the form of reproductory operators. In each new generation, the chromosomes are generally better "adapted" to the environment (quality criterion), which implies that the GA is evolving toward an optimum of the objective function. By representing the solution space in a chromosomal form, each particular value of a chromosome encodes a possible solution to the objective function (environment). The optimum solution is thus given by the value of the fittest chromosome in a convergence status.

In general terms, a GA can be described by the following steps:

1. Initialize a random population of chromosomes.
2. Evaluate chromosomes on the basis of their fitness values.
3. Using a suitable parent selection strategy, pick a number of chromosomes into the "mating pool" of parent. chromosomes
4. Perform reproduction operators (crossover and mutation) on the mating pool members to form child chromosomes. The next generation is formed using a suitable population formation strategy.
5. If the termination criterion is fulfilled, the fittest chromosome in the current population can be regarded as the "winning" chromosome – optimum solution. Otherwise, go to step 2.

In the following sections, an brief overview of reproduction operators is presented.

4.3.3.1. *Reproduction operators*

Two types of reproduction can be functionally distinguished. Reproduction operators used in GA are inspired by the biological concepts of sexual/asexual reproduction. By means of sexual operators (crossover), the genetic material of two parent individuals is combined to form children chromosomes. In contrast, asexual operators (mutation) produce random changes in single genes, gene positions, or gene substrings in a chromosome. Crossover and mutation operators are explained in detail in the following paragraphs.

4.3.3.1.1. Crossover

As explained earlier, crossover operators require the participation of two parent chromosomes to form the children chromosomes. Different types of crossover operators may

be applied within a GA. However, one essential factor to take into consideration for choosing crossover types is the "context sensitivity." As certain crossover types may suit a certain chromosome representation, they may produce invalid chromosomes for other representation. If the crossover type produces invalid chromosomes, the GA must be aware of these situations and reject any possible invalid solution.

– *One point crossover*: in this type of crossover, a point within the chromosome's length is randomly selected. The genetic material of the two parents beyond this point is swapped to form the children.

– k- *point crossover*: this operator is a generalization of the one-point crossover. k points along the chromosome length are selected, swapping the genetic material of the parents between each pair of consecutive points.

– *Uniform-based crossover*: while the previous operators are only suitable for binary chromosome representation, the uniform-order-based crossover was proposed by Davis [DAV 91] for the permutation representation of chromosomes. First, a binary mask of the same length as that of a chromosome is randomly selected. This mask determines which child inherits each gene and from which parent, in particular. As an example, if bit i is active ('1'), child 1 inherits gene at position i from parent 1. If the bit is inactive ('0'), child 2 inherits the gene from parent 2. The missing gene values – corresponding to the positions of the mask inactive bits in child 1 and active bits for child 2 – are filled in the remaining gene positions of children 1 and 2, preserving their order of appearance in parents 2 and 1, respectively.

– Partially mapped crossover: also devised for the chromosome permutation representation, this operator is similar to the uniform-order-based crossover. The gene positions to be copied from parents 1 and 2 in children 1 and 2 are specified by a random binary mask, as explained earlier for the uniform-order-based crossover. Then, the child

chromosomes 1 and 2 are completed with the missing gene values, preserving the order *and position* as they appear in parents 2 and 1, respectively, to the highest possible extent.

4.3.3.1.2. Mutation

A variety of mutation types have been proposed in the GA literature for asexual reproduction. In the following, five popular mutation operators are described.

– *Simple bit mutation*: This operator, defined for binary chromosome representation is the simplest mutation operators. An intrinsic parameter of the bit mutation strategy is the bit mutation rate, $m \in [0, 1]$. It indicates a maximum threshold for mutant genes. Mutant genes is identified as follows: First, each gene in a chromosome is assigned a random value, $p \in [0, 1]$ from a uniform probability distribution. Mutant genes are those with $p \leq m$. If a gene is identified as mutant, it means that its value can be reversed $(0 \rightarrow 1, 1 \rightarrow 0)$ with probability 0.5.

– *Flip bit mutation*: this operator is very similar to the simple bit mutation, but differs in that the value of a mutant gene is always reversed (with probability 1.0). Thus, the effective mutation rate is doubled with respect to the simple bit mutation.

– *Swapping mutation*: this operator performs a simple mutation, which is applicable to many types of the chromosome representation. It simply selects two random positions along a chromosome, and exchanges the genes at these positions.

– *Sliding mutation*: as with swapping mutation, this operator can be used with many different types of chromosome representation. Again, two points within a chromosome's length are selected, i, j, with $i < j - 1$. The segment of genes at indexes $i + 1$ to j slided by one position to the left, pushing the gene at position i to position j. In vectorial form, the sliding

mutation operator can be expressed as follows:

$$\text{chromosome}[i..j] = \text{chromosome}[i+1..j], \text{chromosome}[i] \quad [4.15]$$

– *Scramble bit mutation*: this operator was proposed by Davis [DAV 91] for the permutation representation of chromosomes. It selects a substring of genes and combines them randomly, leaving the rest of the genes unchanged.

4.3.3.2. *Forming the next generation*

An overview is now provided of the strategies to form the next generation from the current population of chromosomes. In particular, three approaches are described: *generational replacement*, *elitism and generational replacement*, and *steady-state representation*.

4.3.3.2.1. Generational replacement

Using this strategy, the parent generation is entirely replaced by a new population of children chromosomes obtained by crossover and mutation. One limitation of generational replacement is associated with these reproductory operators. As previously explained, according to the *survival of the fittest* principle, fittest individuals are selected into the mating pool of parent chromosomes. Crossover and mutation operators may, nevertheless, destroy good schemata present in parent chromosomes, by combining or mutating the genes within a schema.

4.3.3.2.2. Elitism with generational replacement

This method attempts to overcome the aforementioned limitation of simple generational replacement through the introduction of elitism or "elite selection." In the frame scope of GAs, the term "elite" refers to the top fittest chromosomes in a population. Denoting P, the population size and e, the elite selection rate, the top $|e \cdot P|$ fittest chromosomes are identified

as elite chromosomes or super-individuals. An exact copy of each elite chromosome is then passed to the next generation. The remaining $P - eP$ children are then produced by simple generational replacement. According to Davis, a small value of the elite selection rate e may be beneficial for the GA performance, since it ensures that fit chromosomes with good schemata are preserved in the next generations. In contrast, large values of this parameter can lead to degradations in the GA performance, or slow down the convergence, due to the subsequent reduction in the "active" search space (in terms of population chromosomes).

4.3.3.2.3. Steady state representation

This method replaces only the m chromosomes with the poorest fitness values in a population, keeping exact copies of the $P - m$ remaining chromosomes. Obviously, the steady-state reproduction is equivalent to elitism for $m = P$ and corresponds to elitism with generational replacement with an elite parameter $e = P - m$.

4.3.3.3. *GAs applied to optimum cluster labeling*

In this book, GAs have been applied to solve the cluster labeling problem. As explained in Section 4.3, the cluster labeling objective is equivalent to find an optimum permutation of class labels, by considering a fixed ordering of the clusters $(C_1 to C_k)$. For this reason, a permutation representation style has been used as chromosome encoding:

$$\Pi = \{\pi_1, \pi_2, \cdots, \pi_k\}$$

In the chromosome, cluster labels are encoded by the gene positions, whereas class labels are indicated by the gene values, i.e. $\pi_k = L(C_k)$. For example, if the gene at position i has a value $\pi_i = j$, the ith cluster is assigned to class label $L(C_i) = j$.

The evaluation of a chromosome is defined as the total cost definition (equation [4.4]) of wrong cluster–class label assignments associated with the class label permutation. In terms of the chromosome gene values, the labeling cost can be expressed as

$$\text{Chromosome evaluation} = \text{total cost } (\Pi) = \sum_{i=1}^{k} \left(\sum_{j \neq i} n_{\pi_i, j} \right) \quad [4.16]$$

Thus, the fitness function that allows us to identify the top chromosomes in a population can be defined as the inverse of equation [4.16]:

$$\text{Fitness } (\Pi) = n - \text{total cost } (\Pi) = \sum_{i=1}^{k} n_{\pi_i, i} \quad [4.17]$$

Note that the inverse of the labeling cost, or fitness, corresponds in this case to the total overlap of class labels due to the following conditions:

– The weight factors in the labeling cost formulation in equation [4.2] are not taken into account (all weights are set to 1).

– The number of labeled examples in the labeled seed is considered constant for all categories.

Owing to the first condition, the cost formula can be expressed as

$$\text{Total cost } (\Pi) = \sum_{i=1}^{k} \left(\sum_{j \neq i} n_{\pi_i, j} \right) = \sum_{i=1}^{k} n_i - n_{\pi_i, i} \quad [4.18]$$

where n_i denotes the number of labeled examples in the initial seed for the class label indicated by the gene π_i. As this value

is constant for all class labels, i.e., $n_1 = n_2 = \ldots, = n_k = n$, the cost in equation [4.18] can be also formulated as

$$\text{Total cost } (\Pi) = \sum_{i=1}^{k} n_i - n_{\pi_i,i} = n - \sum_{i=1}^{k} n_{\pi_i,i} \qquad [4.19]$$

Hence, the chromosome fitness can be identified as the sum of overlapping products in the second term of equation [4.19].

As for parent selection, a simple tournament selection strategy has been selected. This approach divides the population in equally sized blocks, each with M chromosomes. Then, it selects the N fittest chromosomes from each block into the mating pool. In this book, different values of M have been applied, namely, $M = 5$, $M = 10$, and $M = 11$. In all cases, two fittest parents are selected from each block ($N = 2$).

Generational replacement with elitism is the method selected to form the next generation, using different types of crossover and mutation to obtain the genetic material of the children from their parent's chromosomes. Now, the new generation is formed in blocks of M chromosomes. For the first block, the first two parents in the mating pool are selected. The block is filled with M children, obtained from crossovers and mutations from their corresponding two parents. The same is applied for the rest of blocks, using the second, third, etc., pairs of parents in the mating pool. Let c_r and (m_r) denote the crossover and effective mutation rates applied to the GAs, respectively. Different crossover and effective mutation rates have been compared, depending on the block size M. For $M = 5$, two children are generated with crossover ($\mathbf{cr} = 2/5$) and three children with mutations ($\mathbf{mr} = 3/5$). For $M = 10$, the crossover and mutation rates have been set to $\mathbf{cr} = 6/10$ and $\mathbf{mr} = 4/10$. Likewise, for $M = 11$, $\mathbf{cr} = 10/11$ and $\mathbf{mr} = 1/11$.

4.3.3.4. *Comparison of methods*

In the following, the performances of the Hungarian algorithm and GAs for the optimum assignment problem are compared by means of a controlled experiment. As both strategies have been devised for the cluster labeling problem, artificial confusion matrices have been synthesized. They are possible real situations with clusters/class labels. A confusion matrix represents the number of class labels (rows) that fall into the different clusters (columns), with the particularity that the elements in the diagonal correspond to the best cluster–class matchings. The matrix overlapping products O in equation [4.1] can be transformed into a confusion matrix, O', by re-arranging the row elements according to the optimum class label permutation. In this case, the sum of values in the diagonal of O' gives the maximum overlap (minimum of equation [4.2]).

Ideally, the cluster partition of a dataset should perfectly match the underlying class distribution. The resulting confusion matrix is thus a diagonal matrix. However, in most real problems, clusters may deviate from the true classes to a certain extent. This may be due to the inadequate choice of a cluster algorithm that does not fit into the true data distribution, the existence of ambiguities in the dataset, or even the presence of errors in the manual labels. To analyze the cluster labeling performance of the Hungarian algorithm and GAs under these circumstances, the artificial matrices used in this work are intended to reproduce and quantify their potential effects on the confusion matrix O'.

First, it is assumed that each class can be optimally represented by one of the clusters. It is, namely, the cluster that provides the best coverage of the elements in the class. In a non-ideal situation, a certain amount of labeled patterns may fall outside their "representative" cluster, as a sort of interference to the receiver clusters. Two variables have been used to model these situations:

– Variable ratio of "escaping" labels (E): given a number of labels per category, n, and summing each class to be optimally represented by a certain cluster $C^{(\text{opt})}$, this variable indicates the ratio of class labels that "escape" outside cluster $C^{(\text{opt})}$ to other neighbor clusters (also referred to as receiving clusters). For example, if $L = 20$ and $E = 0.5$, 10 labels of the given class escape cluster $C^{(\text{opt})}$. This variable *ratio of escaping patterns* is a uniformly distributed variable in the range $E \in [0 - E_{\max}]$, where $E_{\max} < 1$ is a parameter of choice. In this book, different values of $E_{\max}$ have been applied to "control" the complexity of the cluster labeling problem.

– Variable number of receiving clusters Rx: this variable indicates a number of (undesired) clusters that are recipient of the escaping patterns. Again, this is a uniform random variable in the range $Rx \in [0, Rx_{\max}]$, where $Rx_{\max} < k$ is a parameter specified by the user. Note that the escaping patterns are not homogeneously distributed through the recipient clusters. Instead, it is assumed that some clusters are somewhat closer to the desired cluster $C^{(\text{opt})}$ than others and should therefore receive a larger number of labels than more distant clusters. Hence, a number of labels that fall into the clusters is generated according to a triangular filter centered in the diagonal element, whose weights add to 1. Rx values are obtained, which are randomly rearranged to occupy the row positions adjacent to the diagonal elements.

Four of the artificial confusion matrices used in the experiments are shown in Figure 4.5, which represent potential clusterings with $k = 20$ clusters/classes. These matrices are referred to four different combinations of values for $(E_{\max}, Rx_{\max})$: (0.2,12), (0.2,5), (0.9,12), and (0.9,5) (Figure 4.5(a–d), respectively).

Figure 4.6 shows the labeling performance of the Hungarian algorithm and the three implementations of GAs, with effective mutation rates of mr $= 3/5, 4/10$, and $1/11$.

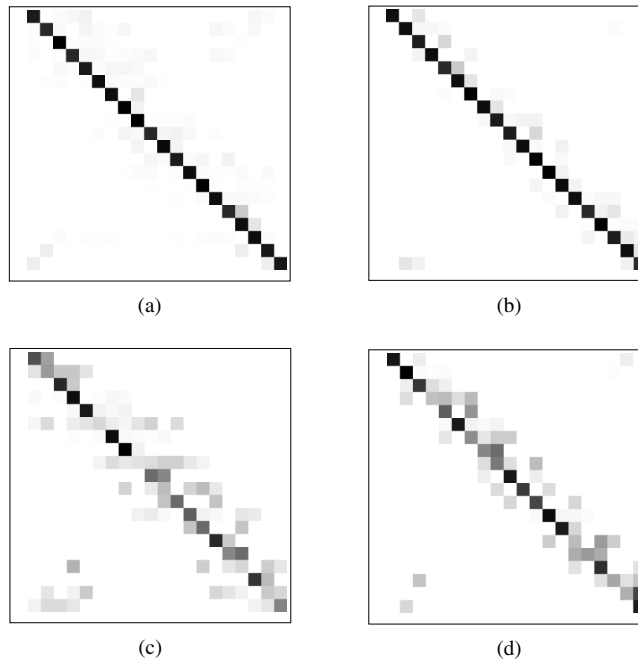


Figure 4.5. *Artificial confusion matrices used to represent the cluster labeling problem, with different values of the parameters, E and Rx. (a) $E = 0.2, Rx = 15$; (b) $E = 0.2, Rx = 5$; (c) $E = 0.9, Rx = 15$; (d) $E = 0.9, Rx = 5$*

In the plots, these three implementations of the GA are referred to as GA1, GA2, and GA3, respectively. The labeling performance is depicted in terms of GA generations required to achieve the optimum labeling (minimum labeling error). Thus, the performance of the Hungarian algorithm is plotted as a constant value.

As can be observed, the most effective GA implementation is the one with the highest effective mutation rate (GA3). It achieves the optimum in around 40 generations, with minimum variations regardless of the different confusion matrices. As the mutation rate decreases, the GA convergence

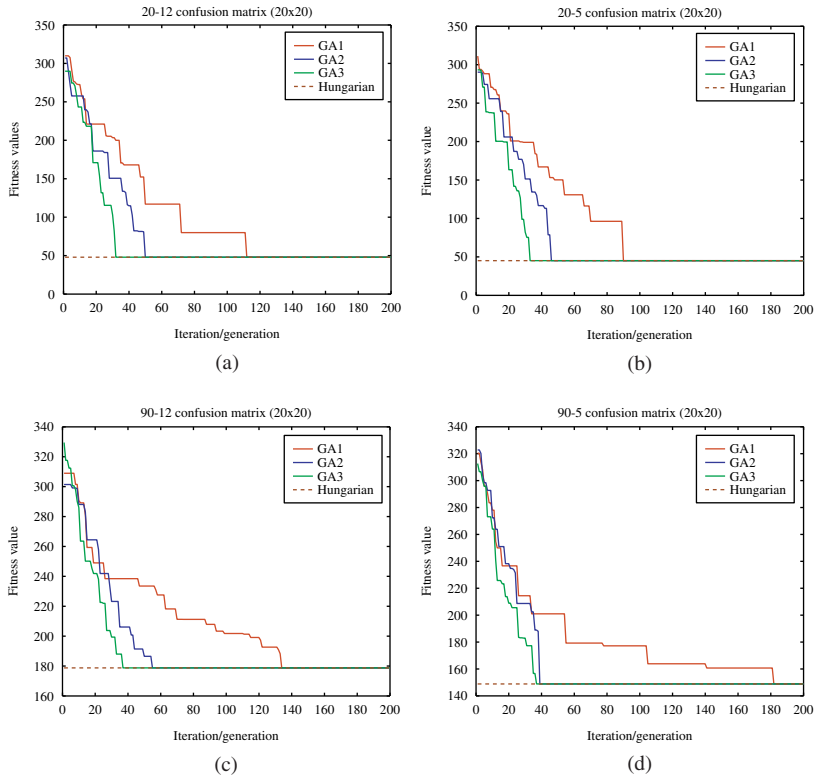


Figure 4.6. *Cluster labeling error produced by the GA and Hungarian algorithm on the confusion matrices in Figure 4.5. (a)* $E = 0.2, Rx = 15$*; (b)* $E = 0.2, Rx = 5$*; (c)* $E = 0.9, Rx = 15$*; (d)* $E = 0.9, Rx = 5$

slows down significantly. As an example, The GA1 implementation (mr $= 3/5$) requires a double number of generations for convergence with respect to the other two implementations for $E_{\max} = 0.2$ and almost a triple number of iterations for $E_{\max} = 0.9$.

Given the previous explanations, the GA3 implementation of the GA has been the one selected in this work to solve the cluster labeling problem in real clustering tasks.

4.4. Supervised classification block

The final stage of the semi-supervised approach developed in this book is the *supervised classification* block. The basic difference of the semi-supervised strategy versus a fully supervised approach relies on the labeled set which is fed to this block. In the semi-supervised approach, the supervised classification is fed with the augmented labeled seed from the cluster labeling step. The supervised counterpart is attained when the initial labeled seeds are directly used to train this classification block instead. According to the block diagram in Figure 4.1, the *supervised classification* block is defined in generic terms. In practice, any supervised classifier available in the ML literature can be applied by just adapting the necessary software interface (a function call in R). In this work, two popular classification models have been applied to the supervised classification block: *support vector machines* (SVMs) and, for a dataset of utterances (see section 4.5.3), the *naive Bayes* rule. These algorithms are described in more depth in the following paragraphs.

4.4.1. *Support vector machines*

SVMs are among the most popular classification and regression algorithms because of their robustness and good performance in comparison with other classifiers [BUR 98, JOA 97, LIN 06]. In its basic form, SVM were defined for binary classification of linearly separable data. Let $\mathcal{X}$ denote a set of training patterns $\mathcal{X} = \{x_1, x_2, \cdots x_L\}$ in $\mathcal{R}^D$. For binary classes, the possible class labels corresponding to the training data elements ($\mathcal{Y} = \{y_1, \cdots, y_L\}$) are $y_i \in \{1, -1\}$ (Figure 4.7).

Assuming that the classes $(+1, -1)$ are linearly separable, the SVM goal is to orientate a hyperplane H that maximizes the margin between the closest members of the two classes

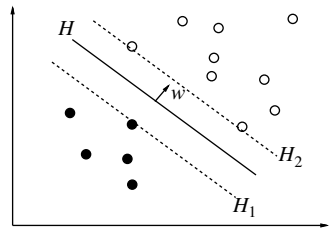


Figure 4.7. *SVM example of binary classification through hyperplane separation (in this two-dimensional case the hyperplane becomes a line)*

(also called support vectors). The searched hyperplane is given by equation [4.20],

$$H := \{x \in \mathcal{R}^D \mid wx + b = 0\} \qquad [4.20]$$

denoting w the normal vector of the hyperplane. In addition, the parallel hyperplanes H_1 and H_2 at the support vectors of classes $y = 1$ and $y = -1$ are defined as

$$\begin{aligned} H_1 &:= \{x \in \mathcal{R}^D \mid wx + b = 1\},\ (y = 1) \\ H_2 &:= \{x \in \mathcal{R}^D \mid wx + b = -1\},\ (y = -1) \end{aligned} \qquad [4.21]$$

It can be demonstrated that the margin between the hyperplanes H_1 and H_2 is $\frac{1}{||w||}$. In addition, the training points at the left/right sides of the hyperplanes H_1 and H_2 need to satisfy

$$y_i(wx_i + b) - 1 \geq 0 \quad \forall i \qquad [4.22]$$

Thus, the maximum margin hyperplane is obtained by solving the following objective:

$$\text{minimie } \parallel w \parallel \text{ such that } y_i(wx + b) - 1 \geq 0 \qquad [4.23]$$

By applying Lagrange multipliers, the objective in equation [4.23] can be solved using constrained quadratic optimization.

After some manipulations, it can be shown that the initial objective is equivalent to

$$\begin{aligned} &\text{maximize} \quad \sum_{i=1}^{L}\left(\alpha_i - \frac{1}{2}\sum_j \alpha_i\alpha_j y_i y_j x_i x_j\right) \\ &\text{subject to} \quad \alpha_i \geq 0\ \forall i, \text{ and } \sum_{i=1}^{L}\alpha_i y_i = 0 \end{aligned} \qquad [4.24]$$

The solution of this quadratic optimization problem is a set of coefficients $\alpha = \{\alpha_1, \cdots, \alpha_L\}$, which are finally applied to calculate the hyperplane variables w and b:

$$\begin{aligned} w &= \sum \alpha_i y_i x_i \\ b &= \frac{1}{N_s}\sum_{s\in\mathcal{S}} y_s - \sum_{m\in\mathcal{S}} \alpha_m y_m x_m x_s \end{aligned} \qquad [4.25]$$

where $\mathcal{S}$ denotes the set of support vectors of size N_s.

Although the solution in equation [4.25] is found for the basic problem of binary, linearly separable classes, SVMs have been extended for both multi-class and nonlinear problems.

4.4.1.1. *The kernel trick for nonlinearly separable classes*

The application of SVMs to nonlinearly separable classes is achieved by substituting the dot product $x_i x_j$ in equation [4.23] by an appropriate function, the so-called kernel $k(x_i, x_j)$. The purpose of this "kernel trick" is that a nonlinear kernel can be used to transform the feature space into a new space of higher dimension. In this high-dimensional space it is possible to find a hyperplane to separate classes that may not

be originally separable in the initial space. In other words, the kernel function is equivalent to the dot product:

$$k(x_i, x_j) =< \phi(x_i)\phi(x_j) > \quad [4.26]$$

where ϕ denotes a mapping of a pattern into the higher dimensional space. The main advantage is that the kernel computes these dot products without the need to specify the mapping function ϕ.

4.4.1.2. *Multi-class classification*

The extension for the multi-class problem is achieved through a combination of multiple SVM classifiers. Two different schemes have been proposed to solve this problem: in a *one-against-all* approach, k hyperplanes are obtained to separate each class from the rest of classes. In a *one-against-one* approach, $\binom{k}{2}$ binary classifiers are trained to find all possible hyperplanes to separate each pair of classes.

4.4.2. *Example*

An example of the models learned by the supervised and semi-supervised approaches in a mixture of two Gaussians is provided in Figure 4.8. In this example, two random examples per category (labeled as "A" and "B") comprise the initial labeled seed (Figure 4.8(a)). In a supervised approach, this seed is used to train SVM model with a radial kernel. The learned hyperplane or the decision curve is depicted in Figure 4.8(b). The areas at both sides of the hyperplane are the decision regions for Gaussians A (in gray color) and B (green color). Figure 4.8(c) indicates the cluster partition of the complete data, and the corresponding cluster labeling performed by the semi-supervised approach. The so-called labeled Gaussians are used to train an SVM model. As can

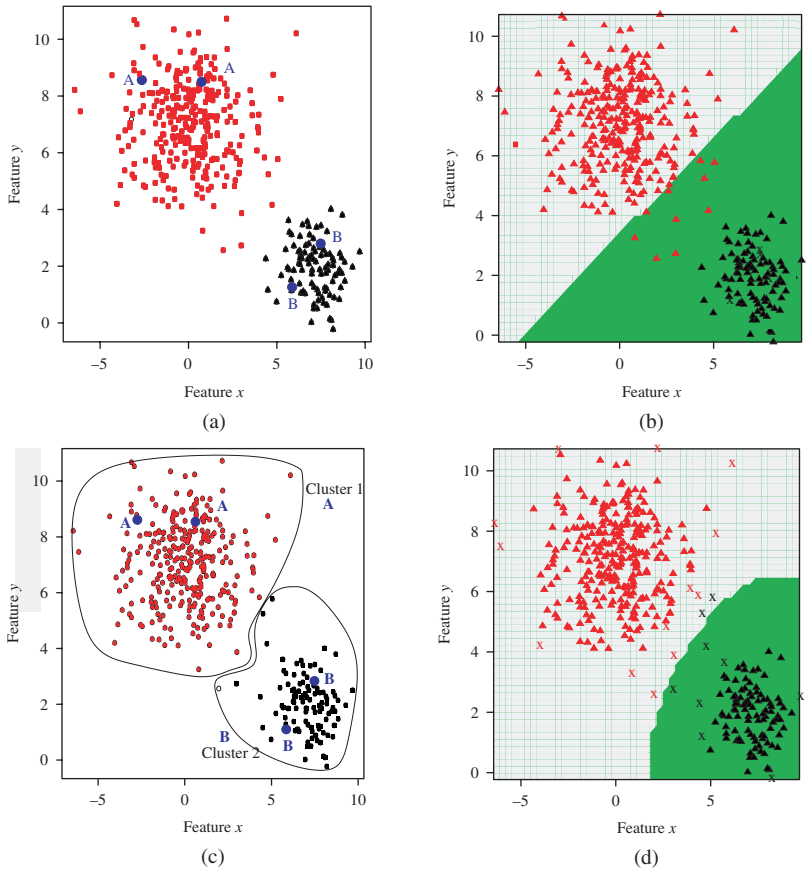


Figure 4.8. *Example of SVM model learned on a mixture of two Gaussians by the supervised and semi-supervised algorithms. (a) Dataset with an initial seed, whose objects are indicated by the labels "A" and "B". (b) Model learned by SVM on the initial labeled seed. (c) Clusters and augmented labels obtained by the semi-supervised algorithm. (d) Model learned by the SVM on the augmented labeled set*

be observed in Figure 4.8(d), following the semi-supervised approach a more accurate hyperplane is learned, which achieves a lower number of errors on the training set in comparison to the supervised model.

4.5. Datasets

This section describes the datasets used in the experiments. As the algorithms presented in the previous sections have a general character and can be applied to any kind of features, they have been tested on several datasets from the UCI ML repository and on a corpus of utterances.

4.5.1. *Mixtures of Gaussians*

This dataset comprises two mixtures of five and seven Gaussians in two dimensions, where a certain amount of overlapping patterns can be observed.

4.5.2. *Datasets from the UCI repository*

4.5.2.1. *Iris dataset (Iris)*

The Iris set is one of the most popular datasets from the UCI repository. It comprises 150 instances iris of 3 different classes of Iris flowers (Iris setosa, I. versicolor, and I. virginica). Two of these classes are linearly separable while the third one is not linearly separable from the second one (Figure 4.9).

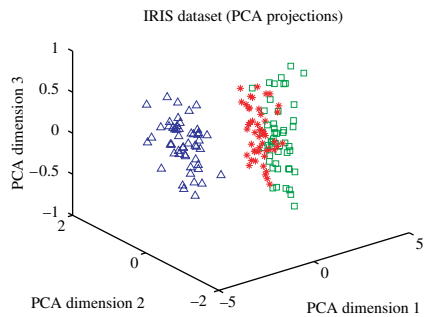


Figure 4.9. *Iris dataset (projection on the three principal components)*

4.5.2.2. *Wine dataset (wine)*

The wine set consists of 178 instances with 13 attributes, representing 3 different types of wines.

4.5.2.3. *Wisconsin breast cancer dataset (breast)*

This dataset contains 569 instances in 10 dimensions, denoting 10 different features extracted from digitized images of breast masses. The two existing classes are referred to the possible breast cancer diagnosis (malignant and benign).

4.5.2.4. *Handwritten digits dataset (Pendig)*

The third real dataset is for pen-based recognition of handwritten digits. In this work, the test partition has been used, composed of 3498 samples with 16 attributes. Ten classes can be distinguished for the digits 0–9.

4.5.2.5. *Pima Indians diabetes (diabetes)*

This dataset comprises 768 instances with 8 numeric attributes. Two classes denote the possible diagnostics (the patients show or do not show signs of diabetes).

4.5.3. *Utterance dataset*

The utterance dataset is a collection of transcribed utterances collected from user calls to commercial troubleshooting agents. The application domain of this corpus is *video* troubleshooting. Reference topic categories or symptoms are also available.

This utterance corpus has been pre-processed using morphological analysis and stop-word removal. First, a morphological analyzer [MIN 01] has been applied to reduce the surface form of utterance words into their word lemmas. Then, the lemmatized words have been filtered using the SMART stop-word list with small modifications. In particular,

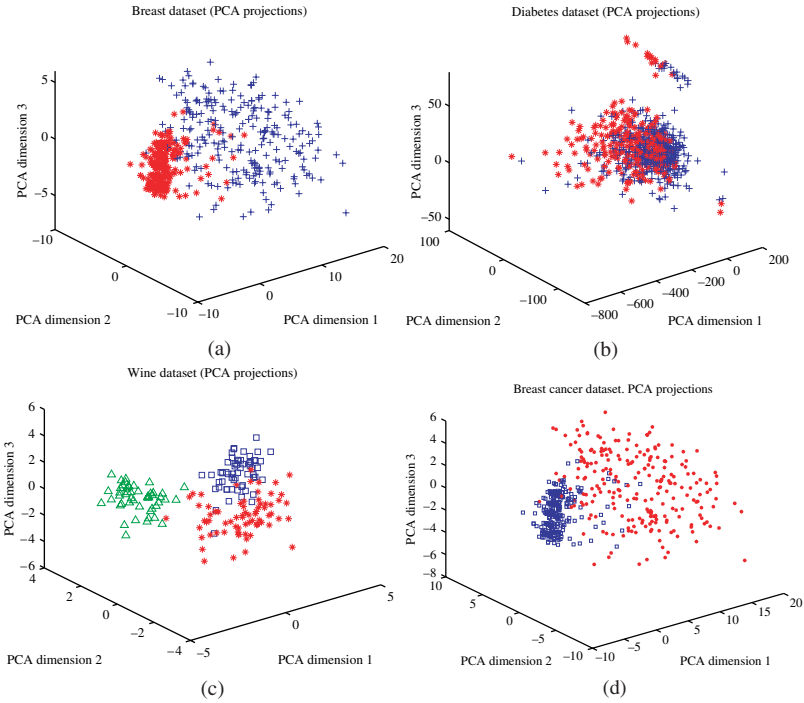


Figure 4.10. *Projection of the datasets on the three principal components. (a) Breast dataset; (b) diabetes dataset; (c) wine; (d) Pendig*

confirmation words (yes, no) have been deleted from the stop word list, while some terms typical for spontaneous speech (eh, ehm, uh, . . .) have been included as stop words. Finally, we have retained the lemmas with two or more occurrences in the pre-processed corpus, resulting in a vocabulary dimension of 554 word lemmas. Afterwards, the pre-processed utterances have been represented in vectors of index terms using Boolean indexing – each binary component in an utterance vector indicates the presence or absence of the corresponding index term in the utterance. The final dataset is composed of 2,940 unique utterance vectors.

An independent test set comprising a total number of 10,000 transcribed utterances has also been pre-processed as described earlier, resulting in 2,940 unique utterance vectors. From this set, a number of utterances ($\sim 20\%$ of the training set size) have been randomly selected as the test applied to the classifiers. To avoid possible biases of a single test set, 20 different test partitions have been generated. From the the training set, 20 different random seeds of labeled prototypes (n labels/category) have also been randomly selected.

4.6. An analysis of the bounds for the cluster and label approaches

As mentioned in section 4.1, the main assumption for the current cluster and label approaches is that the input data should "naturally fall into clusters." If the data are not intrinsically organized in clusters — or the chosen clustering algorithm does not fit into the input data distribution — a cluster and label approach may yield a considerable degradation in the classification performance.

For this reason, the controlled experiment described in this section was intended to analyze the performance of the cluster and label approaches with respect to the quality of the cluster solution.

In these experiments, the Iris dataset from the UCI ML repository has been used. First, a perfect clustering has been considered equal to the set of real labels. Then, different levels of noise have been manually induced to this perfect clustering. This has been done by randomly selecting a ratio p of data patterns from each category and modifying their cluster membership from their actual cluster label $y_i = j$ to a different cluster label $\mathcal{Y} - j$. Thus, p is also referred to as the percentage of misclassification errors. Different values of p have been applied, $p = [0.1, 0.2, \cdots, 0.5]$. For each p value, 20 different

simulated clusterings with a ratio p of induced random noise have been generated. The equivalent normalized mutual information (NMI) between these clusterings and the real labels has also been computed. Also, for each cluster partition, 20 different labeled seeds have been randomly selected. These have been used to train both a supervised classifier and the semi-supervised algorithm with the corresponding simulated clusterings. The study has been carried out for $n = 1$ and $n = 2$ labeled samples per category.

The relationship between the percentage of error and the classification performance is depicted in the scatterplots of Figure 4.11(a) and 4.11(b). Likewise, the relationships in terms of NMI values are shown in Figure 4.11(c) and 4.11(d). In both the cases, horizontal axes denote the cluster qualities (misclassification error or NMI values). Vertical axes indicate the difference in accuracy when using the semi-supervised approach with respect to the supervised SVMs (accuracy semi-supervised – accuracy supervised). Thus, any positive value in the vertical axis indicates an improvement in classification performance through the semi-supervised algorithm. Equivalently, negative values imply a degradation of the semi-supervised approach with respect to the supervised approach.

As can be observed, the accuracy of the semi-supervised classifier with respect to the supervised algorithm increases notably by decreasing the noise ratio induced to the clusterings. If the quality of the cluster partition is not sufficient to capture the underlying class structure — NMI ≤ 0.5 — the supervised approach may outperform the semi-supervised cluster and label algorithm even for $n = 1$. For larger NMI values, the semi-supervised algorithm achieves increasing accuracy gains with respect to the supervised approach, in consistency with the cluster assumption.

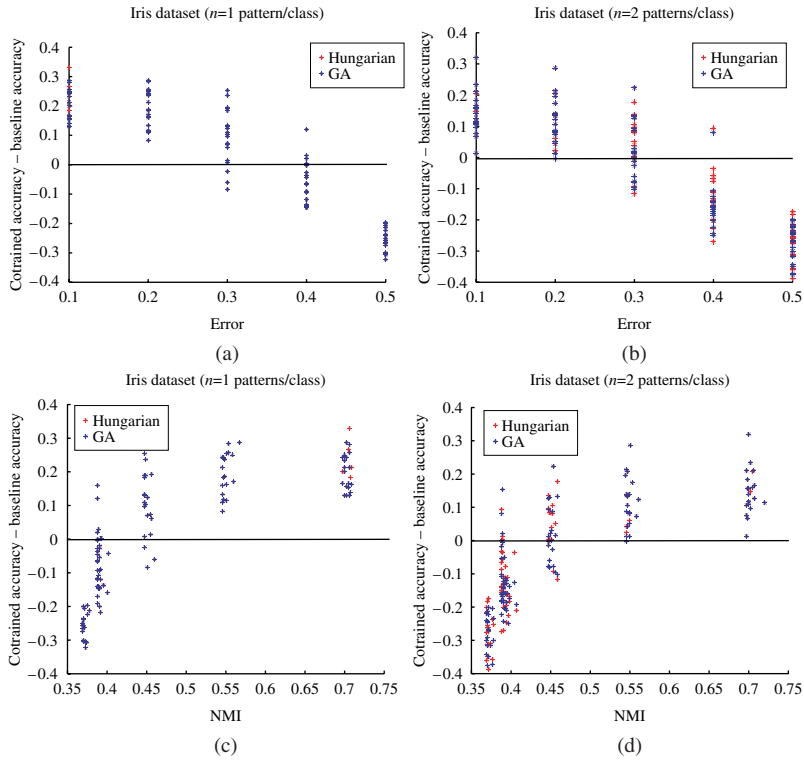


Figure 4.11. *Difference in accuracy scores achieved by the semi-supervised and supervised algorithms with respect to the induced noise p and equivalent NMI parameters. (a) Accuracies of the Iris dataset, $n = 1$ labeled pattern per category, as a function of the induced noise ratio p. (b) Accuracies of the Iris dataset, $n = 2$ labeled pattern per category, as a function of the induced noise ratio p. (c) Accuracies of the Iris dataset, $n = 1$ labeled pattern per category, as a function of the equivalent normalized mutual information NMI. (d) Accuracies of the Iris dataset, $n = 4$ labeled pattern per category, as a function of the equivalent normalized mutual information NMI*

4.7. Extension through cluster pruning

In this book, an optimization to the cluster and label strategy has also been developed. Even though the underlying

class structure may be appropriately captured by a cluster algorithm, the augmented dataset derived by the optimum cluster labeling may contain a number of "misclassification"[2] errors with respect to the real class labels. This happens especially if two or more of the underlying classes show a certain overlapping of patterns. In this case, the errors may be accumulated in the regions close to the cluster boundaries of adjacent clusters.

The general idea behind the proposed optimization method is to improve the (external) cluster quality by identifying and removing such regions with high probability of misclassification errors from the clusters. To this aim, the concept of *pattern silhouettes* has been applied to prune the clusters in $\mathcal{C}$.

The silhouette width of an observation x_i is an internal measure of quality, typically used as the first step for the computation of the average silhouette width of a cluster partition [ROU 87]. It is formulated in equation [4.27],

$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))} \qquad [4.27]$$

where a is the average distance between x_i and the elements in its own cluster, while b is the smallest average distance between x_i and other clusters in the partition. Intuitively, the silhouette of an object $s(x_i)$ can be thought of as the "confidence" to which the pattern x_i has been assigned to the cluster $C(x_i)$ by the clustering algorithm. Higher silhouette scores can be observed for patterns clustered with a higher "confidence," while low values indicate patterns that

2. The term missclasification is not used here to indicate the predicted errors of the end classifiers but the errors after the cluster labeling block. Note that, after cluster labeling, each clustered data pattern has been assigned a class label (the label of its cluster), which can be compared with the real label if the complete labeled set is available.

lie between clusters or are probably allocated in the wrong cluster.

The cluster pruning approach can be described as follows:

– Given a cluster partition $\mathcal{C}$ and the matrix of dissimilarities between the patterns in the dataset, D, calculate the silhouette of each object in the dataset.

– Sort the elements in each cluster according to their silhouette scores, in an increasing order.

– In each cluster, the elements with high silhouettes may be considered as objects with high "clustering confidence." In contrast, such elements with low silhouette values are clustered with lower confidence. This latter kind of objects may thus belong to a class-overlapping region with higher probability. Using the histograms of silhouette scores within the clusters, select a minimum silhouette threshold for each cluster. Further details about the selection of silhouette thresholds by the cluster pruning algorithm are provided in Section 4.7.1.

– Prune each cluster C_i in $\mathcal{C}$ by removing patterns that do not exceed the minimum silhouette threshold for the cluster, chosen in the previous step.

4.7.1. *Determination of silhouette thresholds*

In the proposed cluster pruning method, different silhouette thresholds are applied according to the distribution of silhouette values within each cluster, estimated through histograms. If a significant distortion of the original clusters is introduced through cluster pruning, the learned models

may also deviate from the expected models up to a certain extent. The objective is to remove potential clustering errors while preserving the shape and size of the original clusters to the highest possible extent. In practice, pruning an amount of patterns up to one-third of the cluster size has been considered appropriate for the current purpose. In addition, the selected thresholds also depend on the pattern silhouette values: patterns with a silhouette score larger than $\text{sil} = 0.5$ are deemed to be clustered with a sufficiently high "confidence." Thus, the maximum silhouette threshold applied in the cluster pruning algorithm is $\text{sil}_{\text{th}} = 0.5$. Consequently, if the minimum observed silhouette score in a cluster is larger than $\text{sil}_{th}^{\max} = 0.5$, the cluster remains unaltered in the pruned partitions.

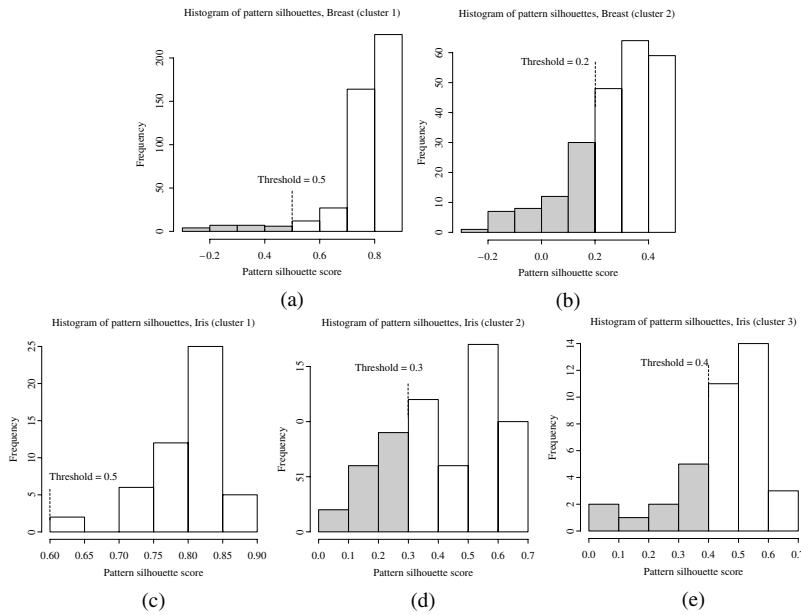


Figure 4.12. *Histograms of silhouette values in the breast (two clusters) and iris (three clusters) datasets, which were used for the determination of the clusters' silhouette thresholds in the cluster pruning approach*

The specific criteria to select the silhouette thresholds can be illustrated by considering the clusters extracted from the iris and breast datasets (Figure 4.12). The distribution of silhouette scores has been estimated by using the histogram function in the R software, which also provides the vectors of silhouette values found as the histogram bin limits and the counts of occurrences in each bin.[3] The silhouette thresholds have been selected to coincide with the histogram bin limits. In the breast dataset (two classes/clusters), the vector of silhouette thresholds for the first and second clusters is [0.5, 0.2]. The value $\text{sil}_{\text{th}} = 0.5$ for the first cluster corresponds to the upper bound $\text{sil}_{\text{th}}^{\max} = 0.5$ as explained in the previous paragraph. This results in the removal of 5.2% of the cluster patterns. For the second cluster, the threshold $\text{sil}_{\text{th}} = 0.2$ is selected. The rejected section associated with sil_{th} corresponds to the first five histogram bins, comprising 25% of the patterns in the cluster. By including the sixth histogram bin in the pruned section, the next possible silhouette threshold level is $\text{sil}_{\text{th}} = 0.3$. However, such threshold level would lead to the removal of a considerable amount (46.28%) of the cluster patterns, which is considered unacceptable for preserving the cluster size/shape.

A similar criterion has been followed for the clusters extracted in the iris dataset. The vector of cluster silhouette thresholds in this case is [0.5, 0.3, 0.4]. The first cluster is left unchanged by the pruning approach, as all observed silhouette scores exceed the upper bound $\text{sil}_{\text{th}}^{\max} = 0.5$. For the second cluster, a silhouette threshold $\text{sil}_{\text{th}} = 0.3$ has been selected, with an equivalent ratio of 27.42% of pruned patterns in the cluster. Note that, by including the next histogram bin ($\text{sil}_{\text{th}} = 0.4$), an inappropriate amount of patterns would be discarded (46% of the cluster size). Finally, the silhouette

3. The bin sizes provided by the R software histogram function are estimated according to the Sturges formula [FRE 81].

threshold for the third cluster is $\text{sil}_{\text{th}} = 0.4$ (26.31% of removed patterns), as the next possible silhouette level (0.5) would result in the removal of 55.26% of cluster patterns.

To summarize, the number of histogram bins corresponding to rejected patterns is determined according to one of these two conditions:

– the upper limit of the last rejected bin should not exceed $\text{sil}_{\text{th}}^{\max} = 0.5$, and

– the amount of rejected patterns (total number of occurrences in the rejected bins) should not exceed one-third of the total number of patterns in the cluster.

Another example of the pruned clusters on the mixtures of five and seven Gaussians is shown in Figure 4.13. The clusters obtained by the PAM algorithm have been tagged as C_1 – C_5 (five Gaussians) and C_1 – C_7 (seven Gaussians). Patterns rejected by the cluster pruning algorithm have been also indicated in red colors. The corresponding histograms of silhouette values are depicted in Figures 4.14 and 4.15. As can be observed, the mixture components in the five Gaussian datasets are well separated and thus easily identified by the clustering algorithm, which also provides well separated clusters. This can also be observed in the silhouette histograms in Figure 4.14. The minimum silhouette value observed in a majority of the clusters exceeds the upper bound for the silhouette threshold, $\text{sil}_{\text{th}}^{\max} = 0.5$. This means that all patterns in the clusters are preserved after cluster pruning. Only clusters C_2 and C_4 have both one pattern with silhouette scores lower than $\text{sil}_{\text{th}}^{\max}$. Such patterns can be observed in red color in Figure 4.13(a). It can be observed that these patterns are outliers in their respective Gaussian components.

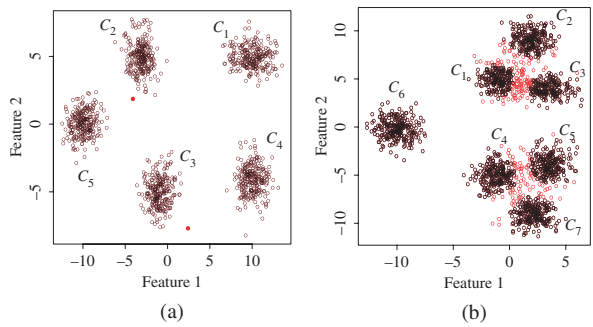


Figure 4.13. *Example of patterns rejected by the cluster pruning approach in two mixtures of five and seven gaussians. The rejected patterns are indicated in red colors. (a) Five Gaussians; (b) seven Gaussians*

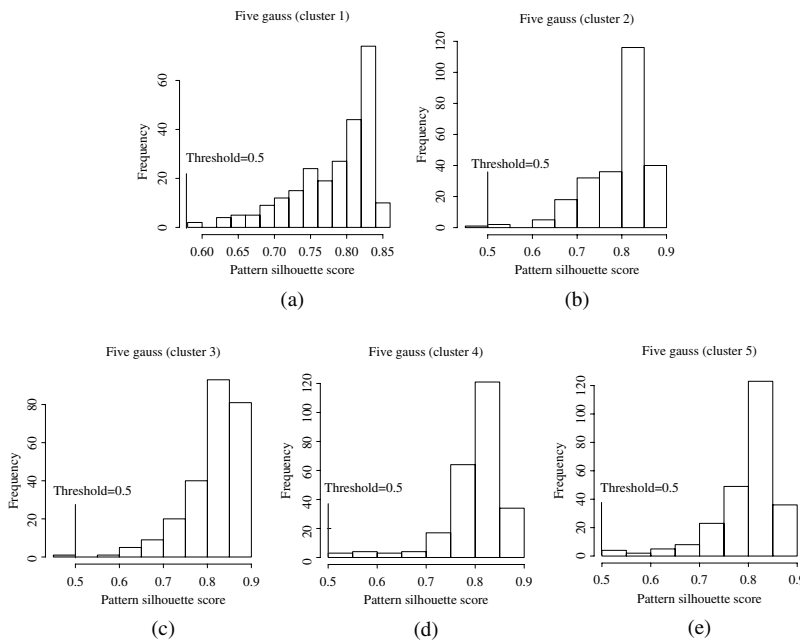


Figure 4.14. *Histograms of silhouette values in the five Gaussians datasets, which were used for the determination of the clusters' silhouette thresholds in the cluster pruning approach*

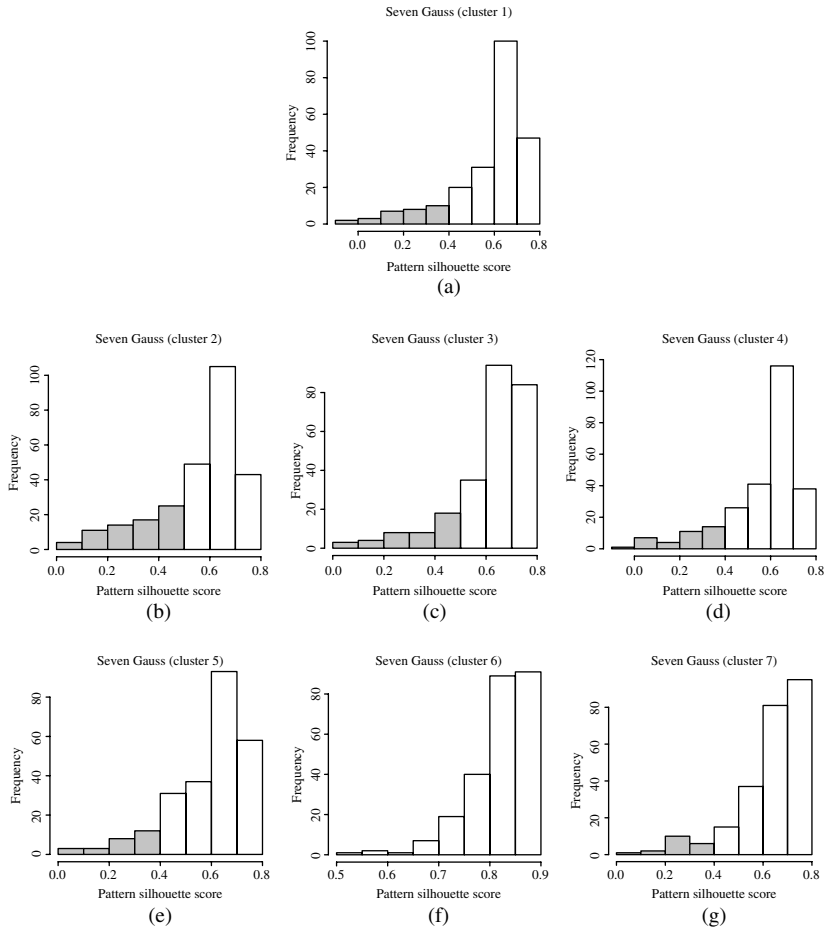


Figure 4.15. *Histograms of silhouette values in the seven Gaussian datasets, which were used for the determination of the clusters' silhouette thresholds in the cluster pruning approach. Shadow histogram bars correspond to the silhouette values of the rejected patterns*

4.7.2. *Evaluation of the cluster pruning approach*

In this section, the efficiency of the cluster pruning method for rejecting misclassification errors from the clustered data

is evaluated through an analysis of the algorithm outcomes on the iris, wine, breast cancer, diabetes, Pendig, and seven Gaussian datasets. [4]

For the purpose of evaluating the cluster pruning algorithm, the cluster labeling task has been performed using the complete set of labels for each dataset. The resulting misclassification error rates as well as the NMI results observed in Table 4.1 confirm the adequate behavior of the cluster pruning algorithm for removing such sections from the clusters with a high probability of resulting misclassification errors after cluster labeling. For instance, while the pruned sections comprise around 10–20% of the patterns in the datasets, the percentage of remaining misclassification errors

Data set	Silhouette thresholds	% Removed patterns	Error 1 (%)	Error 2 (%)	NMI 1	NMI 2
Iris	[0.5 0.3 0.4]	17.33%	10.66 %	4.03%	0.758	0.888
Wine	[0.2 0.14 0.24]	22.40 %	8.98 %	0.72%	0.783	0.967
Breast	[0.5 0.2]	11.56 %	4.09 %	0.99%	0.741	0.910
Diabetes	[0.5 0.1]	16.35 %	40.10%	38.16%	0.012	0.022
Pendig	[0.2 0.3 0.2 0.2 0.25 0.2 0.15 0.15 0.25 0.2]	20.10 %	31.93%	21.22%	0.701	0.796
Seven Gaussians	[0.4 0.5 0.4 0.4 0.4 0.4 0.4]	11.77 %	0.27% %	0.02%	0.944	0.993
Utterances A	[0 0.1]	31.94%	20.48%	6.63 %	0.269	0.64
Utterances B	[0.1 0.05]	32.29%	36.11 %	21.02 %	0.100	0.297
Utterances C	[0.02 -0.02 0 0]	17.39 %	36.20%	29.36%	0.355	0.4652
Utterances D	[0 0.1 0.1 0.15]	38.30 %	28.53 %	30.4%	0.333	0.434

Table 4.1. *Some details about the cluster pruning approach in the Iris, Wine, Breast cancer, Diabetes, Pendig, Seven Gaussians and Utterances data sets*

4. Note that the cluster partitions obtained in there experiments comprise all instances of the datasets (without prior partitions into test/training).

has been substantially reduced. As an example, the error rate has dropped from 10.66 to 4.03% after pruning on the iris data (62.2% error reduction), while error rate reductions of 75.8 and 92% have been attained on the breast and wine datasets, respectively. An exception to the previous observations is the diabetes dataset, in which the error rate after cluster pruning (38.16%) remains very similar to the original misclassification rate (40.10%). Note that, for two clusters, as in the case of the diabetes data, the worst possible error rate that can be observed is of 50%. Any error rate larger than 50% is not observed as it would just produces an inversion of the cluster labels. In other words, the original error in the diabetes dataset implies a roughly uniform distribution of patterns from any of the two underlying classes in the extracted clusters. This fact is also evidenced by the NMI score NMI=0.012. In consequence, the error rate is roughly the same after cluster pruning, and the removal of patterns by means of cluster pruning algorithm is just as efficient as removing the same amount of patterns at random.

4.8. Simulations and results

In the experimental setting, SVMs have been used as the baseline classifier. First, each dataset has been divided into two training ($\sim 90\%$) and test ($\sim 10\%$) sets. In order to avoid possible biases of a single test set, such partition of the dataset has been randomly repeated to generate 20 different partitions. Also, for each one of these partitions, 20 different random seeds of labeled prototypes (n labels/category) have been selected. In total, 400 different prototype seeds (20×20) have been obtained. In the experiments, only prototype labels are assumed to be known *a priori*. No other class label knowledge has been applied to any of the algorithm stages. Each prototype seed has been used as the available training set for the supervised SVM. In the semi-supervised approach,

these labeled prototype seeds have been used to trigger the automatic cluster labeling.

Both supervised and semi-supervised SVM classifiers have been evaluated on an accuracy basis, considering different number of labeled prototypes (samples) per category, from $n = 1$ to $n_{\max} = 30$. The accuracy results obtained on the different datasets are shown in Figures 4.16–4.21. In particular, left plots are referred to the supervised and the semi-supervised approach without cluster pruning, while right plots are referred to the semi-supervised approaches by incorporating the cluster pruning approach. In all the cases, horizontal axes are referred to the sizes of the initial prototype seeds, whereas vertical axes indicate the mean accuracy scores, averaged over the 400 prototype initializations.

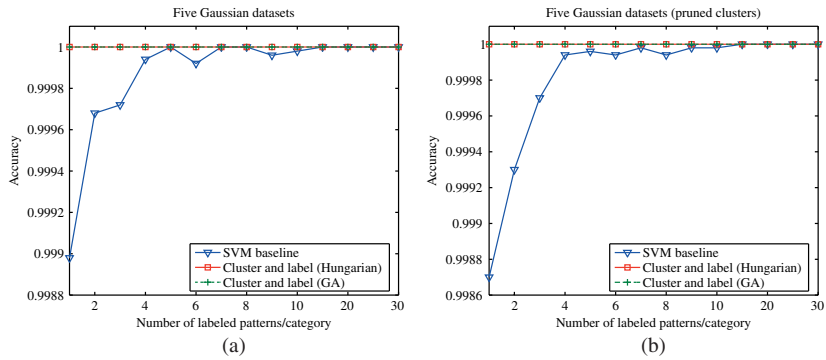


Figure 4.16. *Comparison of the accuracy scores achieved by the supervised SVM and semi-supervised approach on the five Gaussian datasets. (a) Before cluster pruning; (b) after cluster pruning*

As can be observed in these Figures 4.17 and 4.21, the mean accuracy curves of the semi-supervised algorithm are roughly constant with the labeled set size. Only certain random variations can be observed, since the experiment outcomes for different seed sizes are referred to different random prototype seeds. However, it should be noted that, for each labeled set

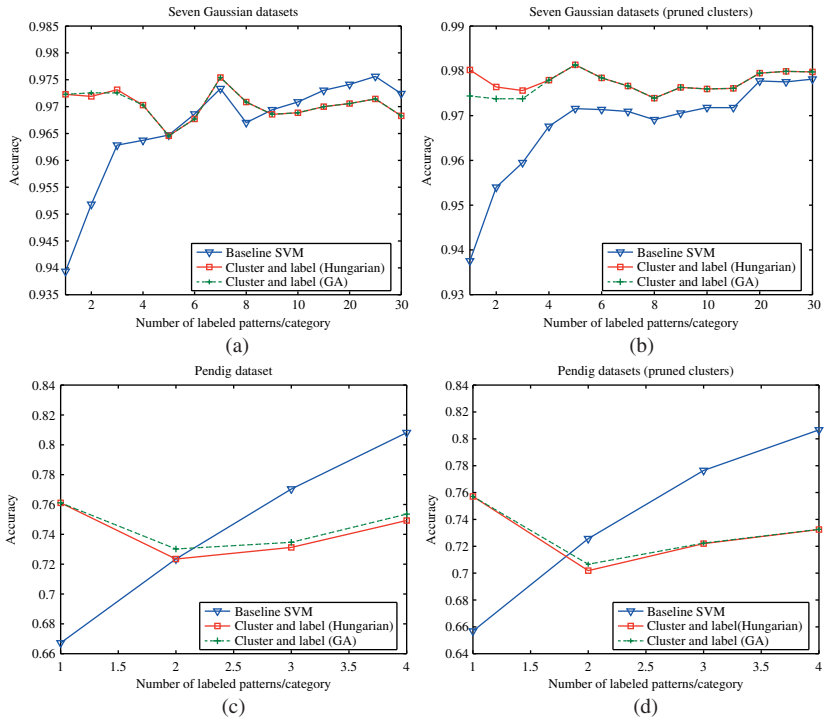


Figure 4.17. *Comparison of the accuracy scores achieved by the supervised SVM and semi-supervised approach on the seven Gaussian and Pendig datasets. (a and c) Before cluster pruning; (b and d) after cluster pruning*

size, both supervised and semi-supervised outcomes have been simultaneously obtained with identical sets of prototypes, so that their respective accuracy curves can be compared. In contrast, the accuracy curves of the supervised approach show an increasing trend with the labeled set sizes. In the seven Gaussian, iris, Pendig, wine, and breast cancer datasets, the mean accuracy curves for the supervised and semi-supervised algorithms intersect at certain labeled set sizes, n'. For smaller labeled seed sizes ($n < n'$), the training "information" available in the augmented labeled sets (after cluster labeling) seems to be clearly superior than the the

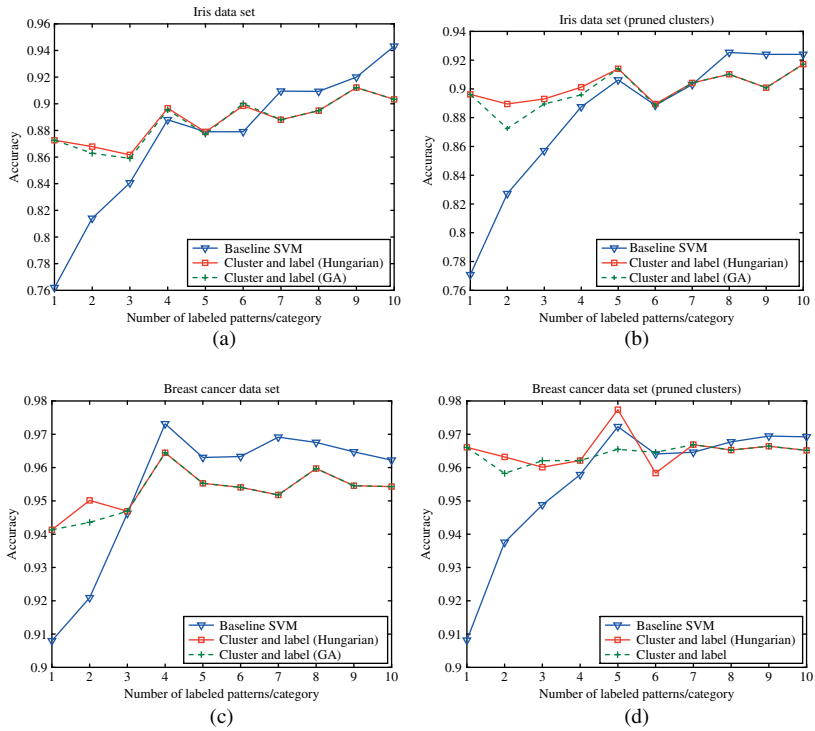


Figure 4.18. *Comparison of accuracy scores achieved by the supervised SVM and semi-supervised approach on the iris and breast datasets. (a and c) Before cluster pruning; (b and d) after cluster pruning*

small labeled seeds. Therefore, although the augmented labels are not exempted from misclassifications due to clustering errors, higher prediction accuracies are achieved by the semi-supervised approach with respect to the supervised classifier. For $(n \geq n')$, the information in the increasing labeled seeds compensates for the misclassification errors present in the augmented sets and thus the supervised classifier outperforms the semi-supervised approach. As shown in the previous section, these errors present in the augmented datasets can be notably reduced by means of cluster pruning.

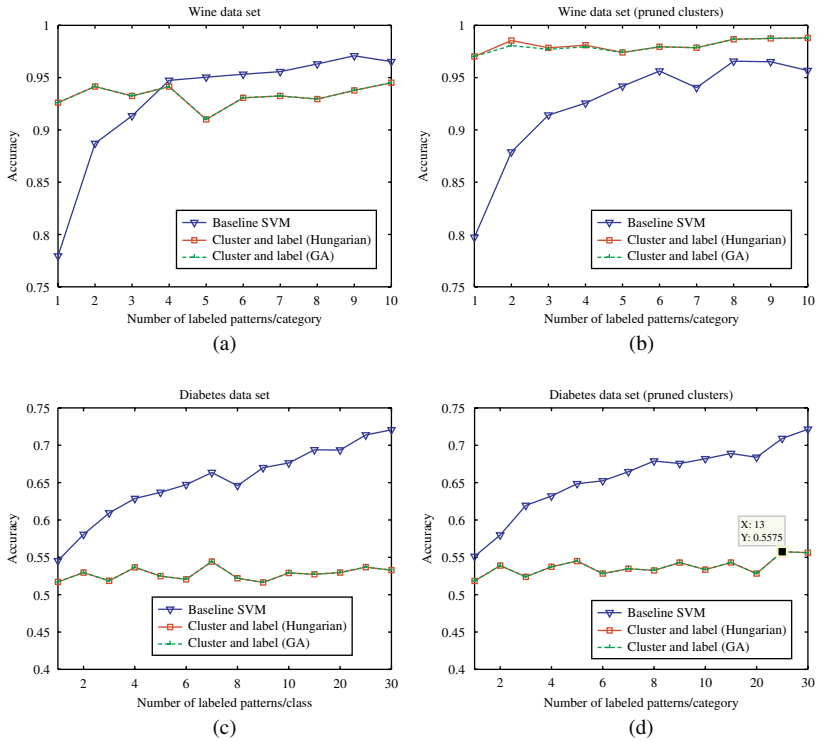


Figure 4.19. *Comparison of accuracy scores achieved by the supervised SVM and semi-supervised approach on the wine and Diabetes datasets. (a and c) Before cluster pruning; (b and d) after cluster pruning*

Consequently, an improvement in the prediction accuracies achieved by the semi-supervised algorithm is generally observed by incorporating the cluster pruning algorithm (Figures 4.16b; 4.17b; 4.17 and 4.18b; 4.18 and 4.19b; and 4.19).

Unlike the accuracy results observed in the seven Gaussian, iris, Pendig, Wine, and breast cancer datasets, a degradation in the semi-supervised classification performance with respect to the supervised classifier is observed in the

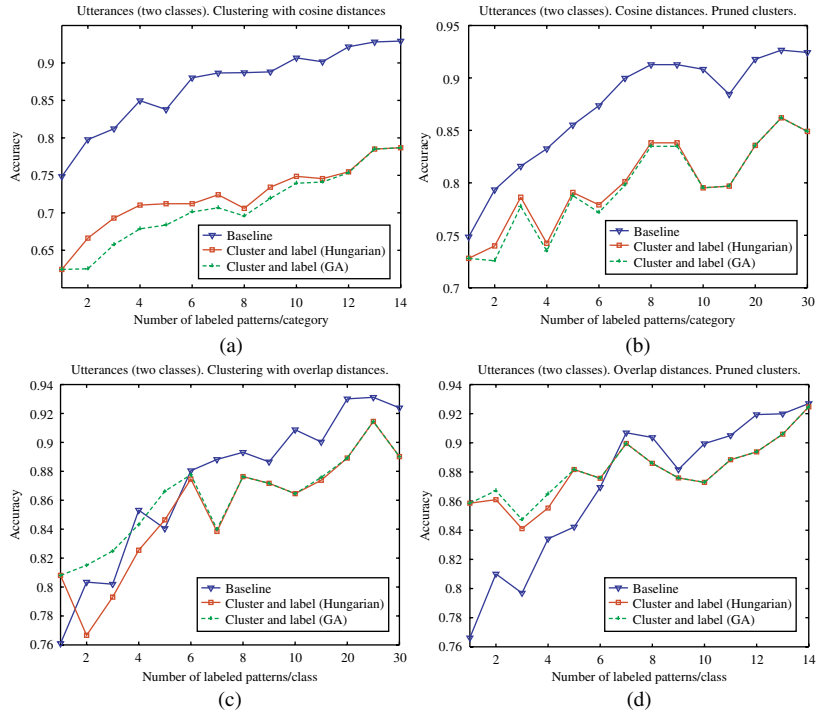


Figure 4.20. *Comparison of accuracy scores achieved by the supervised and semi-supervised approaches in a data set of utterances (2 classes). (a) and (c): before cluster pruning (b) and (d): after cluster pruning*

diabetes dataset, regardless of the initial labeled seed sizes. This observation is strictly associated with the NMI scores of the extracted clusters presented in the previous section (NMI=0.012), which corresponds to a misclassification rate of 40.10%. This means that almost no information concerning class labels is present in the augmented datasets used to train the SVM models. The semi-supervised performance on the diabetes dataset is thus comparable to the trivial classifier. In other words, the main condition for cluster and label approaches – the patterns in a dataset should naturally fall

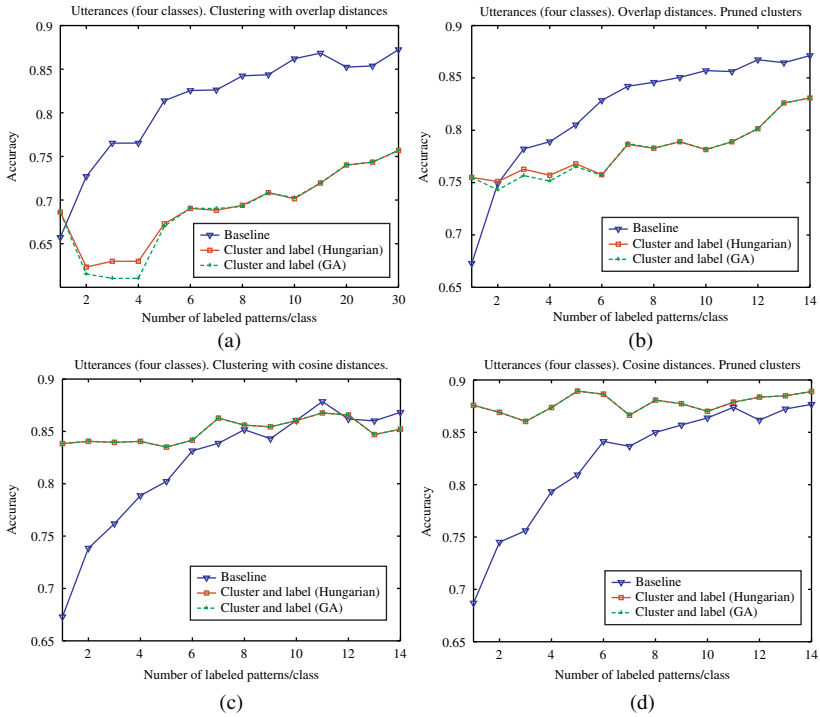


Figure 4.21. *Comparison of accuracy scores achieved by the supervised and semi-supervised approaches in a data set of utterances (4 classes). (a) and (c): before cluster pruning (b) and (d): after cluster pruning*

into clusters – is not fulfilled in the diabetes dataset (using the PAM clustering algorithm).

4.9. Summary

In this chapter, a semi-supervised approach has been presented based on the cluster and label paradigm. In contrast to previous research in semi-supervised classification, in which labels are commonly integrated in the clustering process, the cluster and labeling processes are independent from each other in this work. First, a

conventional unsupervised clustering algorithm, the PAM, is used to obtain a cluster partition. Then, the output cluster partition as well as a small set of labeled prototypes (also refered to as labeled seeds) is used to decide the optimum cluster labeling related to the labeled seed. The cluster labeling problem has been formulated as a typical assignment optimization problem, whose solution may be obtained by means of the Hungarian algorithm or GA. Experimental results have shown significant improvements in the classification accuracy for minimum labeled sets, in such datasets where the underlying classes can be appropriately captured by means of unsupervised clustering.

In addition, an optimization of the semi-supervised algorithm has been also developed by discarding the patterns clustered with small silhouette scores. Thereby, it has been shown that the quality of the pruned clusters can be improved, as significant reductions in the misclassification errors present in the clustered data are achieved through the removal of relatively small amounts of patterns from the clusters.

Future work is to analyze further alternatives for the definition of the cost matrix used by the Hungarian algorithm. Currently, the cost matrix is based on a number of (labeled) patterns from each class that are observed in the clusters. Hence, the Hungarian algorithm provides the solution with the maximum overlapping of clusters and class labels. However, if an inappropriate labeled seed contains a large amount of patterns from class overlapping regions, the optimum solution based on such labeled seed may result in cluster labeling errors (i.e., two or more clusters may be assigned to a different class label to the one which is generally represented by the patterns in the cluster). Obviously, in these situations, the amount of misclassification errors in the augmented datasets leads to incorrect learned models

and false predictions concerning the erroneous class labels. This problem may be palliated, for example, through a new probabilistic definition of the cost matrix, by estimating, for each class, the conditional probabilities of the clusters given the class' labels in the initial seeds.

PART 3

Contributions to Unsupervised Classification – Algorithms to Detect the Optimal Number of Clusters

Chapter 5

Detection of the Number of Clusters through Non-Parametric Clustering Algorithms

5.1. Introduction

As described in Chapter 1, the identification of the optimum number of clusters in a dataset is one of the fundamental open problems in unsupervised learning. One solution to this problem is (implicitly) provided by the pole-based clustering (PoBOC) algorithm proposed by Guillaume Cleizou [CLE 04a]. Among the different clustering approaches described in Chapter 1, the PoBOC algorithm is the only method that does not require the specification of *any kind* of *a priori* information from the user. The algorithm is an overlapping, graph-based approach that iteratively identifies a set of initial cluster prototypes and builds the clusters around these objects based on their neighborhoods.

However, one limitation of the PoBOC algorithm is related to the global formulation of neighborhood applied to extract the final clusters. The neighborhood of one object is defined

Semi-Supervised and Unsupervised Machine Learning: Novel Strategies Amparo Albalate and Wolfgang Minker

in terms of its average distance to all other objects in the dataset (see section 2.1). This global parameter may be suitable for discovering uniformly spread clusters on the data space. However, the algorithm may fail to identify all existing clusters if the input data are organized in a hierarchy of classes, in such a way that two or more subclasses are closer to each other than the average class distance.

To overcome this limitation, a new hierarchical strategy based on PoBOC has been developed called "hierarchical pole-based clustering" (HPoBC). The hierarchy of clusters and subclusters is detected using a recursive approach. First, the PoBOC algorithm is used to identify the clusters in the dataset, also referred to as "poles." Next, under the hypothesis that more subclusters may exist inside any pole, PoBOC is again locally applied to each initial pole. A cluster validity based on silhouette widths is then used to validate or reject the subcluster hypothesis. If the subcluster hypothesis is rejected by the silhouette score, the candidate subclusters are discarded and the initial pole is directly attached to the final set of clusters. Otherwise, the hypothesis is validated and the new identified poles (subclusters) are stored, and a similar analysis is performed inside each one of these poles. This procedure is applied recursively until the silhouette rejects any further hypothesis. The HPoBC method is explained in more detail in the following sections and is compared with other traditional algorithms (hierarchical single, complete, average, and centroid linkages as well as the partitioning around medoids algorithms).

5.2. New hierarchical pole-based clustering algorithm

The new clustering method is a combination of the PoBOC algorithm and hierarchical divisive clustering strategies.

In a divisive manner, the proposed hierarchical approach is initialized with the set of poles identified by PoBOC (see Chapter 1) and is recursively applied to each obtained pole, searching for possible subclusters.

5.2.1. *Pole-based clustering basis module*

To detect the set of poles in the hierarchical method HPoBC, the graph construction, pole construction, and pole restriction stages of POBOC have been preserved. However, the affectation step has been replaced by a new procedure called *pole regrowth*:

Algorithm 5.1 Pole-regrowth $(\tilde{\mathcal{P}}, \mathcal{R}, D)$

Input: sets of poles and residual from the pole-reduction step: $\tilde{\mathcal{P}}, \mathcal{R}$; dissimilarity matrix D
Output: set of regrown poles $\hat{\mathcal{P}}$
Initialise: $\hat{\mathcal{P}} = \tilde{\mathcal{P}}$
while $\mathcal{R} \neq \emptyset$ **do**
  Find the pair $(x_i \in \mathcal{R}, \hat{P}_j \in \hat{\mathcal{P}})$ with minimum distance:
    $(x_i, \hat{P}_j) = \text{argmin}_{x \in \mathcal{R}, \hat{P} \in \hat{\mathcal{P}}} D_{min}(x, \hat{P})$,
    with $D_{min}(x_i, \hat{P}_j) = \min_{x_k \in \hat{P}_j} D_{ik}$
  Attach the point x_i to its closest pole and remove it from the residual set:
    $\hat{P}_j = \hat{P}_j \cup x_i$
    $\mathcal{R} = \mathcal{R} - x_i$
end while
Return $\hat{\mathcal{P}}$

The pole-regrowth procedure is an alternative to the PoBOC single affectation for re-allocating overlapping objects into one of the restricted poles. As it can be observed in Figure 5.1(a), not only a pole but also an overlapping region may contain potential subclusters. If each overlapping object

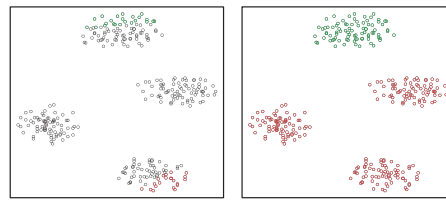


Figure 5.1. *Example of the pole growth. (a) Two restricted poles (red and green circles) and their overlapping objects (black circles) – from the database 1000p9c, see Figure 5.2(q). (b) New poles obtained after the re-allocation of the overlapping objects by the pole regrowth method*

x_i is individually assigned to the pole maximizing membership $u(x_i, \tilde{P})$, the objects inside a single cluster might be assigned to different poles.[1] The pole-regrowth procedure is intended to avoid any undesired partitioning of clusters existing in overlapping areas while re-allocating residual objects.

An example of the pole-regrowth method is shown in Figure 6.3. Figure 5.1(a) shows two restricted poles in red and green colors, respectively. All points between these restricted poles are overlapping points. It can be observed that many of these points build another two clusters, which PoBOC fails to detect. The re-allocation of overlapping points by the pole-regrowth procedure is illustrated in Figure 5.1(b). A single affectation would have split each overlapping cluster into two halves (upper and bottom). Using the pole-regrowth method, all objects inside each overlapping cluster have been jointly assigned to a single pole. This fact allows us to detect the overlapping clusters in further recursive steps.

1. Note that the hierarchical approach is independently applied to the grown poles.

We refer to the modified PoBOC algorithm as pole-based clustering module, which is the basis for the hierarchical approach described in the following paragraphs.

Algorithm 5.2 Pole-based clustering module ($\mathcal{X}$)

Input: set of data points to be clustered $\mathcal{X}$
Output: set of regrown poles $\hat{\mathcal{P}}$
Compute dissimilarity matrix of $\mathcal{X}$: D
Compute dissimilarity graph over $\mathcal{X}$, $G(\mathcal{X}, V, D)$
$\mathcal{P} \leftarrow$ Pole Construction $(\mathcal{X}, D, G(\mathcal{X}, V, D))$
$\tilde{\mathcal{P}}, \mathcal{R} \leftarrow$ Pole Restriction $(\mathcal{P})$
$\hat{\mathcal{P}} \leftarrow$ Pole Regrow $(\tilde{\mathcal{P}}, \mathcal{R}, D)$
Return $\hat{\mathcal{P}}$.

5.2.2. *Hierarchical pole-based clustering*

The new hierarchical version of PoBOC is called HPoBC.

First, the pole-based clustering module is applied to the entire dataset to obtain an initial set of poles. Then, a recursive function, the *pole-based subcluster analysis*, is triggered on each pole with more than one object. If an individual pole is found, the corresponding object is attached to the set of final clusters as an individual cluster. This recursive function is continuously called with the objects of each obtained pole, internally denoted $P^{\hat{top}}$, because it refers to an upper level in the hierarchy. Analogously, the new set of poles found on $P^{\hat{top}}$ is denoted $P^{\hat{sub}}$, indicating a lower hierarchy level. These poles represent candidate subclusters. To decide whether $P^{\hat{sub}}$ compounds are "true" subclusters or not, a criterion typically used for cluster validity is applied, namely, the *average silhouette width*, described in Chapter 1 [ROU 87]. The average silhouette width of a cluster partition returns a quality score in the range [−1,1], where 1 corresponds to a perfect clustering. According to [TRE 05], a silhouette score smaller or equal to sil = 0.25 is an indicator for wrong cluster solutions. However,

from our experiments, a more rigorous threshold sil > 0.5 has proven adequate for validating the candidate subclusters.[2] The problem of deciding whether a dataset contains a cluster structure or not is commonly referred to as cluster tendency in the cluster literature [JOL 89]. If the quality criterion is not fulfilled (sil < 0.5) the subcluster hypothesis is rejected, and the top cluster $\hat{P^{top}}$ is attached to the final clusters. Otherwise, we continue exploring each subcluster to search for more possible sublevels.

Algorithm 5.3 Hierarchical pole-based clustering – HPoBC ($\mathcal{X}$)

Input: Set of data points to be clustered $\mathcal{X}$
Output: A cluster partition of $\mathcal{X}$: Clusters
Initialise: Clusters = $\{\emptyset\}$
Obtain set of grown poles on all $\mathcal{X}$ objects:
$\hat{\mathcal{P}} \leftarrow$ Pole-Based Clustering Module ($\mathcal{X}$)
for all $\hat{P}_i \in \hat{\mathcal{P}}$ **do**
  if $|\hat{P}_i| > 1$ **then**
    Trigger recursive search for subclusters:
    Pole-Based Subcluster Analysis ($\hat{P}_i$, Clusters)
  else
    Add $\hat{P}_i$ to Clusters
  end if
end for
Return Clusters

5.3. Evaluation

The PoBOC algorithm as well as the HPoBC algorithm has been compared with other hierarchical approaches:

2. The relationship of the *average silhouette width* threshold applied in this chapter with the *pattern silhouette scores* described in Chapter 3 for cluster pruning has to be noted. Although a *pattern silhouette* is only the first metric involved in the calculation of the clustering *average silhouette width*, in Chapter 3, it was also shown that a pattern silhouette equal to sil = 0.5 indicates well-clustered patterns.

Algorithm 5.4 Pole-based subcluster analysis ($\hat{P}^{top}$, Clusters)

$\hat{\mathcal{P}^{sub}} \leftarrow$ Pole-Based Clustering Module ($\hat{P}^{top}$)
stop $\leftarrow$ (silhouette-width ($\hat{\mathcal{P}^{sub}}$) ≤ 0.5)
if stop=true **then**
  Add $\hat{P}^{top}$ to Clusters
  Return
else
  for all $\hat{P}_i^{sub} \in \hat{\mathcal{P}^{sub}}$ **do**
    if $|\hat{P}_i^{sub}| > 1$ **then**
      Pole-Based Subcluster Analysis ($\hat{P}_i^{sub}$, Clusters)
    else
      Add $\hat{P}_i^{sub}$ to Clusters
      Return
    end if
  end for
end if

the single, complete, centroid, and average linkage and the divisive analysis (DiANA) algorithm. These classical algorithms are examples of clustering approaches that require the target number of clusters (k) to find the cluster solutions. To enable a comparison of PoBOC and HPoBC to the hierarchical agglomerative and divisive approaches, these algorithms have been called with different values of the k-parameter. Thus, the average silhouette width has been applied to validate each solution and predict the optimum number of clusters, k_{opt}. It should be noted that, in the HPoBC algorithm, the average silhouette width has also been applied with a different purpose. Instead of using the silhouette width as a (global) validation index, it has been applied in a recursive and *local* manner to evaluate the cluster tendency inside each obtained pole.

5.3.1. *Cluster evaluation metrics*

For a comprehensive evaluation of the discussed algorithms, their cluster solutions have also been compared with the

reference category labels, available for evaluation purposes, using three typical external cluster validation methods: entropy, purity, and normalized mutual information (NMI) (see Chapter 2).

5.4. Datasets

The described approaches have been applied to the synthetic datasets of Figure 5.2. The first dataset (100p5c) comprises 100 objects in 5 spatial clusters (Figure 5.2(a)), the second dataset (6 Gauss) is a mixture of six Gaussians (1500 points) in two dimensions (Figure 5.2(e)). The third dataset (3 Gauss) is a mixture of three Gaussians (800 points) in which the distance of the biggest class to the other two is larger than the distance among the two smaller Gaussians (Figure 5.2(i)). This dataset illustrates a typical example in which using cluster validity based on Silhouettes may fail to predict the number of classes due to the different interclass distances. The fourth and fifth data (560p8c and 1000p9c) contain 560 and 1000 points in two dimensions, with 8 and 9 spatial clusters, respectively (Figure 5.2(m) and (q)).

The cluster solutions provided by the PoBOC, the HPoBC, and an example hierarchical agglomerative approach (average linkage) are shown in the plots of Figure 5.2 (different colors are used to indicate different clusters).

5.4.1. *Results*

As can be seen in Tables 5.1–5.5, the performance of the HPoBC algorithm is consistently superior to the original PoBOC algorithm on all datasets and metrics. The classical (divisive and agglomerative) approaches with the help of silhouettes to determine the optimum k are also able to detect the class structure in three datasets (100p5c, 1000p9c, and 6 Gauss). The performance of classical approaches is

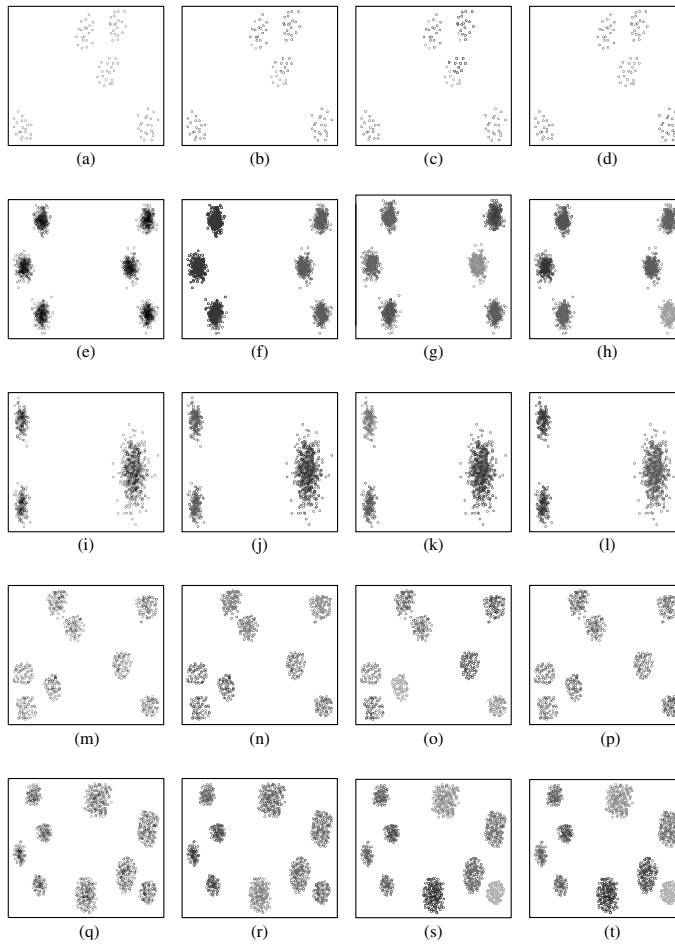


Figure 5.2. *Spatial databases and the extracted clusters using PoBOC, HPoBC, and the average linkage clustering algorithms. (a) Dataset with 100 points in 5 spatial clusters (100p5c), (e) mixture of six Gaussians, 1500 points (6 Gauss), (i) mixture of three Gaussians (3 Gauss), (m) 560 points, 8 clusters (560p8c), and (q) 1000 points, 9 clusters (1000p9c). (b), (f), (j), (n), and (r) Poles detected by PoBOC in the datasets. (c), (g), (k), (o), and (t) Clusters detected by the new HPoBC algorithm (black circles indicate patterns detected as outliers by the algorithms). (d), (h), (l), (p), and (s) Clusters detected by the average linkage algorithm*

Clustering algorithm	# Clusters	NMI	Purity	Entropy
PAM	5	0.850	0.660	0.840
DiANA	5	0.850	0.660	0.840
Single linkage	5	0.850	0.660	0.840
Complete linkage	5	0.850	0.660	0.840
Average linkage	5	0.850	0.660	0.840
Centroid linkage	5	0.850	0.660	0.840
PoBOC	4	0.801	0.548	1.048
HPoBC	7	0.944	0.867	0.287

Table 5.1. *560p8c Data*

thus comparable to the HPoBC algorithm on the explained datasets. Note that, in some cases, the NMI score achieved by the HPoBC is marginally lower ($\leq 1.8\%$) than other hierarchical approaches, due to the false discovery by the HPoBC of tiny clusters in the boundaries of a larger cluster. In contrast to the previous datasets, the DiANA and agglomerative hierarchical approaches fail to capture accurately the existing classes on the datasets 560p8c and 3 Gauss. The problem lies in the silhouette scores, which fail to place the maximum (k_{opt}) at the correct number of clusters. This happens because the intraclass separation differs significantly among the clusters. However, this problem is not observed in the HPoBC algorithm, since silhouette scores are used to evaluate the local cluster tendency. This implies a more "relaxed" condition in comparison to the use of silhouettes for validating global clustering solutions. Thus, in these cases, the HPoBC algorithm is advantageous with respect to the classical hierarchical approaches, as evidenced by absolute NMI improvements around 10%.

Clustering algorithm	# Clusters	NMI	Purity	Entropy
PAM	5	1	1	0
DiANA	5	1	1	0
Single linkage	5	1	1	0
Complete linkage	5	1	1	0
Average linkage	5	1	1	0
Centroid linkage	5	1	1	0
PoBOC	3	0.801	0.693	0.817
HPoBC	5	1	1	0

Table 5.2. *100p5c Data*

Clustering algorithm	# Clusters	NMI	Purity	Entropy
PAM	6	1	1	0
DiANA	6	0.980	0.992	0.049
Single linkage	6	1	1	0
Complete linkage	6	1	1	0
Average linkage	6	1	1	0
Centroid linkage	6	1	1	0
PoBOC	3	0.606	0.693	0.817
HPoBC	7	0.982	1	0

Table 5.3. *Mixture of six Gaussians*

5.4.2. *Complexity considerations for large databases*

Denoting n as the total number of objects in the dataset, the complexity of the PoBOC algorithm is estimated in the order of $O(n^2)$, similar to the classical hierarchical schemes. The complexity of the HPoBC algorithm depends on factors such as the number and size of poles retrieved at each step and the maximum number of recursive steps necessary to obtain the final cluster solution. The worst case in terms of the algorithm efficiency would occur if a pole with $n - 1$

Clustering algorithm	# Clusters	NMI	Purity	Entropy
PAM	6	1	1	0
DiANA	9	1	1	0
Single linkage	9	1	1	0
Complete linkage	9	1	1	0
Average linkage	9	1	1	0
Centroid linkage	9	1	1	0
PoBOC	5	0.837	0.634	0.637
HPoBC	11	0.993	1	0

Table 5.4. *1000p9c*

Clustering algorithm	# Clusters	NMI	Purity	Entropy
PAM	2	0.847	0.812	0.375
DiANA	2	0.847	0.812	0.375
Single linkage	2	0.847	0.812	0.375
Complete linkage	2	0.847	0.812	0.375
Average linkage	2	0.847	0.812	0.375
Centroid linkage	2	0.847	0.812	0.375
PoBOC	2	0.847	0.812	0.375
HPoBC	4	0.990	1	0

Table 5.5. *Mixture of 3 Gaussians*

elements was continuously found until all elements composed individual clusters. In this case, the algorithm would reach a cubic complexity $O(n(n+1)(2n+1))$. In general terms, if k is the number of recursive steps (levels descended in the hierarchy) necessary to reach the solution, the maximum complexity of the algorithm can be approximated as $O(k \cdot n^2)$. As for the analyzed datasets, the algorithm needed three recursive steps at most to achieve the presented results.

It leads to a quadratic complexity, comparable to the PoBOC algorithm and the rest of hierarchical approaches.

5.5. Summary

In this chapter, clustering algorithms that are capable to detect the optimum number of clusters have been investigated. In particular, the focus is placed on the PoBOC, which only needs the objects in a dataset as input, in contrast to the rest of clustering algorithms discussed in Chapter 1, which require the number of clusters or other equivalent input parameters. One limitation of PoBOC is, however, related to the use of global object distances. The algorithm may fail to recognize the optimum clusters if the data are organized in a hierarchy of clusters or if the clusters show very dissimilar intercluster distances. To overcome this limitation of PoBOC, a hierarchical version has been introduced, called HPoBC, which applies recursively inside each identified cluster to adapt the object distances to local regions and accurately retrieve clusters as well as subclusters. Results obtained on the five spatial databases have proven the better performance of the new hierarchical approach with respect to the baseline PoBOC, also comparable or superior with respect to other traditional hierarchical and partitional (PAM) approaches. Other algorithms such as the self-organizing map, neural gas, or density-based approaches have not been considered for comparison due to their difficulty in analysis, since they depend on other (or more) parameters such as the number of clusters. Thus, the comparison by means of silhouettes is difficult to control.

Chapter 6

Detecting the Number of Clusters through Cluster Validation

6.1. Introduction

The general approach to the identification of the number of clusters by means of cluster validation is to evaluate the quality of each k-cluster solution provided by the clustering algorithm and to select the value of k that originates the optimum partition according to the quality criterion [HAL 00]. Over the past decades, many approaches to cluster validation have been proposed in parallel to the advances in clustering techniques. Some of the most popular approaches have been introduced in Chapter 1, namely, the Dunn index [DUN 74, BEL 98, HAV 08], the Krzanowski and Lai test [KRZ 85], the Davies Bouldin score [DAV 79, HAL 02b], the Hubert's γ [HAL 02a], the silhouette width [ROU 87], or, more recently, the gap statistic [ROB 01] (see chapter 1 for further details). Many of these strategies attempt to minimize/maximize the intra/intercluster dispersion.

Unfortunately, the performance of validation techniques usually depends on the dataset or the cluster algorithm used

Semi-Supervised and Unsupervised Machine Learning: Novel Strategies Amparo Albalate and Wolfgang Minker

for partitioning the data. In addition, the distance metrics applied before clustering has proven a relevant factor for the final cluster solution. It may also influence the cluster validity success in determining the optimum number of clusters. In a few cases, prior assumptions about the dataset can be made. This enables the choice of the best fitting clustering technique and distance model. However, unsupervised models are often applied to more complex, multi-dimensional datasets for which little or no prior assumptions can be made, as it occurs with user utterances in troubleshooting agents.

In this chapter, a validity combination strategy is introduced to predict the number of clusters in a dataset even though no prior assumptions can be made about the clustering technique or distance measure. Our approach to cluster validation is to perform multiple simulations on a dataset varying the distance and clustering technique as well as the number of clusters k. Then, the different partitions obtained from these simulations are evaluated in parallel by several cluster validation criteria, thereby, a validation diversity is achieved, which can be exploited to measure the agreement/consistency of the different scores at each value k. This combination strategy is based on the calculation of quantile statistics of the validation curves, as explained in the following sections.

The individual validation indexes as well as the combination strategy have been first evaluated in four datasets: two synthetic sets (mixtures of five and seven Gaussians), the NCI60 microarray dataset, and the Iris dataset. Finally, the combination approach has been also applied for detecting the number of classes in a corpus of speech utterances from the framework of automated troubleshooting agents.

6.2. Cluster validation methods

In the following, let $\mathcal{C} = \{C_1, \ldots, C_k\}$ denote a cluster partition composed of k clusters and N the total number of objects in a dataset. Some existing techniques for validating the cluster partition are the following.

6.2.1. *Dunn index*

The Dunn index aims to identify the optimal solution that consists of compact and well-separated clusters [DUN 74, BEL 98]. According to this criterion, the index minimizes the intracluster distances while maximizing the intercluster distances. The Dunn index is defined as follows:

$$\text{Dunn}(k) = \frac{\min\limits_{1 \leq i \leq k} \left(\min\limits_{1 \leq i \leq k} (D_{\text{inter}}(C_i, C_j)) \right)}{\max\limits_{1 \leq l \leq k} \left(D_{\text{intra}}(C_l) \right)} \qquad [6.1]$$

where the notations D_{intra} and D_{inter} stand for the intra- and intercluster distances, respectively. According to the Dunn objective, the optimum number of clusters corresponds to value k that *maximizes* Dunn(C).

6.2.2. *Hartigan*

This validation metric was proposed by Hartigan, the inventor of the k-means clustering [BAR 00] for detecting the optimum number of clusters k to apply in the k-means algorithm [HAR 75]:

$$H(k) = \gamma(k) \frac{W(k) - W(k+1)}{W(k+1)}, \quad \gamma(k) = N - k - 1 \qquad [6.2]$$

where $W(k)$ denotes the intracluster dispersion. The dispersion defined as the total sum of square distances of the objects to their cluster centroids. The γ parameter avoids

an increasing monotony with increasing k. In this work, a small modification to the Hartigan metric has been introduced by treating the $W(k)$ parameter as the average intracluster distance.

According to Hartigan, the optimum number of clusters is the smallest k that produces $H(k) \leq \eta$ (typically $\eta = 10$). However, to allow a better alignment of the Hartigan index to other scores in the combination approach, a correction of the index has been applied in the form $Hc(k) = H(k-1)$. Thus, a modification of the optimum criterion has been accordingly applied by maximizing $Hc(k)$. In other words, the new criterion maximizes the relative improvement at k with respect to $k-1$, in terms of decreasing dispersion. This allows for a direct application of the corrected index $Hc(k)$ in the combination approach without resorting to a previous inversion of the scores.

6.2.3. *Davies Bouldin index*

The Davies Bouldin index [DAV 79] was proposed to find compact and well-separated clusters. It is formulated as

$$\mathrm{DB}(k) = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\Delta(C_i) + \Delta(C_j)}{\delta(C_i, C_j)} \right) \qquad [6.3]$$

where $\Delta(C_i)$ denotes the intracluster distance, calculated as the average distance of all the C_i cluster objects to the cluster medoid, whereas the intercluster distance between two clusters $\delta(C_i, C_j)$ is the distance between the medoids of clusters C_i and C_j. The optimum number of clusters corresponds to the minimum value of $\mathrm{DB}(k)$.

6.2.4. *Krzanowski and Lai index*

This metric belongs to the so-called "elbow models" [KRZ 85]. These approaches plot a certain quality function

over all possible values for k and detect the optimum as the point where the plotted curves reach an elbow, i.e. the value from which the curve considerably decreases or increases. The Krzanowski and Lai index is defined as

$$\mathrm{KL}(k) = \left|\frac{\mathrm{Diff}_k}{\mathrm{Diff}_{k+1}}\right| \quad [6.4]$$

$$\mathrm{Diff}_k = (k-1)^{\frac{2}{m}} W_{k-1} - k^{\frac{2}{m}} W_k \quad [6.5]$$

The parameter m represents the feature dimensionality of the input objects (number of attributes), and W_k is calculated as the within-group dispersion matrix of the clustered data

$$W_k = \sum_{i=1}^{k} \sum_{j \in C_i} (x_{ij} - c_i)(x_{ij} - c_i)^{\mathrm{T}} \quad [6.6]$$

In this case, x_{ij} represents an object assigned to the ith cluster and c_i denotes the centroid or medoid of the ith cluster. The optimum k corresponds to the maximum of $\mathrm{KL}(k)$.

6.2.5. *Silhouette*

This method is based on the *silhouette width*, an indicator for the quality of each object i [ROU 87]. The silhouette width is defined as

$$\mathrm{sil}(x_i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad [6.7]$$

where $a(i)$ denotes the average distance of the object i to all objects of the same cluster, and $b(i)$ is the average distance of the object i to the objects of the closest cluster.

Based on object silhouettes, we may extend the silhouette scores to validate each individual cluster using the average of

the cluster object silhouettes

$$\text{sil}(C_j) = \frac{1}{|C_j|} \sum_{x_i \in C_j} \text{sil}(x_i) \qquad [6.8]$$

Finally, the silhouette score that validates the whole partition of the data is obtained by averaging the cluster silhouette widths

$$\text{sil}(k) = \frac{1}{k} \sum_{r=1}^{k} \text{sil}(C_r) \qquad [6.9]$$

The optimum k is detected by maximizing $\text{sil}(k)$.

6.2.6. *Hubert's* γ

The Hubert's γ statistic [HAL 02a, CHE 06] calculates the correlation of the distance matrix, D, to an *a priori* matrix, Y,

$$Y(i,j) = \begin{cases} 0, & \text{objects } i \text{ and } j \text{ of the same cluster} \\ 1, & \text{otherwise} \end{cases} \qquad [6.10]$$

The γ statistic is formulated as

$$\gamma(k) = \sum_{i=1}^{N} \sum_{j=i+1}^{N-1} D(i,j)X(i,j) \qquad [6.11]$$

where N denotes the total number of objects in the dataset. The optimum number of clusters produces a maximum of the γ statistic.

6.2.7. *Gap statistic*

The idea behind the gap statistic is to compare the validation results of the given dataset to an appropriate reference dataset drawn from an *a priori* distribution, thereby, this formulation tries to avoid the increasing or decreasing monotony of other validation scores with increasing number of clusters.

First, the intracluster distance is averaged over the k clusters

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} \sum_{i,j \in C_r} D(i,j) \qquad [6.12]$$

where n_r denotes the number of elements of the cluster r. The gap statistic is defined as

$$\text{Gap}(k) = E(\log(W_k)) - \log(W_k) \qquad [6.13]$$

where $E(\log(W_k))$ is the expected logarithm of the average intracluster distance. In practice, this expectation is computed through a Monte-Carlo simulation on a number of sample realizations of a uniform distribution B [1]

$$\text{Gap}(k) = (1/B) \sum_{b} E(\log(W_{kb})) - \log(W_k) \qquad [6.14]$$

where W_{kb} denotes the average intracluster distance of the bth realization of the reference distribution using k clusters. The optimum number of clusters is the smallest value k such that $\text{Gap}(k) \geq \text{Gap}(k+1) - s_{k+1}$, where $s_k = \sqrt{1 + 1/B} sd_k$ is a factor that takes into account the standard deviation of the Monte-Carlo replicates (W_{kb}).

1. Note that the reference data drawn from this uniform distribution consists of a number, N, of objects identical to the dataset, with identical number of features m. The values of each feature in each object are assigned randomly in the original feature range.

6.3. Combination approach based on quantiles

As already introduced in section 6.1, a limitation of the validation indices is that their individual performances may vary significantly, depending on the dataset and/or clustering algorithm used for partitioning the data. According to [ROB 01] many validation indices are, to some extent, ad hoc approaches that were defined for specific problems, by assuming certain characteristics of a dataset or for their joint application with a given clustering algorithm. An example is the Hartigan index and the k-means clustering. Moreover, the clustering solution provided by the clustering algorithms may also depend on a particular distance function used to compute object dissimilarities. Therefore, the distance function is an additional factor that may influence the validation performance.

In this work, a combination approach has been implemented to achieve a robust solution. It does not require prior knowledge about the dataset nor the selection of a validation index, clustering method, or distance technique. The combination scheme aggregates multiple validation results obtained with the different validation indices, varying the clustering algorithm and distance functions. Any attempt to combine existing methods to increase the robustness of the solutions is in general known as the "knowledge reuse framework" [STR 02].

– *Clustering techniques*: In this work, four clustering algorithms have been used (from the R statistics package cluster) – the partitioning around medoids (PAM) algorithm [LEO 05] and the hierarchical complete, centroid, and average linkage methods [ALD 64, HAS 09, WAR 63].

– *Distance functions*: The aforementioned algorithms have been applied to two different distance matrices representing the dissimilarities between the dataset objects. To compute

the distance matrices, two popular dissimilarity metrics have been selected – the Euclidean and cosine distances.

– *Validity indices*: Finally, some validity indices have been selected – the Hartigan, the Davies Bouldin, the Krzanowski and Lai test, the silhouete, and the gap statistic, which showed less strong monotony effects than the Dunn and Hubert´s γ indices in the analyzed datasets.

The different clusterings of the data obtained with the different clustering techniques and distance functions have been evaluated in parallel with the aforementioned validity indices [2]. This results in a diversity of validity outcomes, also referred to "validity curves." In the following, let $\{\mathcal{V}\}$ denote the set of validity curves obtained from the previous validation experiments. Hence, each validity curve V_i indicates the validation scores obtained with each triple t (clustering algorithm, distance, and validity index), as a function of the number of clusters k. Note that Davies Bouldin scores have been inverted before applying the combination approach, so that the optimum can be generalized to the maximum scores and that the gap has been presented to the combination algorithm as

$$\text{Gap}'(k) = \text{Gap}(k) - \text{Gap}(k+1) + s_{k+1} \qquad [6.15]$$

The proposed method has been motivated by the observation that, although individual validation curves may fail to determine of the optimum k, k_{opt}, this value is located among the top scores in many cases. This fact suggested the combination of validation scores by means of p quantiles.

2. Note: While Hartigan, Krzanowski and Lai, Davies, and silhouette have been used in combination with the four clustering algorithms, the gap statistic has been only applied with the PAM and average linkage algorithms.

The p quantile of a random variable X is defined as value x which is only exceeded by a proportion $1-p$ of the variable samples [SER 80,FRO 00]. In mathematical terms, if denoting the probability density function of the random variable X as $pdf(X)$, the p quantile is defined as

$$Q(X,p) = x : \quad \int_{-\infty}^{x} pdf(X) = p \qquad [6.16]$$

Figure 6.1 illustrates this concept for an hypothetic random variable with a normal distribution of mean = 500.

For the application of quantiles to the detection of the number of clusters, the different validation curves are treated as random variables. The quantile function is then applied to each single curve, V_i. The p quantile $Q(V_i,p)$ equals the validation score V_{i_p} only when exceeded by the $1-p$ proportion of k values in the considered range. This is exemplified in Figure 6.2 for the validation curve obtained by applying the Hubert's γ metric to a mixture of five Gaussians using the pair PAM-cosine.

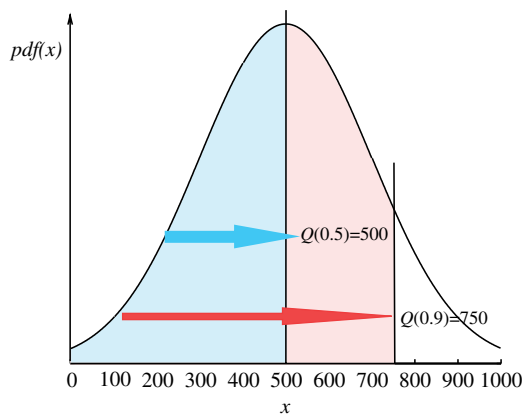


Figure 6.1. *Illustrative example of a p quantile: 0.5 and 0.9 quantiles of variable samples with normal distribution of mean = 500*

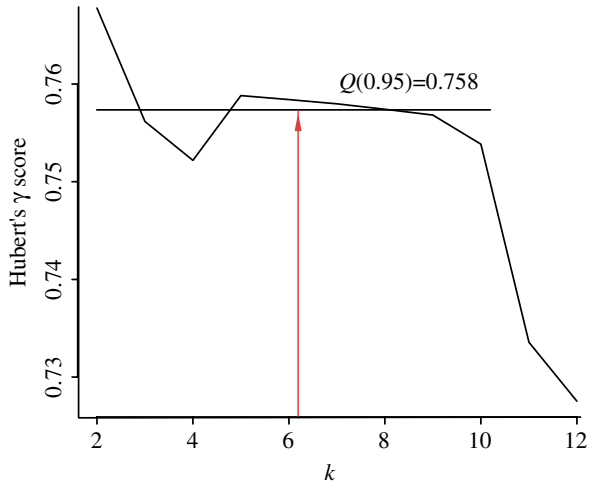


Figure 6.2. *Application of quantiles to a validity curve (Hubert's γ index, PAM clustering, and cosine distance on a mixture of five Gaussians). "Top scores" can be identified as such scores that exceed the p quantile level. In this example, $p = 0.90$*

A basic approach to measuring the consensus of validity scores could be achieved by directly applying p quantiles to the set of validation curves and counting the number of times that each k value outperforms the score $Q(V_i, p)$. This method has been called *quantile validation (Qvalid(V,p))*.

However, the $Qvalid$ results show a certain dependency with the quantile probability parameter p. For example, using low p values often leads to maximum scores at the optimum k_{opt}. However, maximum scores are also observed at undesired k, since there is a high proportion of samples that usually trespass the levels $Q(V_i, p)$ for low p. In contrast, if a high p value is selected, a maximum peak can be clearly discerned. However, this maximum might be misplaced at $k \neq k_{\text{opt}}$. This happens, in particular, if an increasing/decreasing monotony with k is observed in some validity outcomes. These

Algorithm 6.1 Quantile validation: $Qvalid(\mathcal{V}, p)$

Input:
$\mathcal{V}$: set of validation curves
p: quantile parameter
for k=2 to $k_{\max}$ **do**
  Qval[k] = 0
  for all V_i **do**
    if $V_i[k] \geq$ quantile(p, V_i) **then**
      Qval$[k] \leftarrow$Qval$[k] + 1$
    end if
  end for
end for
Output: Qval

monotony effects may be captured in the $Qvalid$ result in the form of maximum peaks misplaced at low/high k.[3]

For these reasons a "supra-consensus" function has been proposed, called *quantile detection*, which aims to combine a set of quantile validation results obtained with different p values. Two alternatives have been analyzed to this aim: the first algorithm, $QdetectA$, is performed in two steps: first, the Qvalid algorithm ($Qvalid(\mathcal{V}, p)$) is called with different p values: $p = [0.1, 0.2, \ldots, 0.9]$. Then, the nine different $Qvalid$ outputs are aggregated by counting the total number of

3. Note that p quantiles for very high p values (e.g. $p = 0.9$) pose a strong condition to the validity curves: the captured k values for this high quantile often correspond to the global maximum peaks or k values in a close neighborhood. However, an important proportion of the validation curves may not be capable to place the (global) maximum peak at k_{opt}, as motivated by the present work. Hence, the "agreement" measured by the $Qvalid$ function for high p is considerably lower than the agreements achieved for low p. Owing to this low scores, if monotony effects are observed in some validation curves, it is possible that a "false agreement" is produced with a $Qvalue$, which can even exceed the $Qvalid$ score at k_{opt}.

(global or local) maxima placed at each k across these $Qvalid$ outcomes.

The second alternative, $QdetectB$, is similar to the first algorithm except for a modification based on the scores given by $Qvalid(\mathcal{V}, p = 0.9)$. These scores are now placed with special emphasis with to reject potential spurious peaks in $Qvalid$ scores for low p, which may yield false maxima in the aggregate solutions. In other words, the optimum number of clusters k_{opt} should both lead to numerous maxima peaks of the $Qvalid$ outcomes for any value p while still reaching a considerable $Qvalid$ level for high p ($p = 0.9$).

Algorithm 6.2 Quantile detection(A): $QdetectA(\mathcal{V})$

Input:
$\mathcal{V}$: set of validation curves
for p=0.1 to 0.9 **do**
  $Q_d[p, 2 : k_{\max}]$ = ***Qvalid***(V, p)
  $Q_d[p, 1]$ = 0
  $Q_d[p, k_{\max} + 1]$ = 0
end for

for k=2 to $k_{\max}$ **do**
  QdetectA[k] = 0
  for all p **do**
    if $Q_d[p, k] = \max_{k'}(Q_d[p, k'])$ or
    $Q_d[p, k-1] < Q_d[p, k] > Q_d[p, k+1]$ **then**
      QdetectA$[k] \leftarrow$***QdetectA***$[k] + 1$
    end if
  end for
end for

Output: *QdetectA*

Algorithm 6.3 Quantile detection(B): $QdetectB(\mathcal{V})$

Input:
$\mathcal{V}$: set of validation curves

for p=0.1 to 0.9 **do**
  $Q_d[p, 2 : k_{\max}]$ = Qvalid(V, p)
  $Q_d[p, 1]$ = 0
  $Q_d[p, k_{\max} + 1]$ = 0
end for

{Note: code lines 7 to 11 differ from *QdetectA* (restricted results by using $p = 0.9$ quantiles).}
for all k, p **do**
  if $Q_d[p = 0.9, k] < 0.5 \cdot \max_{k'}(Q_d[p = 0.9, k'])$ **then**
    $QD_p[k]$ = 0
  end if
end for

for k=2 to $k_{\max}$ **do**
  QdetectB[k] = 0
  for all p **do**
    if $Q_d[p, k] = \max_{k'}(Q_d[p, k'])$ or $Q_d[p, k-1] < Q_d[p, k] > Q_d[p, k+1]$ **then**
      QdetectB$[k] \leftarrow$*QdetectB*$[k] + 1$
    end if
  end for
end for
  Output: *QdetectB*

6.4. Datasets

6.4.1. *Mixtures of Gaussians*

These synthetic datasets are mixture of five and seven Gaussians in two dimensions. The five Gaussian data

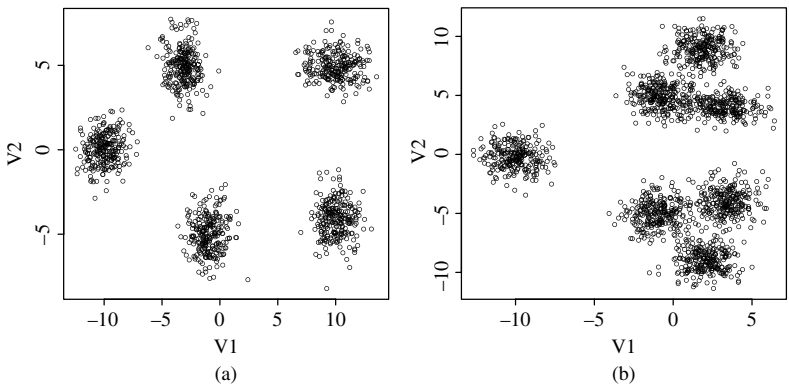


Figure 6.3. *Mixture of Gaussian datasets. (a) Five Gaussians; and (b) seven Gaussians*

comprise five well separable clusters without overlapping objects (Figure 6.3a). In turn, the seven Gaussian dataset contains a hierarchy of Gaussians with three well-differentiated groups and seven groups in which a high overlapping of objects can be observed. This second dataset was intended to evaluate how validation metrics behave on corpora with an underlying hierarchy of classes.

6.4.2. *Cancer DNA-microarray dataset*

The NCI60 dataset [ROS 00] of the University of Standford has been also used (publicly available at [NCI 06]). It consists of gene expression data for 60 cell lines derived from different organs and tissues. The data are a $1,375 \times 60$ matrix where each row represents a gene and each column a cell line related to a human tumor. A dendogram of the 60 clustered cell lines by using a complete-link hierarchical clustering algorithm can be observed in Figure 6.4. Nine known tumor types and one unknown can be distinguished. The cancer types associated with the labels of the dendogram leaves are as follows: LE, leukemia; CO, colon; BR, breast; PR, prostate; LC, lung;

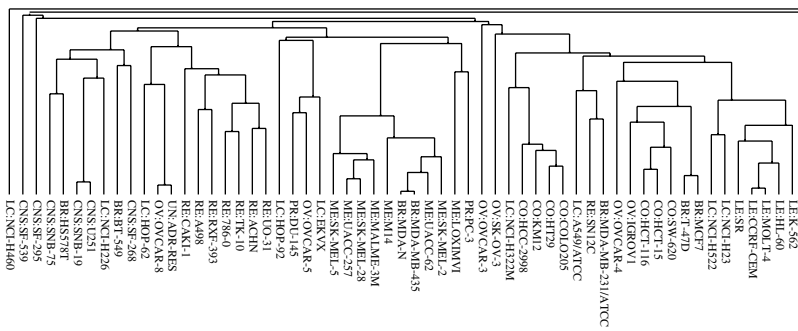


Figure 6.4. *Example dendogram of DNA microarray data obtained by applying hierarchical complete-link clustering to the columns of the data matrix. Nine tumor types plus one unknown can be distinguished. The tumor types associated with the labels of the dendogram leaves are as follows: LE:leukemia; CO:colon; BR:breast; PR:prostate; LC:lung; OV:ovarian; RE:renal; CNS:cns; ME:melanoma; and UN:unknown*

OV, ovarian; RE, renal; CNS, cns; ME, melanoma; and UN, unknown.

6.4.3. *Iris dataset*

The last dataset used in our validation experiments is the Iris dataset from the UCI machine learning repository. The dataset comprises 150 instances with four attributes related to three classes of Iris plants (*Iris setosa, I. versicolor, and I. virginica*). Two of the classes are linearly separable while one of them is not linearly separable from the other two.

6.5. Results

This section presents the validation scores obtained with the validity indices described in section 6.2 and our combination approach. Results obtained with the mixtures of Gaussians, NCI60, and Iris datasets are shown in Tables 6.1–6.4, respectively. The first rows show validation

outcomes obtained with the validation indices (Hartigan, Dunn, Krzanowski and Lai, Hubert's γ, Davies Bouldin, silhouette, and gap statistic) applied in combination with the PAM and average linkage clustering algorithms,[4] using the cosine and Euclidean distances.

Finally, the second last row and the last row show validation scores obtained by the quantile detection approaches (*QdetectA* and *QdetectB*). It should be noted that, although the maximum k value used in our combination approach was $k = 40$, only an excerpt of the validation results for relevant k values close to the optimum is shown in Tables 6.1–6.4. Results outside this range have been omitted due to the limited space. Instead, significant results have been marked with asterisk symbols: (***) for first maxima (minima in the case of Davies Bouldin scores) versus (**) and (*) for second and third maxima, respectively.[5] Also, the columns corresponding to the correct number of clusters in each table have been highlighted in gray.

6.5.1. *Validation results of the five Gaussian dataset*

Results obtained with the mixture of five Gaussians can be observed in Table 6.1.

Most validation curves are able to identify the correct number of clusters ($k = 5$). In addition, the dependency of the

4. Note that the resulting validation curves V have been obtained as a combination of the Pam and hierarchical average, single and centroid linkage methods. However, concerning individual validation curves, only results with the Pam and hierarchical linkage methods are shown in the tables for simplicity purposes.

5. A local maxima is considered at k if the score is higher than the values of $k+1$ and $k-1$. For edge values ($k = 2, k = 40$), a local maxima is considered if these scores are greater than their adjacent in-range neighbors' scores.

Validation index	Clustering, distance	k				
		2	3	4	5	6
Hc(k)	PAM, cos	1282.815	1980.240**	382.633	7962.503***	247.503
	PAM, Euc	−1059.244	414.798**	410.202	1327.809***	19.267
	havg cos	1239.751	2036.104**	879.937	6811.177***	16.123
	havg, Euc	−1053	419.758	407.699	1324.030	2.938
KL(k)	PAM, cos	1.852	2.655	0.452	18.701**	0.767
	PAM, Euc	3.763	0.335	0.549	19.369*	1.031
	havg, cos	1.780	2.266	0.521	11.185***	1.272
	havg, Euc	3.379	0.368	0.572	14.906*	1.002
Dunn(k)	PAM, cos	5.9e-06	8.6e-02***	5.6e-02	2.7e-05	2.7e-05
	PAM, Euc	0.314**	0.009	0.011	0.469***	0.044
	havg, cos	8.2e-02	8.6e-02***	2.6e-02	4.3e-02**	3.1e-02
	havg, Euc	0.314	0.292	0.296	0.469***	0.327
$\gamma(k)$	PAM, cos	0.720	0.838***	0.735	0.676	0.669
	PAM, Euc	0.767	0.749	0.745	0.758***	0.746
	havg, cos	0.716	0.838***	0.764	0.677	0.677
	havg, Euc	0.767***	0.756	0.752	0.758**	0.758
DB(k)	PAM, cos	0.892	0.855	0.721	0.496***	1.402
	PAM, Euc	0.758	0.693	0.540	0.323***	0.551
	havg, cos	0.913	0.855	0.488	0.398***	0.411
	havg, Euc	0.758	0.613	0.498	0.323***	0.333
sil(k)	PAM, cos	0.671	0.828	0.872	0.947***	0.912
	PAM, Euc	0.583	0.578	0.668	0.796***	0.702
	havg, cos	0.674	0.828	0.894	0.960***	0.929
	havg, Euc	0.583	0.590	0.681	0.796***	0.745
Gap'(k)	PAM cos	1.136	−0.474	0.289**	−1.610	0.178
	PAM euc	0.794	−0.052	−0.178	−0.581	0.129***
	PAM euc	0.794	−0.052	−0.178	−0.581	0.129***
	ha euc	0.844	−0.116	−0.118	−0.553	0.137***
QdetectA		0	8**	0	9***	0
QdetectB		0	0	0	9***	0

Table 6.1. *Validation results of the mixture of five Gaussian dataset. The first 28 rows show some of the validation results obtained with the Hartigan, Krzanowski and Lai, Davies Bouldin, Dunn, Hubert's γ, silhouette, and gap statistic indices. Next nine rows show the results of the Qvalid function for $p = 0.1, 0.2, \ldots, 0.9$. Finally, the second last and last rows show the combined results obtained with the QdetectA and QdetectB algorithms*

Validation index	Clustering, distance	k				
		3	4	5	6	7
Hc(k)	PAM,cos	197.4	4441.9***	1085.3	567.1	556.9
	PAM,euc	1010.5	185.6	406.6	265.7	53.1
	havg, cos	3222.8***	581.0	1376.6*	33.8	976.9
	havg, Euc	1897.2***	184.1	388.6*	346.3	526.9**
KL(k)	PAM, cos	0.703	4.510	0.861	1.092	0.535
	PAM, Euc	13.773**	0.694	0.748	3.175	0.207
	havg, cos	4.396*	0.534	0.496	5.781**	0.460
	havg, Euc	52.829***	0.336	0.842	0.770	2.667
DB(k)	PAM, cos	0.908	0.708***	1.671	2.028	1.508
	PAM, Euc	0.484***	0.809	0.855	0.667**	0.859
	havg, cos	0.445***	0.654	0.784	0.810	0.791**
	havg, Euc	0.484***	0.669	0.768	0.684	0.602
Dunn(k)	PAM, cos	1.06e-03	9.78e-05***	9.10e-06	1.70e-05	2.91e-05
	PAM, Euc	0.037	0.473***	0.014	0.018**	0.016
	havg, Euc	0.316	0.473***	0.034	0.034	0.034
	havg, cos	6.29e-02	7.68e-02***	4.70e-04	4.70e-04	9.92e-05
$\gamma(k)$	PAM, cos	0.757	0.762***	0.745	0.661	0.623
	PAM, Euc	0.709	0.815***	0.724	0.699	0.677
	havg, Euc	0.709	0.815***	0.731	0.730	0.730
	havg, cos	0.757	0.763***	0.682	0.681	0.634
Sil(k)	PAM, cos	0.590	0.769***	0.707	0.717	0.737
	PAM, euc	0.684***	0.546	0.487	0.550*	0.538
	havg, cos	0.872***	0.793	0.742	0.668	0.706*
	havg, Euc	0.684***	0.584	0.519	0.570	0.610**
Gap'(k)	PAM, cos	−0.713	−0.059	0.204**	0.193	−0.258
	PAM, euc	0.057**	−0.015	−0.010	0.106***	−0.195
	havg, cos	0.257*	0.074	0.478***	−0.036	0.336**
	havg, Euc	0.148***	−0.056	−0.075	−0.151	0.083**
QdetectA		8***	1	0	0	7**
QdetectB		8**	1	0	2	9***

Table 6.2. *Validation results of the mixture of seven Gaussians The first 28 rows show some of the validation results obtained with the Hartigan, Krzanowski and Lai, Davies Bouldin, Dunn, Hubert's γ, silhouette, and gap statistic indices. For the Hartigan, KL, silhouette width, and gap statitics, the positions of global / local maxima in the range $k = [2, 39]$ are indicated with asterisks. For the Davies Bouldin scores, asterisk symbols denote the position on global or local minima. Finally, the second last and last rows show the combined results obtained with the QdetectA and QdetectB algorithms*

Validation index	Clustering, distance	k = 6	7	8	9	10
Hc(k)	PAM, cos	2.28**	1.22	2.18	2.15	0.32
	PAM, Euc	0.79	−0.21	0.75	0.92*	0.13
	havg, cos	6.58**	1.84	1.81	0.67	0.09
	havg, Euc	0.04	0.15	0.01	2.99***	0.06
Sil(k)	PAM, cos	0.195*	0.191	0.205	0.215	0.216
	PAM, Euc	0.102	0.091	0.093	0.104	0.110
	havg, cos	0.145	0.134	0.158	0.173	0.172
	havg, Euc	0.090	0.092	0.092	0.135**	0.134
Gap'(k)	PAM cos	−1.7e-02	−3.5e-02	−3.5e-02	−6.4e-04	−2.8e-02
	PAM euc	0.007***	−0.010	−0.014	0.001	0.0007
	havg, cos	−0.028	−0.027	−0.004	0.006**	−0.113
	havg, Euc	0.001	0.004**	−0.051	0.002	−0.005
$\gamma(k)$	PAM, cos	0.650	0.581	0.525	0.501	0.458
	PAM, Euc	0.645	0.640	0.596	0.570	0.546
	havg, cos	0.692	0.619	0.617	0.532	0.531
	havg, Euc	0.645	0.602	0.602	0.601	0.600
dunn(k)	PAM, cos	3.93e-06	4.51e-06**	1.68e-06	3.14e-06	3.14e-06
	PAM, Euc	0.010	0.012	0.019	0.009	0.010
	havg, cos	3.67e-05	5.86e-05	5.86e-05	2.85e-05	2.85e-05
	havg, Euc	0.022	0.036	0.036	0.036	0.036
QdetectA		3	0	0	9***	0
QdetectB		3	0	0	9***	0

Table 6.3. *Validation results of the NCI60 dataset*

validation scores with respect to the clustering conditions can be also observed. Some validation indices (Dunn, Krzanowski and Lai, Hubert, and gap statistic) are in some situations unable to detect the true number of clusters as a global maximum. These errors may be associated with the use of the cosine distance function, which, therefore, proves unsuitable for this dataset. However, the Krzanowski and Lai test and the gap statistic indices are still able to identify the optimum even with cosine distances.

Validation index	clustering, distance	k				
		2	3	4	5	6
Hc(k)	PAM, cos	184.397***	143.755	65.769	9.514	18.967
	PAM, Euc	−53.932	39.640***	13.550	12.844	1.497
	havg, cos	175.052***	143.247	4.289	74.159**	30.881
	havg, Euc	−53.932	6.979	1.017	27.146**	3.909
Dunn(k)	PAM, cos	0.289	0.363	0.435**	0.401	0.161
	PAM, Euc	0.479	0.468	0.515	0.525	0.525
	havg, cos	0.430	0.430	0.430	0.430	0.426
	havg, Euc	0.565	0.553	0.486	0.486	0.486
$\gamma(k)$	PAM, cos	0.384	0.464	0.517**	0.481	0.473
	PAM, Euc	0.298	0.319	0.381	0.430*	0.401
	havg, cos	0.417	0.472	0.474	0.483	0.558
	havg, Euc	0.482	0.488	0.555	0.566	0.568
KL(k)	PAM, cos	67.827***	0.067	5.409	1.322	0.137
	PAM, Euc	5.030	4.090	0.714	5.853*	0.263
	havg, cos	4.863*	2.952	13.340***	0.026	2.896
	havg, Euc	11.051	0.691	0.513	14.283	0.085
DB(k)	PAM, cos	0.621	0.878	1.230	1.189	1.210
	PAM, Euc	0.672	0.973	1.133	1.104	1.204
	havg, cos	0.617***	0.902	0.911	1.080	1.268
	havg, Euc	0.672	0.625	0.547***	0.680	0.677**
Sil(k)	PAM, cos	0.737***	0.669	0.646	0.559	0.553
	PAM, Euc	0.581***	0.447	0.386	0.335	0.321
	havg, cos	0.729	0.721	0.662	0.641	0.625
	havg, Euc	0.581***	0.480	0.406	0.374	0.324
Gap'(k)	PAM, cos	−0.420	−0.149	0.097**	0.025	−0.112
	PAM, Euc	−0.121	−0.004	0.003	0.059***	−0.024
	havg, cos	−0.430	0.101**	−0.1643	0.0008	0.176***
	havg, Euc	0.047	0.109***	−0.093	0.063	−0.102
QdetectA		2	7**	1	6	0
QdetectB		3	8***	1	0	0

Table 6.4. *Validation results of the Iris dataset*

The *QdetectA* algorithm produces a global maximum at the true number of clusters. However, it should be noted that another global maximum peak is wrongly placed at $k = 38$. Finally, the second variant (*QdetectB*) places a unique

maximum at $k = 5$ and is therefore able to discriminate the optimum number of clusters, $k_{\text{opt}} = 5$.

6.5.2. *Validation results of the mixture of seven Gaussians*

The seven Gaussians mixture can be also observed as a hierarchy of three well separable groups and seven less separable clusters.

The high overlapping among classes in this dataset misleads the validity indices, as can be observed in Table 6.2. Some of the validity curves place the global maxima at $k = 3$, thus identifying the three bigger groups (upper hierarchy level). However, none of the analyzed curves has been able to place a global maximum at the true number of Gaussians ($k_{\text{opt}} = 7$). Furthermore, only 5 validation curves of the 28 curves shown in Table 6.2 have achieved second local maxima at k_{opt}.

Regarding the combination approach, the first proposed algorithm (*QdetectA*) also fails to detect the number of Gaussians as a global optimum. However, it is able to place a second local maximum at k_{opt}, coinciding with the best validation curves. Finally, in contrast to the previous results, the *QdetectB* approach has correctly detected k_{opt} as a global maximum.

6.5.3. *Validation results of the NCI60 dataset*

Table 6.3 shows some results obtained with the NCI cancer dataset. Note that validation curves obtained with the Davies Bouldin and Krzanowski and Lai metrics have not been included. This is due to the existence of missing values in the dataset, which yields missing values in the Davies Bouldin and Krzanowski and Lai functions used.

In this dataset, only the (corrected) Hartigan index is able to detect the number of classes in one of the validation curves. This occurs when hierarchical average clustering is used in combination with the Euclidean distance. Other three validation curves place local maxima at $k_{\text{opt}} = 9$. Second local maxima are achieved at k_{opt} by the silhouette and gap metrics with hierarchical average clustering, and a third local maximum is placed by the Hartigan in combination with the PAM algorithm and Euclidean distance.

Regarding our combination approach, both algorithms, *QdetectA* and *QdetectB*, have discovered the correct number of classes at k_{opt}.

6.5.4. *Validation results of the Iris dataset*

Table 6.4 shows the validation results with the Iris dataset. This dataset is composed of two linearly separable Iris types and one third Iris class nonlinearly separable from the other two. Therefore, most validation curves fail to detect the number of clusters. Validation maxima are often misplaced at $k = 2$. Only the Hartigan and gap indices, in combination with the Euclidean distances and the PAM and hierarchical average algorithms, respectively, are able to identify the correct number of classes. Also, the gap statistic with the hierarchical average clustering and the cosine distance places a second maxima at $k = 3$.

These validation outcomes have been slightly improved by the *QdetectB* algorithm. Using this method, a second local maximum is placed at $k_{\text{opt}} = 3$. The global maximum is found at $k = 24$, and other local maxima are located at $k = 26$ and 32. Finally, as with the previous datasets, the *QdetectB* variant is able to identify the true number of clusters in the Iris dataset.

6.5.5. *Discussion*

The mixture of seven Gaussians can be divided in a hierarchy of three well separable clusters and seven, less separable clusters (optimum). In this situation, the evaluated validation indices generate numerous errors: the three top clusters are frequently identified instead of the seven existing Gaussians. Only the Hartigan, silhouette, and gap statistic find some local maxima at the optimum. However, despite the numerous errors, the combination approach is still able to identify the seven Gaussians if using the *QdetectB* algorithm. The first alternative (Qdetect) places a second local maximum at k_{opt}, coinciding with the best validation curves.

Validation errors can again be observed in individual validation curves on the NCI60 and Iris datasets, for which global and local maxima are barely placed at $k_{\text{opt}} = 9$ and $k_{\text{opt}} = 3$. In these datasets, combined results clearly outperform individual validation results. On the NCI60 dataset, both algorithms (*QdetectA* and *QdetectB*) place the global maxima at $k_{\text{opt}} = 9$. On the Iris dataset, *QdetectB* achieves a global maximum at $k_{\text{opt}} = 3$ versus a second local maximum obtained by *QdetectA*.

Furthermore, the observed results allow a comparative evaluation of validation indices. In general, the Hartigan, silhouette, and gap statistic indices have provided better performances than the Dunn, Hubert's γ and KL indices, although their performances clearly depend on the clustering settings (clustering algorithm and distances).

To summarize, it can be concluded that the $QdetectA$ algorithm performs equal or better than the best of the validity curves, in terms of the maximum peaks achieved at k_{opt}, while the second variant, $QdetectB$, has been able to detect the optimum number of cluster on all analyzed datasets. In the next section, the performances of both

validity combination approaches on a corpus of SLDS speech utterances are analyzed.

6.6. Application of speech utterances

The validation approaches to detect the number of clusters have been also applied to a corpus of speech utterances in the framework of automated troubleshooting agents.

As outlined in Chapter 1, a set of problem categories in a troubleshooting domain is currently defined by the dialog interaction designers. However, it is desirable to develop tools that aim to automatically redefine the set of problem categories with minimum human supervision. Such tools can enable a rapid adaptation of the systems and their portability to different domain data.

Therefore, two combination approaches (*QdetectA* and *QdetectB*) have also been applied for detecting the number of potential problems in a given troubleshooting domain, which provided a corpus of transcribed utterances related to this domain. In particular, the utterance corpus used in this work is related to the video troubleshooting domain and has been collected from user calls to commercial video troubleshooting agents. The corpus comprises 10,000 transcribed training utterances. Reference topic categories (symptoms), defined by the agent interaction designers, are also available. The number of reference topic categories is $k = 79$. Some examples of utterances and their associated reference categories are

– "*Remote's not working*" (CABLE)

– "*Internet was supposed to be scheduled at my home today*" (APPOINTMENT)

– "*I´m having Internet problems*" (INTERNET)

6.7. Summary

In this chapter, different approaches for the discovery of the true number of clusters in a dataset have been analyzed. First, seven existent approaches in the literature to the optimum detection have been introduced: the Hartigan, Dunn, Hubert's γ statistic, Davies Bouldin, gap, and silhouette width indices. The weakness of individual validity indices is related to the fact that the methods have been proposed for solving somewhat ad hoc problems or datasets and, therefore, their performance depends on the clustering algorithm, distance metric, or even on the dataset on which the validation indices are applied. This dependency has been evidenced through an analysis of the scores obtained by the different indices in synthetic and real datasets using four different clustering algorithms and two classical distance metrics (Euclidean and cosine).

Two approaches to the discovery of the number of clusters have been proposed to reduce the "ad hoc" performances of individual metrics. The main assumption was that, although the optimum could not be clearly visible in all individual metrics, the multiplicity of existing methods can be exploited to detect underlying "agreements" between the scores. In other words, the optimum (k_{opt}) is hidden among the top scores in the validation curves and can be uncovered by adequately measuring the consensus between the different validation outcomes. The proposed approach is to measure this agreement by calculating p quantiles (points among the $1 - p$ "top scores"). The quantile approach has proven highly adequate for discovering the hidden optimum in most datasets.

However, the optimum detection on the video troubleshooting agent has proven a task of high complexity, possibly due to the broad range of k values analyzed for the optimum search. The optimum ($k = 79$) could not be identified

either by the validation curves or by the quantile approach due to a rapid variation of the obtained scores with k, as well as the strong monotony of the curves with k. The *QdetectA* produced six maxima at $k = \{6, 18, 72, 89, 102, 126\}$, while the *QdetectB* algorithm failed to detect the optimum due to an aggressive exclusion of k values using $p = 0.9$ quantiles.

In parallel to the number of clusters detected, our future steps are toward the identification of the best fitting clustering algorithm and distance model, given a fixed number of clusters. A similar approach to the one developed in this work based on p quantiles can be used for this aim, applied to the set of validation curves obtained with each pair (clustering technique and distance function) with the set of validity indices and comparing these scores with the "global" scores obtained for detecting the optimum number of clusters. Another alternative is the application of cluster ensembles to reach a good compromise solution between the different clustering algorithms [STR 02].

Bibliography

[ACO 07] ACOMB K., BLOOM J., DAYANIDHI K., HUNTER P., KROGH P., LEVIN E., PIERACCINI R., "Technical Support Dialog Systems: Issues, Problems, and Solutions", *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, Rochester, USA, 2007.

[ALD 64] ALDENDERFER M., BLASHFIELD R., *Cluster Analysis*, Sage, Newbury Park, CA, 1964.

[ARI 06] ARIFIN A., ASANO A., "Image segmentation by histogram thresholding using hierarchical cluster analysis", *Pattern Recognition Letters*, vol. 27 no. 13, p. 1515-1521, 2006.

[ART 06] ARTHUR D., VASSILVITSKII S., "How slow is the *k*-mean method", *Proceedings the 22nd Annual ACM Symposium on Computational Geometry (SOCG 06)*, Sedona, Arizona, 2006.

[BAG 06] BAGIROV A. M., MARDANEH K., "Modified global *k*-means algorithm for clustering in gene expression data sets", *Proceedings of the 2006 Workshop on Intelligent Systems for Bioinformatics (WISB '06)*, Darlinghurst, Australia, p. 23-28, 2006.

[BAR 00] BARDIA S. H.-P., HAR-PELED S., SADRI B., *On Lloyd's k-means Method*, 2000.

[BEA 97] BEAULIEU M. M., GATFORD M., HUANG X., ROBERTSON S. E., WALKER S., WILLIAMS P., "Okapi at TREC-5", *Fifth Text Retrieval Conference (T-REC)*, MD, USA, NIST Special publication, 1997.

[BEL 98] BELZEK J. C., PAL N. R., "Some new indexes of cluster validity", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, p. 301-315, 1998.

[BLA 55] BLASHFIELD R. K., "The growth of cluster analysis: Tryon, Ward, and Johnson", *Multi-variate Behavioral Research*, vol. 15 no. 4, p. 439-458, 1955.

[BLA 07] BLANCHARD A., "Understanding and customizing stopword lists for enhanced patent mapping", *World Patent Information*, vol. 29 no. 4, p. 308-316, 2007.

[BLU 01] BLUM A., CHAWLA S., "Learning from labeled and unlabeled data using graph mincuts", *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, San Francisco, CA, p. 19-26, 2001.

[BOL 99] BOLEY D., GINI M., GROSS R., HAN E.-H., KARYPIS G., KUMAR V., MOBASHER B., MOORE J., HASTINGS K., "Partitioning-based clustering for web document categorization", *Decision Support System*, vol. 27 no. 3, p. 329-341, Elsevier Science Publishers B.V., 1999.

[BRO 99a] BROWN P. O., BOTSTEIN D., "Exploring the new world of the genome with DNA microarrays", *Nature Genetics*, vol. 21 no. 1, p. 33-37, Nature Publishing Group, 1999.

[BRO 99b] BROWN P.O., BOTSTEIN D., "Nature genetics", *Exploring the New World of the Genome with DNA Microarrays*, Nature Publishing Group, vol. 21 no. 1, p. 33-37, January 1999.

[BUR 98] BURGES C. J. C., "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, vol. 2, p. 121-167, 1998.

[BUZ 80] BUZO A., GRAY A. H., GRAY R. M., MARKEL J. D., "Speech coding based upon vector quantization", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28 no. 5, p. 562-574, 1980.

[CAR 02] CARSON C., BELONGIE S., GREENSPAN H., MALIK J., "Blobworld: image segmentation using expectation–maximization and its application to image querying", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, p. 1026-1038, 2002.

[CAS 95] CASTELLI V., COVER T. M., "On the exponential value of labeled samples", *Pattern Recognition Letters*, vol. 16 no. 1, p. 105-111, 1995.

[CHE 00] CHENG Y., CHURCH G. M., "Biclustering of expression data", *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, La Jolla, California, USA, p. 93-103, 2000.

[CHE 06] CHENG-YUAN TANG Y.-L. W., LEE Y.-C., "Cluster and clustering algorithm validity in image retrieval", *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2006, p. 3318-3323, 2006.

[CHE 09] JUN CHEN C., ZHAO ZHAN Y., JUN WEN C., "Hierarchical face recognition based on SVDD and SVM", *ESIAT (2)*, IEEE Computer Society, p. 692-695, 2009.

[CHU 99] CHU-CARROLL J., CARPENTER B., "Vector-based natural language call routing", *Computational Linguistics*, vol. 25, p. 361-388, 1999.

[CHU 02] CHURCHILL G. A., "Fundamentals of experimental design for cDNA microarrays", *Nature Genetics*, vol. 32 Suppl., p. 490-495, 2002.

[CLA 95] CLARKE L.P., VELTHUIZEN R.P., CAMACHO M.A., HEINE J.J., VAIDYANATHAN M., HALL L.O., THATCHER R.W., SILBIGER M.L., "MRI segmentation: methods and applications", *Magnetic Resonance Imaging*, vol. 13 no. 3, p. 343-368, 1995.

[CLE 04a] CLEUZIOU G., MARTIN L., VRAIN C., "PoBOC: un algorithme de 'soft-clustering'. Applications a l'apprentissage de regles et au traitement de donnees textuelles", vol. RNTI-E-2 of *Revue des Nouvelles Technologies de l'Information*, Cepadues-Editions, p. 217-228, 2004.

[CLE 04b] CLEUZIOU G., MARTIN L., VRAIN C., "PoBOC: an overlapping clustering algorithm, application to rule-based classification and textual data", *Proceedings of ECAI*, Valencia, Spain, p. 440-444, 2004.

[COT 00] COTTRELL M., HAMMER B., HASENFUSS E., VILLMANN T., "Batch neural gas", *WSOM 2005. Fifth International Wokshop on self organising maps*, Paris, France, 2000.

[COX 61] COX D., "Tests of separate families of hypotheses", *Proceedigs of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, California, USA, p. 105-123, 1961.

[COX 62] COX D., "Further results on tests of separate families of hypotheses", *Journal of the Royal Statistical Society*, vol. 24, p. 406-423, 1962.

[CUT 92] CUTTING D. R., PEDERSEN J. O., KARGER D., TUKEY J. W., "Scatter/gather: a cluster-based approach to browsing large document collections", *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, p. 318-329, 1992.

[DAR 02] DARA R., STACEY D., "Clustering unlabeled data with SOMs improves classification of labeled real-world data", *Proceedings of the Ninth International Conference on Fuzzy Systems*, Honolulu Hawai, USA, 2002.

[DAV 79] DAVIES D., BOULDIN D., "A cluster separation measure", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, p. 224-227, 1979.

[DAV 91] DAVIS L. D., MITCHELL M., "Handbook of Genetic Algorithms", *Van Nostrand Reinhold*, 1991.

[DEE 90] DEERWESTER S., DUMAIS S. T., FURNAS G. W., LANDAUER T. K., HARSHMAN R., "Indexing by latent semantic analysis", *Journal of the American Society for Information Science*, vol. 41 no. 6, p. 391-407, 1990.

[DEM 77] DEMPSTER A. P., LAIRD N. M., RUBIN D. B., "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society, Series B*, vol. 39 no. 1, p. 1-38, 1977.

[DEM 99] DEMIRIZ A., BENNETT K., EMBRECHTS M. J., "Semi-supervised clustering using genetic algorithms", *Artificial Neural Networks in Engineering (ANNIE-99)*, p. 809-814, 1999.

[DEP 00] DEPARTMENT Y. K., KO Y., SEO J., "Automatic text categorization by unsupervised learning", *Proceedings of COLING-2000*, p. 453-459, 2000.

[DES 00] DESIGN I. T., GABRYS B., PETRAKIEVA L., "Combining labelled and unlabelled data", *International Journal on Approximate Reasoning*, vol. 35, p. 251-273, 2004.

[DES 09] DESHMUKH K. S., "Color image segmentation using fuzzy *c*-means clustering", *IICAI: International Indian Conference on Artificial Intelligence*, p. 1802-1813, 2009.

[DRA 09] DRAGUT E., FANG F., SISTLA P., YU C., "Stop word and related problems in web interface Integration", *Proceedings of the VLDB Endowment*, vol. 2 no. 1, p. 349-360, 2009.

[DUN 74] DUNN J., "Well separated clusters and optimal fuzzy partitions", *Journal of Cybernetics*, vol. 4, p. 95-104, 1974.

[EIS 98] EISEN M. B., SPELLMAN P. T., BROWN P. O., BOTSTEIN D., "Cluster analysis and display of genome-wide expression patterns", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 95 no. 25, p. 14863-14868, 1998.

[EQU 89] EQUITZ W. H., "A new vector quantization clustering algorithm", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37 no. 10, p. 1568-1575, 1989.

[EST 96] ESTER M., KRIEGEL H.-P., SANDER J., XU X., "A density-based algorithm for discovering clusters in large spatial databases with noise", *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, KDD-96, p. 226-231, August 1996.

[FEN 09] FENG W., XIE L., LIU Z.-Q., "Multicue graph mincut for image segmentation", *ACCV (2)*, vol. 5995 of *Lecture Notes in Computer Science*, Springer, p. 707-717, 2009.

[FIS 58] FISHER, W. D., "On Grouping for Maximum Homogeneity", *Journal of the American Statistical Association*, vol. 53, p. 789-798, 1958.

[FOR 03] FORMAN G., "An extensive empirical study of feature selection metrics for text classification", *Journal of Machine Learning Research*, vol. 3, p. 1289-1305, MIT Press, 2003.

[FRE 81] FREEDMAN D., DIACONIS P., "On the histogram as a density estimator: L2 theory", *Probability Theory and Related Fields*, vol. 57 no. 4, p. 453-476, 1981.

[FRI 95] FRITZKE B., "A growing neural gas network learns topologies", *Advances in Neural Information Processing Systems 7*, MIT Press, p. 625-632, 1995.

[FRO 00] FROHNE, H., Sample quantiles, Research Project, 2000.

[FUH 89] FUHR N., "Models for retrieval with probabilistic indexing", *Information Processing and Management*, p. 55-72, 1989.

[FYF 06] FYFE C., "The topographic neural gas", *Proceedings of the 7th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL06*, Burgos, Spain, p. 241-249, 2006.

[GER 91] GERSHO A., GRAY R. M., *Vector Quantization and Signal Compression*, Kluwer Academic Press, 1991.

[GET 00] GETZ G., LEVINE E., DOMANY E., "Coupled two-way clustering analysis of gene microarray data", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 97, p. 12079-12084, 2000.

[GOL 89] GOLDBERG D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, 1989.

[GOL 03] GOLIN M. J., *Bipartite matching and the Hungarian method*, http://www.cs.ust.hk/~golin/COMP572/Notes/Matching.pdf, 2003.

[GRA 05] GRAHAM J. R., *MMPI-2: Assessing Personality and Psychopathology*, Oxford University Press, 2005.

[GUY 03] GUYON I., "An introduction to variable and feature selection", *Journal of Machine Learning Research*, vol. 3, p. 1157-1182, 2003.

[HAL 00] HALKIDI M., VAZIRGIANNIS M., BATISTAKIS Y., "Quality scheme assessment in the clustering process", *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, London, Springer-Verlag, p. 265-276, 2000.

[HAL 02a] HALKIDI M., BATISTAKIS Y., VAZIRGIANNIS M., "Cluster validity methods: part I", *ACM SIGMOD Record*, vol. 31, p. 2002, 2002.

[HAL 02b] HALKIDI M., BATISTAKIS Y., VAZIRGIANNIS M., "Clustering validity checking methods: part II", *SIGMOD Record*, vol. 31 no. 3, p. 19-27, ACM, 2002.

[HAR 75] HARTIGAN J., *Clustering Algorithms*, Wiley, New York, NY, 1975.

[HAR 05] HARPELED S., SADRI B., "How fast is the k-means method", *Algorithmica*, vol. 41, p. 185-202, 2005.

[HAS 00] HASTIE T., TIBSHIRANI R., EISEN M., ALIZADEH A., LEVY R., STAUDT L., CHAN W., BOTSTEIN D., BROWN P., " 'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns", *Genome Biology*, vol. 1 no. 2, p. 1-21, 2000.

[HAS 09] HASTIE T., TIBSHIRANI R., FRIEDMAN J., *Hierarchical Clustering. The Elements of Statistical Learning*, Springer, New York, NY, 2009.

[HAV 08] HAVENS T., BEZDEK J., KELLER J., POPESCU M., "Dunn's cluster validity index as a contrast measure of VAT images", *International Conference on Pattern Recognition (ICPR)*, Tampa Convention Center, Tampa, Florida, USA, p. 1-4, December 2008.

[HEA 96] HEARST M. A., PEDERSEN J. O., "Reexamining the cluster hypothesis: scatter/gather on retrieval results", *Proceedings of the 19th Annual International ACM/SIGIR Conference*, Zurich, Switzerland, p. 76-84, 1996.

[JAI 99] JAIN A. K., MURTY M. N., FLYNN P. J., "Data clustering: a review", *ACM Computing Surveys*, vol. 31 no. 3, p. 264, 1999.

[JAI 04] JAIN A. K., "Lanscape of clustering algorithms", *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, 2004.

[JOA 97] JOACHIMS T., INFORMATIK F., INFORMATIK F., INFORMATIK F., INFORMATIK F., VIII L., "Text categorization with support vector machines: learning with many relevant features", *Proceedings of the First European Conference on Machine Learning (ECML 98)*, p. 137-142, 1998.

[JOL 89] JOLION J.-M., ROSENFELD A., "Cluster detection in background noise", *Pattern Recognition*, vol. 22 no. 5, p. 603-607, Elsevier Science, Inc., 1989.

[JON 07] JONES K. S., "Information retrieval and digital libraries: lessons of research", *Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries (IWRIDL '06)*, Kolkata, India, p. 1-7, 2007.

[KAR 98] KARYPIS G., KUMAR V., METIS: a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, University of Minnesota, USA, 1998.

[KLE 00] KLEIN S. T., "On the use of negation in Boolean IR queries", *Information Processing and Management*, vol. 45, no. 2, p. 291-311, 2000.

[KOH 90] KOHONEN T., "The self-organizing map", *Proceedings of the IEEE*, vol. 79 no. 9, 1990.

[KOH 01] KOHONEN T., SCHROEDER M. R., HUANG T. S., Eds., *Self-Organizing Maps*, Springer-Verlag New York, Inc., Secaucus, NJ, 2001.

[KRZ 85] KRZANOWSKI W., LAI Y., "A criterion for determining the number of groups in a data set using sum of squares clustering", *Biometrics*, vol. 44, p. 23-34, 1985.

[KUH 55] KUHN H. W., "The Hungarian method for the assignment problem", *Naval Research Logistics Quarterly*, vol. 2, p. 83-97, 1955.

[LAN 98] LANDAUER T. K., FOLTZ P. W., LAHAM D., "An introduction to latent semantic analysis", *Discourse Processes*, no. 25, p. 259-284, 1998.

[LAS 09] LASHKARI A. H., MAHDAVI F., GHOMI V., "A Boolean model in information retrieval for search engines", *Information Management and Engineering, International Conference on ICIME 2009*, Kuala Lumpur, Malaysia, p. 385-389, IEEE Computer Society, 2009.

[LEO 05] LEONARD KAUFMAN P. J. R., *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Series in Probability and Statistics, New York, NY, 2005.

[LI 98] LI Y. H., JAIN A. K., "Classification of text documents", *The Computer Journal*, vol. 41, p. 537-546, 1998.

[LIN 06] LIN Y.-M., WANG X., NG W., CHANG Q., YEUNG D., WANG X.-L., "Sphere classification for ambiguous data", *Machine Learning and Cybernetics, 2006 International Conference on ICMLC*, Dalian, China, p. 2571-2574, 2006.

[LO 07] LO E. H. S., PICKERING M. R., FRATER M. R., ARNOLD J. F., "Image segmentation using invariant texture features from the double dyadic dual-tree complex wavelet transform", *Proceeding of ICASSP*, IEEE, Honolulu, Hawai'i USA, p. 609-612, 2007.

[MAC 08] MACHLACHLAN G., KRISHNAN T., *The EM Algorithm and Extensions*, Wiley Series in Probability and Statistics, 2008.

[MAE 04] MAEIREIZO B., LITMAN D., HWA R., "Co-training for predicting emotions with spoken dialogue data", *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, Companion Volume to the proceeding of 42nd Annual Meeting of the Association for Computational Linguistics (ACL), July 2004, Barcelona, Spain.

[MAN 00] MANCAS M., GOSSELIN B., MACQ B., "Segmentation using a region-growing thresholding", *Image Processing: Algorithms and Systems*, Proceedings of the SPIE, p. 388-398, 2000.

[MAN 08] MANNING C. D., RAGHAVAN P., SCHÜTZE H., *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[MAR 60] MARON M. E., KUHNS J. L., "On relevance, probabilistic indexing and information retrieval", *Journal of the ACM*, vol. 7 no. 3, p. 216-244, ACM, 1960.

[MIN 01] MINNEN G., CARROL J., PEARCE D., "Applied morphological processing of English", *Natural Language Engineering*, vol. 7 no. 3, 2001.

[MUK 08] MUKHOPADHYAY A., BANDYOPADHYAY S., MAULIK U., "Combining multiobjective fuzzy clustering and probabilistic ANN classifier for unsupervised pattern classification: application to satellite image segmentation", *IEEE Congress on Evolutionary Computation*, IEEE, p. 877-883, 2008.

[MUR 07] MURRAY G., RENALS S., "Towards online speech summarization", *Proceedings of the Interspeech '07*, Antwert, Belgium, 2007.

[MUR 08] MURRAY G., RENALS S., "Term-weighting for summarization of multi-party spoken dialogues", *Proceedings of the 4th International Conference on Machine Learning for Multimodal Interaction*, Berlin, p. 156-167, 2008.

[NA 09] NA S.-H., NG H. T., "A 2-Poisson model for probabilistic coreference of named entities for improved text retrieval", *SIGIR '09: Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, p. 275-282, 2009.

[NCI 06] NCI Cancer Microarray Data (Stanford University), 2006, http://genome-www.stanford.eud/nci60.

[NIG 99] NIGAM K., MCCALLUM A. K., THRUN S., MITCHELL T., "Text classification from labeled and unlabeled documents using EM", *Machine Learning*, May 2000, vol. 39, no. 2, p. 103-134, 1999.

[NIG 00] NIGAM K., MCCALLUM A. K., THRUN S., MITCHELL T., "Text classification from labeled and unlabeled documents using EM", *International Journal of Machine Learning*, vol. 39 no. 2-3, p. 103-134, 2000.

[OUY 09] OUYANG C.-S., CHOU C.-T., JHAN C.-F., HUANG J.-Y., "An improved approach for image segmentation based on color and local homogeneity features", *ICASSP*, IEEE, Taipei, Taiwan, p. 1225-1228, 2009.

[PAR 07] PARK J.-A., KANG S. K., JEONG I., RASHEED W., PARK S., AN Y., "Web based image retrieval system using HSI color indexes", *ICIC (3)*, vol. 2 of *Communications in Computer and Information Science*, Springer, p. 199-207, 2007.

[PIC 99] PICARD J., "Finding content-bearing terms using term similarities", *Proceedings of Ninth Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway, p. 241-244, June 1999.

[POR 80] PORTER M. F., "An algorithm for suffix stripping", *Program*, vol. 14 no. 3, p. 130-137, 1980.

[POR 96] R. PORTER, CANAGARAJAH N., "A robust automatic clustering scheme for image segmentation using wavelets", *IEEE Transactions on Image Processing*, vol. 5 no. 4, p. 662-665, 1996.

[PRI 03] PRIEGO J. L. O., "A vector space model as a methodological approach to the triple helix dimensionality: a comparative study of Biology and Biomedicine Centres of two European National Research Councils from a webometric view", *Scientometrics*, p. 429-443, 2003.

[RAB 93] RABINER L., JUANG B. H., *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[RAM 00] RAMOS J., "Using TF-IDF to determine word relevance in document queries", http://www.cs.rutgers.edu/mlitmann/courses/ml03/ICML03/papers/ramos.pdf, 2000.

[RIJ 79] VAN RIJSBERGEN C. J., *Information Retrieval*, Butterworths, London, 1979.

[ROB 92] ROBERTSON S. E., WALKER S., HANCOCK-BEAULIEU M., GULL A., LAU M., "Okapi at TREC", *Text Retrieval Conference*, p. 21-30, 1992.

[ROB 99] ROBERTSON S., WALKER S., BEAULIEU M., WILLETT P., "Okapi at TREC-7: automatic *ad hoc*, filtering, VLC and interactive track", , *Seventh Text Retrieval Conference (TREC-7) Gaithersburg*, Maryland, USA, vol. 21, p. 253-264, 1999.

[ROB 00] ROBERTSON S. E., WALKER S., BEAULIEU M., "Experimentation as a way of life: Okapi at TREC", *Information Processing and Management*, January 96, vol. 36, no. 1 p. 95-108, 2000.

[ROB 01] ROBERT TIBSHIRANI G. W., HASTIE T., "Estimating the number of clusters in a dataset via the gap statistic", *Journal of the Royal Statistical Society*, vol. 63, p. 411-423, 2001.

[ROE 00] ROELLEKE T., WANG J., "Binary independence retrieval model, probabilistic relational modelling, integration of database and information retrieval (DB+IR)", *Probabilistic Logical Modeling of the Binary Independence Retrieval Model*. *In Proceedings of the First International Conference of Information Retrieval (ICTIR 07) – Studies in Theory of Information Retrieval*, 2000.

[ROS 00] ROSS D. T., SCHERF U., EISEN M. B., PEROU C. M., REES C., SPELLMAN P., IYER V., JEFFREY S. S., DE RIJN M. V., WALTHAM M., PERGAMENSCHIKOV A., LEE J. C., LASHKARI D., SHALON D., MYERS T. G., WEINSTEIN J. N., BOTSTEIN D., BROWN1 P. O., "Systematic variation in gene expression pattenrns in human cancer cell lines", *Nature Genetics*, vol. 24 no. 3, p. 227-235, 2000.

[ROU 87] ROUSSEEUW P., "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis", *Journal of Computational and Applied Mathematics*, vol. 20, p. 53-65, 1987.

[SAL 71] SALTON G., *The SMART Retrieval System – Experiments in Automatic Document Processing*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1971.

[SAL 75] SALTON G., WONG A., YANG C.-S., "A vector space model for automatic indexing", *Communications of the ACM*, vol. 18 no. 11, p. 613-620, 1975.

[SAL 88] SALTON G., BUCKLEY C., "Term-weighting approaches in automatic text retrieval", *Information Processing and Management*, vol. 4, no. 5, p. 513-524, 1988.

[SAN 98] SANDER J., ESTER M., KRIEGEL H.-P., XU X., "Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications", *Data Mining and Knowledge Discovery*, vol. 2 no. 2, p. 169-194, 1998.

[SCH 94] SCHMID H., "Probabilistic part-of-speech tagging using decision trees", *Proceedings of the International Conference on New Methods in Language Processing*, Umist, Manchester, UK, p. 44-49, 1994.

[SEE 01] SEEGER M., Learning with labeled and unlabeled data, Report, 2001.

[SER 80] SERFLING R., *Approximation Theorems of Mathematical Statistics*, John Wiley and Sons, 1980.

[SHE 96] SHEPPARD A. G., "The sequence of factor analysis and cluster analysis: differences in segmentation and dimensionality through the use of raw and factor scores", *Tourism Analysis*, vol. 1, p. 49-57, 1996.

[SIN 96a] SINGHAL A., SALTON G., BUCKLEY C., "Length normalization in degraded text collections", *Proceedings of Fifth Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, Nevada, USA, p. 15-17, 1996.

[SIN 96b] SINGHAL A., SALTON G., MITRA M., BUCKLEY C., Document length normalization, Report, Ithaca, NY, 1996.

[SIN 09] SINGHAL A., BUCKLEY C., MITRA M., "Pivoted document length normalization", *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, p. 21-29, 1996.

[SOK 63] SOKAL R., SNEATH P., *Principles of Numerical Taxonomy*, Morgan Freeman, San Francisco, CA, 1963.

[STE 00] STEIN B., EISSEN S. M. Z., POTTHAST M., "Syntax versus semantics: analysis of enriched vector space models", 2000.

[STO 00] Stop word list of the Smart Information Retrieval Project, 2000, http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop.

[STR 02] STREHL A., GHOSH J., CARDIE C., "Cluster ensembles – a knowledge reuse framework for combining multiple partitions", *Journal of Machine Learning Research*, vol. 3, p. 583-617, 2002.

[TAM 99] TAMAYO P., SLONIM D., MESIROV J., ZHU Q., KITAREEWAN S., DMITROVSKY E., LANDER E. S., GOLUB T. R., "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96 no. 6, p. 2907-2912, 1999.

[THE 09] THEWES R., "Introduction to electronic DNA microarrays", *EPFL Summer School on Nano-Bio-Sensing*, Sommer School, Lausanne, Switzerland, 2009.

[TOU 00] TOUTANOVA K., MANNING C. D., "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger", *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Morristown, NJ, Association for Computational Linguistics, p. 63-70, 2000.

[TRE 05] TREECK B., *Entwicklung und Evaluierung einer Java-Schnittstelle zur Clusteranalyse von Peer-to-Peer Netzwerken. Bachelorarbeit*, Heinrich Heine University, Duesseldorf, 2005.

[TRY 55] TRYON R., "Identification of social areas by cluster analysis", *University of California Publications in Psychology*, no. 8, p. 1-100, 1955.

[TRY 68] TRYON R., "Comparative cluster analysis of variables and individuals: Holzinger abilities and MMPI attributes", *Multivariate Behavioral Research*, vol. 3, p. 115-144, 1968.

[TRY 70] TRYON R., BAILEY D., *Cluster Analysis*, McGraw Hill, New York, NY, 1970.

[VES 00] VESANTO J., ALHONIEMI E., "Clustering of the self-organizing map", *IEEE Transactions on Neural Networks*, vol. 11 no. 3, p. 586-600, 2000.

[WAN 08] WANG Y., ZUO W., PENG T., HE F., HU H., "Clustering web search results based on interactive suffix tree algorithm", *Convergence Information Technology, International Conference on*, vol. 2, p. 851-857, IEEE Computer Society, 2008.

[WAR 63] WARD J., "Hierarchical grouping to optimize an objective function", *Journal of the American Statistical Association*, vol. 58 no. 301, p. 236-244, 1963.

[WOR 98] *WordNet An Electronic Lexical Database*, Cambridge, MA, London, 1998.

[WU 09] WU J., CHEN J., XIONG H., XIE M., "External validation measures for *K*-means clustering: a data distribution perspective", *Expert Systems with Applications*, vol. 36 no. 3, p. 6050-6061, 2009.

[WUL 97] WULFEKUHLER M. R., PUNCH W. F., "Finding salient features for personal web page categories", *Proceedings of 6th International World Wide Web Conference*, p. 6-118, 1997.

[YAN 97] YANG Y., PEDERSEN J. O., "A comparative study on feature selection in text categorization", *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Nashville, TN, p. 412-420, 1997.

[YAN 09] YANG X., CLAUSI D. A., *ICIP*, IEEE, p. 1721-1724, 2009.

[YAR 95] YAROWSKY D., "Unsupervised word sense disambiguation rivaling supervised methods", *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, 1995.

[YE 09] YE P., WENG G., "Microarray image segmentation using region growing algorithm and mathematical morphology", *IAS*, IEEE Computer Society, p. 373-376, 2009.

[ZAM 98] ZAMIR O., ETZIONI O., "Web document clustering: a feasibility demonstration", *Research and Development in Information Retrieval*, p. 46-54, 1998.

[ZHU 06] ZHU X., "Semi-supervised learning literature survey", http://pages.cs.wisc.edu/ jerryzhu/pub/ssl_survey.pdf, 2006.

Index

Semi-Supervised and Unsupervised Machine Learning: Novel Strategies Amparo Albalate and Wolfgang Minker