Text Mining and Applications in Life Sciences

Habilitation thesis
to obtain the academic degree
Dr. rer. nat. habil.

submitted to the
Faculty of Natural Sciences III
of the Martin-Luther-University Halle-Wittenberg

by
Dr. rer. nat. Alexander Hinneburg
born on February 28, 1975 in Halle/Saale, Germany

Reviewer:
1. Prof. Dr. rer. nat. Stefan Brass
2. Univ.-Prof. Dr. rer. nat. Thomas Seidl
3. Prof. Dr. Myra Spiliopoulou

Day of the defense: January 13th 2011

Text Mining and Applications in Life Sciences

Alexander Hinneburg

April 25, 2012

Contents

1 Introduction

Text is one of most prevalent forms of data that is directly accessible to humans. The availability of huge electronic document collections and the increasing growth rate of these databases created the demand for automatic processing of natural language in text form. The long list of technologies that were developed in response includes web search engines (information retrieval) and email spam filters (text mining) at the first ranking positions. The preprocessing of many concepts to handle text documents in automatic ways first breaks the text of each document into words, reduces the words to normal forms by heuristically stripping grammatical modifications, and last represents the documents as bags of words. Experiments demonstrated that the bag of words representation, which neglects all information about the order of the words in the original documents, is sufficient for satisfactory completion of most of the automated tasks [94]. Thus, we observe that most techniques for information retrieval and text mining see documents as sets of words that are accompanied only by word frequencies as additional information.

Text Mining, which in this work is mainly data mining and machine learning of text data, includes models for standard analyses such as classification and clustering and additionally a unique kind of models, namely topic models. Topic models belong to the class of unsupervised learning techniques that learn a function without the help of given examples of the form: function input , function output. Topic models learn a representation of the documents as mixtures of topics. A set of topics is learned during the training process using statical correlations between co-occurring words. A topic is a probability distribution over the vocabulary. Words that belong to the same topic are given high probability by the respective distribution. Therefore, topics are easily interpretable: In most cases one can get a rough idea of the semantic meaning of a topic by looking at the top words with the largest probabilities. The output of a topic model can serve as an overview about a document collection because of the ease of interpretation of the results. The example in table 1.1, which is reproduced from [74], shows the respective top ten words of four topics that are part of a topic model with 128 topics in total. The model is learned for the TDT1 benchmark collection[1], which consists of news stories from Reuters and CNN taken between 7/1994 and 6/1995. Topic models can reflect many properties of natural language, e.g. words that are used with several meanings (polysemy) as well as synonyms, which are different words used for the same meaning. Furthermore, the usefulness of document representations learned by topic models has been shown to improve retrieval quality of text search [74, 75] and text classifications [18].

[1] `http://projects.ldc.upenn.edu/TDT/`

Table 1.1: Top 10 words of four topics of a topic model (PLSA) with 128 topics trained on the TDT1 benchmark collection [74].

Topic 1	Topic 2	Topic 3	Topic 4
plane	space	home	film
airport	shuttle	family	movie
crash	mission	like	music
flight	astronauts	love	new
safety	launch	kids	best
aircraft	station	mother	hollywood
air	crew	life	love
passenger	nasa	happy	actor
board	satellite	friends	entertainment
airline	earth	cnn	star

Our initial motivation was to study whether and how statistical text mining methods could be transferred to applications in life sciences. The demands for such methods in life science have several reasons. First, the advent of high throughput technology for screening bio-assays produced large data volumes that reach the critical mass to allow more sophisticated statistical methods than simple variance analysis and confidence tests. Second, public collections of data from such experiments allowed the comparison between different experiments that lead to a kind of content based similarity search in those databases. Furthermore, new types of experiments became more widely used that output a whole set of inter-dependent measurements per experiment. Thus, the result of such an experiment could be seen as an abstract compound data object. Those data exhibit close analogies to the structure of the bag of words representation for documents. Last, the analogy to text data provides useful ways to show to non-experts in data analysis that such analysis methods work in principle and to explain the basic concepts behind the used models.

In this work, we study three different application domains in the life sciences, namely two-dimensional nuclear resonance (2D-NMR) spectography (chapter 2 and chapter 3.2), proteomics experiments (chapter 3.3) and protein precursor sequences that function as transit peptides for mitochondria (chapter 5.2). All these kinds of data can be transformed to a compound data representation that is analogous to the bag of words representation of documents. One general question is, whether the bag of words representation and the subsequently used text mining methods are applicable in these scenarios. The corresponding studies have the character of pilot analyses that help to decide what methods are suitable for the mentioned applications domains. Furthermore, the usefulness of the outcomes of the studies help to decide whether a large collecting of results of many biological experiments is an interesting investment of resources or not. Showing such a usefulness would provide an important stimulus for the efforts of collecting such experimental data in the life sciences.

The text mining methods we concentrate on are probabilistic latent semantic analysis and its applications to similarity search (chapter 3.2 and chapter 3.3), near-duplicate detection of documents based on locality sensitive hashing (chapter 2), and cluster analysis based on fractal dimension (chapter 5). Text Mining models base on probabilistic models. Our contribution to text mining is a new connection between kernel density estimation (KDE) and the expectation maximization (EM) algorithm. We introduce the new technique in a simple setting of cluster analysis (chapter 4.2) and demonstrate the applicability to text mining by using KDE as prior distribution for probabilistic latent semantic analysis (chapter 4.3). The clustering by fractal dimension (chapter 5) uses a distance or similarity matrix of a data set as input. The methods estimates for each data objects a local fractal dimension and a local density. Objects are assigned to the same cluster if they have similar local dimension and density. The method can be used to find subsets in the data that differ in the degrees of freedom.

1.1 Text Mining

In this section, we give a brief introduction into the relevant subjects of text mining. The term was coined in an article by Feldman and Dagan [49] presented at the First International Conference on Knowledge Discovery, 1995. It describes now interdisciplinary research with contributions from many fields including linguistics, information retrieval, databases, machine learning, statistics, and data mining. The subject is the automatic extraction of hidden knowledge from unstructured text document collections. In contrast to data mining, which works on structured data, text documents are much less structured. Therefore, the preprocessing steps bring the documents into the bag of words model, which represents each document as a high-dimensional vector that stores for each word in the vocabulary how often it occurs in the respective document. The heuristics used for that reduction are mainly developed in computational linguistics and involve the use of general and domain specific dictionaries as well as parsers that tag words with their grammatical functions.

The machine learning and data mining part of text mining is mainly concerned with the analysis of documents represented as bag of words. A typical text mining methodology is document clustering that analyzes the relations among documents. The goal is to group the total set of documents into clusters of similar documents. A special case of document clustering is near-document duplicate detection [21, 64]. A cluster is in this case a set of duplicate documents that differ only in minor parts. An application of this type of clustering is the detection of mirror web pages, which helps to improve web search by reporting duplicate content only once in the result list. The application of clustering at the scale of the web is only possible due to the use of approximation techniques such as locality sensitive hashing (LSH) [24].

A major problem that appears during the analysis of text is caused by the enormous flexibility of natural language. Well known phenomenons such as polysemy and synonyms are difficult to handle by algorithms. Polysemy denotes words with

ambiguous meanings, e.g. figure means the shape or stature of a person but it is also used to talk about digits or numbers. Synonyms are words with the same or similar meaning like car and automobile. Machine learning methods that can handle these phenomena are topic models like probabilistic latent semantic analysis PLSA [74], because they can learn correlations among words from word co-occurrences.

Mathematically, topic models have close connections to matrix factorizations like singular value decomposition (SVD) [59]. The idea is to concatenate the vector representations of documents of a whole collection to a term document matrix. SVD can be used to find a low rank approximation of such a matrix, which can be written as the product of two much smaller matrices. Because the word count vectors of documents can be normalized to describe multinomial distributions, probabilistic methods to approximate the term document matrix are more successful than SVD-based approximations [74].

Probabilistic topic models are mixture models that model the probability distribution over the vocabulary as a mixture of simple probability distributions such as multinomial distributions. Probabilistic topic models represent documents by document specific topic mixture proportions. Topics are multinomial distributions over the vocabulary. A word may have high probability in the distributions of several topics. This may happen, when it is just a frequently used word, however, this pattern is also typical for words exhibiting polysemy. Words that have high probability in a single topic distribution may be just frequently used together, however, synonyms are also modeled this way. Thus, words that are difficult to handle can be represented in a topic model in a natural way.

Topics in topic models do not correspond to clusters in probabilistic document clustering models. A document belongs to a cluster as a whole object even when the assignment is done probabilistically. In contrast, documents do not necessarily belong to a topic as a whole object but a subset of the words in a document may belong to a particular topic. Other subsets of words of the same document may belong to a second topic. This is reflected in the document specific topic mixture proportions. Therefore, topic models are more flexible than clustering models. The result of a clustering could occur as a special case in a topic model, namely when all documents have only a single dominant topic in the respective topic mixture proportions.

Probabilistic modeling is a mathematically rigorous framework that rely on a small set of rules for working with probabilities, which can be found in the appendix. The mathematical framework eases the transfer of probabilistic topic models to other application domains.

1.2 Applications in Life Sciences

We give a short overview about the application domains where we apply text mining models. We discuss the analogies to the bag of word representation of documents. Furthermore, we discuss how phenomena in natural language like polysemy and

synonyms could be related to the application domains.

1.2.1 2D-NMR-Spectography

We describe the basics of NMR spectrography according to [81]. Nuclear magnetic resonance spectography is a method to analyse organic molecules. Some atom nuclei have a small magnetic field called spin. Among those nuclei are the isotopes ^{1}H and ^{13}C. When placed in a magnetic field the atoms of a particular isotope type resonate at a characteristic frequency. However, when such an atom takes part in a chemical bond its resonate frequency is slightly distorted. That distortion can be measured and is called chemical shift. Chemical shifts are very small, e.g. the characteristic resonance frequency of ^{1}H is 42.58 MHz and the deviation may be 100 Hz. Therefore, the chemical shift is measured in parts per million (ppm). Typical chemical shifts for ^{1}H are in the range of 1 to 8 ppm and those of ^{13}C are from 10 to 200. Chemical shifts are characteristically distorted depending on the chemical environment inside the respective molecule, e.g. ^{1}H-NMR spectrum for ethanol (CH_3-CH_2-OH) has three shifts, one shift for CH_3, one for CH_2 and one for the OH group. When plotted these shifts constitute a one-dimensional NMR spectrum.

A series of one-dimensional NMR measurements with respect to ^{1}H and ^{13}C isotopes can be performed over time. Combining these measurements by Fourier transformation, correlations between the shifts of ^{1}H and ^{13}C can be found that correspond to chemical bonds between ^{1}H and ^{13}C atoms. The result is a specific type of a 2D-NMR spectrum.

A 2D-NMR-spectrum of a molecule consists of a set of two-dimensional chemical shifts, called peaks, which represent local parts of the respective molecule. Peaks are in this setting analogous to words in documents and the spectra take the place of documents. Organic molecules are complex structures, which are composed of elementary building blocks. A single chemical shift (peak) does no uniquely identify such a building block in general. Thus, the same peak could be produced in different chemical contexts. This is a kind of polysemy of chemical shifts (peaks) in 2D-NMR spectra.

Similarity search in large spectra databases would be a valuable tool for the elucidation of the chemical structure of an unknown molecule. Similar NMR-spectra with known chemical structure could give hints about the chemical family the unknown molecule might belong to.

1.2.2 Proteomics

A proteomics experiment aims at identifying all proteins that are present in a certain tissue sample. There are many different experimental approaches known to this end that differ in preprocessing steps to separate the proteins in the complex mixture obtained from the given sample. The last steps, however, are common practice. After separation, the intact protein strings are chopped at specific positions into small fragments called peptides. The mixture of peptides is analyzed by mass spectrom-

etry. As small peptides have unique characteristic masses, the individual peptides present in the mixture can be identified by this experiment. The original proteins are identified by looking up the peptides in protein sequence databases. When a small peptide sequence fragment that was identified in the experiment is found to be part of a larger protein sequence in the databases of known proteins, the presences of this protein in the sample can be conjectured. However, the whole process is subject to many errors. The major problem is the mass spectrometry, which usually identifies only a fraction of the present peptide sequences.

A proteomics experiment generates a set of peptide sequences that are identified by mass spectrometry. Thus, peptides take the role of words and proteomics experiments with a particular tissue sample act as documents. Peptides do not uniquely identify proteins, thus, the presence of a peptide could indicate the presence of two or more different proteins. This is a case of polysemy on the level of peptides. On the other hand several peptide sequence fragments of the same protein could be identified in the mass spectrometry step. These peptide fragments would be synonyms. Similarity search in large collections of proteomics experiments, such as the PRIDE database[2], could reveal connections between the tissue samples.

1.2.3 Transit Peptides

The endosymbiotic theory [95, 96] describes the origins of mitochondria and chloroplasts in eukaryotic cells as the result of endosymbiotic events. Animals and fungi have only mitochondria, plants have both mitochondria and chloroplasts.

In the first endosymbiotic event, a host cell without mitochondria engulfed an α-proteobacterium that became a symbiont in the cell and developed into mitochondria. Genes originally possessed by mitochondria were transferred into the host cells nucleus. The proteins coded by the transferred genes are synthesized in the cytosol and than transported into the mitochondria. That transport is triggered by transit peptides. Transit peptides are short protein sequences of about 30 to 80 amino acid residues located at the N-terminus of the protein. After the transport of the protein through the membrane of the mitochondrion the transit peptide is cut away.

In the second endosymbiotic event, a host cell that already possessed mitochondria engulfed a cyanobacterium that became the origin of chloroplasts. Also many genes of chloroplasts are transferred to the nucleus. The reimport of the corresponding proteins into the chloroplasts is also mediated by transit peptides. Transit peptides are of very divers nature. There are no patterns of transit peptides known that encode the target of transit peptides, mitochondria or chloroplasts. However, mistargeting of proteins above a tolerance level would disturb the cells integrity. Thus, we must assume that targeting information is coded in transit peptides of plants to differentiate between mitochondria and chloroplasts.

An interesting question is whether the mitochondrial transit peptides in plants adapted after the appearance of a second target, the chloroplasts, to avoid mistar-

[2] `http://www.ebi.ac.uk/pride`

geting. If that is the case, mitochondrial transit peptides in plants would have less degrees of freedom than mitochondrial transit peptides in animals or fungi that have no chloroplasts. The question is tackled by analyzing the similarities among mitochondrial transit peptides in plants on the one hand and those in animals and fungi on the other hand.

1.3 Contributions

We contributed to both the development of new techniques to transfer text mining methods to application domains in life sciences as well as research on new methods in text and data mining. In detail, the contributions of this work are:

- We develop new transformation techniques for 2D-NMR spectra that allow the application of text mining methods like duplicate detection and similarity search based on probabilistic latent semantic analysis.
- We show the improvements of computing similarity between proteomics experiments by using probabilistic latent semantic analysis.
- We developed a new technique to cast hill climbing on a kernel density estimate as an expectation maximization algorithm and demonstrated the improvements by the new technique in density based clustering and probabilistic latent semantic analysis.
- We developed new methods for data analysis based on fractal dimension and used those methods to analyze mitochondrial transit peptide sequences.

Acknowledgements

This work has cumulative character. Most parts are published as peer reviewed papers in journals or conferences proceedings that result from collaborations with students in their diploma theses supervised by the author or/and with collaborators from life science research institutes. In detail, the results described in chapter 2 on duplicate search in 2D-NMR spectra are published in [43, 69] in collaboration with researchers from the Leibniz Institute of Plant Biochemistry (IPB). The findings about the applications of similarity search based on latent semantic analysis to 2D-NMR spectra (chapter chapter 3.2) and proteomics data (chapter 3.3) are published in [73, 82, 83, 137] in collaboration with my diploma students Karina Wolfram and Sebastian Klie and collaborators from the IPB and the European Bioinformatics Institute (EBI). The new connection between KDE and EM theories and the use of KDE priors in PLSA, which is described in chapter 4, is published in [70,71] in collaboration with my diploma student Hans-Henning Gabriel and my PhD. student Andre Gohr. Cluster analysis based on fractal dimension [56], chapter 5.3, was developed during my work in Finland in collaboration with Aris Gionis, Spiros Papadimitriou and Panaiotis Tsaparas. The application of data analysis based on fractal dimension

to mitochondrial transit peptides, chapter 5.2, was published in [128] in collaboration with my diploma student Christine Staiger and Ralf Bernd Klösgen of the Institute of Biology, Martin-Luther University Halle-Wittenberg.

I am thankful for the productive collaborations with all these people. Furthermore, I want to thank my colleagues in the database research group and especially Prof. Dr. Stefan Brass for his steady support of my work and the research atmosphere of freedom he provided in the last years. I want thank the people at the Institute of Informatics of the Martin-Luther University Halle-Wittenberg for their support. Finally, I am thankful for the love and encouragement of my wife, my daughter and the people of the house community I am living in.

2 Duplicate Detection

Duplicates are redundant data representations of the same semantic entity. Typical examples include multiple versions of the same address in a database, which slightly differ in spelling or punctuation, and multiple, scaled and differently compressed versions of the same photo in an image collection. We do not consider exact duplicates that share the same data representation. Finding exact duplicates is another problem of its own. We are dealing with duplicates, the data representations of which may vary among the instances. Detecting those kinds of duplicates is not merely an efficiency problem but requires first and foremost a robust definition of what is considered a duplicate. The material presented in this section is a revised combination of [43, 69].

The tradeoff between efficiency (meaning runtime) of the duplicate detection algorithm and effectiveness of the conceptual definition of duplicates is chosen depending on application scenario. Finding duplicates in large address databases requires fast algorithms while it is sufficient to find the majority of the duplicates. The definition of address duplicates is mainly straight forward and found by domain experts in heuristical ways. Thus, in this application domain, speed of algorithms is the main concern while the duplicate definition should suffice for most common cases.

On the other hand, finding duplicates in photo image collections requires very sophisticated definitions of what is considered a duplicate. While runtime may be important for special cases of large image databases, the effectiveness of the duplicate definition is here a major concern. Most duplicate definition base on some definition of similarity between images. For moderate sized image collections, algorithms with quadratic runtime in the size of the collection may be acceptable as long as the effectiveness of the duplicate detection is guarantied.

An interesting intermediate case is the text mining problem of finding near-document-duplicates. A motivating example is finding near-duplicates in very large collections of web pages. Duplicate detection is used to improve the quality of web search by eliminating mirror pages from the result lists. The tasks is demanding as mirror pages are often not identical with the original but differ in very various ways. This domain combines both requirements, a non-straight forward, robust definition of duplicates and the need for sub-quadratic algorithms that scale to very large collections. Like in the case of image duplicates, similarity between documents is the basis for near-document duplicates.

Finding near-document-duplicates lead to the development of an important theoretical framework called locality sensitive hashing (LSH), that allows to find very similar documents in an approximative, but very fast way. The framework applies to several definitions of similarity mostly from information retrieval. The main idea is

to apply randomly chosen hash functions to the documents. The hash functions are from a proper family of hash functions that fulfills the LSH property. The framework guaranties that hash collisions between two documents happen with a probability equal to the similarity the two documents. Thus, very similar documents, which are candidates for finding near-document-duplicates, have hash collisions with high probability. Finding documents with equal hash values (hash collision) can be done with sub-quadratic algorithms based on sorting.

We use LSH-based solutions for finding near-document-duplicates as a guide to develop methods to find duplicates in large collections of 2D-NMR spectra. 2D-nuclear magnetic resonance (NMR) spectroscopy is a powerful analytical method to elucidate the chemical structure of organic molecules. In contrast to standard one-dimensional NMR spectroscopy, advanced two-dimensional NMR spectroscopy is able to capture the influences of two different atom types at the same time, e.g. ^{1}H (hydrogen) and ^{13}C (carbon).

The result of a 2D-NMR measurement can be seen as an intensity function measured over two independent variables[1]. Regions of the plane with high intensity are called peaks that contain the real information about the underlying molecular structure. A usual visualization of 2D-NMR spectra are contour plots as shown in figure 2.1 (^{1}H,^{13}C-HSQC NMR spectrum)[2]. Contour lines in low intensity regions are clipped away, because they are produced by not reproducible fluctuations. An ideal peak would register as small dot. In the biochemical literature, peaks are noted by their two-dimensional positions. Thus, 2D-NMR spectra are much like documents, peaks take the role of words and spectra can be seen as documents.

However, due to the limited resolution of the devices available (depending on the strength of the magnetic field) multiple peaks may appear as a single merged object with non-convex shape, and after thresholding two different peaks that are close together may be merged and so both are represented by a single point. This is usually accepted. The pattern of peaks is very characteristic and specific for a particular substance.

As modern NMR devices allow the automatic analysis of many samples per day, the number of spectra in a database can be up to several thousands per laboratory. Yet, a lot of manual work is needed to deduce the chemical structure of a complex organic substance from the spectrum. Thus, most of the NMR data is unpublished but contains a lot of experimental knowledge. Duplicate detection is needed for a use case where two or more libraries are merged, and the experimental knowledge for a pair of duplicates needs to be manually merged and curated. The matching of two spectra has to be robust against merged peaks and measurement deviations. The problem is, given an automatically measured spectrum find all matching spectra on the basis of their peaks with annotations. We cast the specific problem in a more general setting: given a set of spectra find all pairs which are near-duplicates.

After briefly reviewing the relevant literature about detecting duplicates (Section

[1] The measurements are in parts per million (ppm).
[2] HSQC: Heteronuclear Single Quantum Coherence

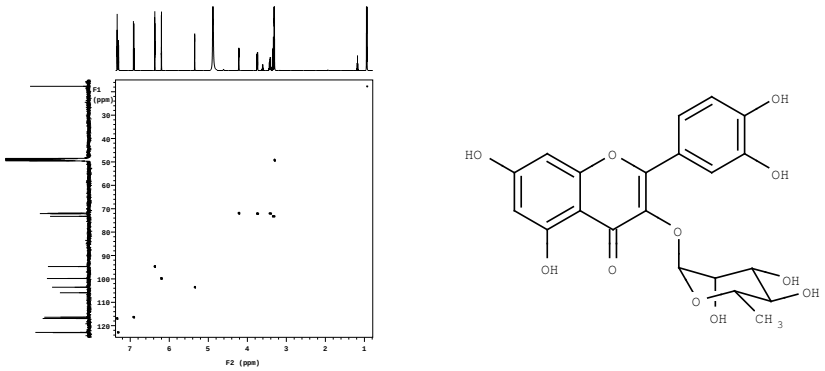


Figure 2.1: 2D-NMR (HSQC) spectrum of Quercetrin, the one-dimensional plots at the axes are projections of the original two-dimensional intensity function including the respective signal intensities. Each peak captures characteristic ^{13}C,^{1}H- atomic resonance interactions present in the specific molecule.

2.1), we give an short overview about locality sensitive hashing in Section 2.2. Fuzzy duplicates of 2D-NMR spectra are defined in Section 2.3. Based on the definition we propose exact methods to find the fuzzy duplicates in Section 2.4. In Section 2.5 we propose approximate methods based on LSH techniques. All methods are experimentally analysed in Section 2.6.

2.1 Applications of Duplicate Detection

2.1.1 Duplicate Detection in Databases

Duplicate detection in databases is mainly known under record linkage. Record linkage and especially the sorted neighborhood method [67] is also related to our approach. Sorted neighborhood determines for every object a key by which the objects are ordered. A sliding window is moved over the sorted sequence and objects within a window are checked for duplicates. The assumption behind the method is that duplicates have keys that are close in the sorted sequence of objects. Therefore, key selection is crucial for the method and is often manually done by domain experts. The sorted neighborhood method has been successfully used for identifying duplicates in customer databases with data objects consisting mainly of discrete attributes. The process of key generation consists primarily of selecting a proper subset of the discrete attributes.

The detection of duplicate records in data streams [37] or click streams [102] are new variants of the problem. Again, duplicates in this context have simple definitions and the records have fixed length. The approaches use Bloom filters, a methods also

based on hashing. The filter is applied to each record and can efficiently answer the question whether the record has no duplicates by using only constant amount of space. The alternative answer of a Bloom filter is: May be the record has a duplicate, in which case the record has to be further processed. As most records does not have duplicates, a Bloom filter can significantly reduce the runtime. The need for constant space makes Bloom filters a well suited technique in streaming scenarios with the requirement that a record cannot be processed multiple times.

While the techniques are useful in database scenarios with simply structured records of fixed length that consists mainly of discrete attributes, they are of limited use for 2D-NMR spectra. As 2D-NMR spectra do not have discrete attributes, the construction of a key as required for record linkage is much more difficult. So far no promising technique is known for multiple numeric attributes. NMR spectra have not fixed size, e.g. the number of peaks may differ between spectra (due to the experimental setup even for chemical duplicates). Also the streaming scenario does not appear naturally for 2D-NMR spectra.

2.1.2 Content-based Duplicate Detection

Duplicate detection can be seen as a special case of content-based similarity search, where pairs of spectra are considered duplicates if their similarity exceeds a certain cutoff value. While content-based similarity search is already in use for 1D-NMR spectra [1,10,85,91,129], to the best of our knowledge, no effective similarity search method is known for 2D-NMR-spectra. Besides technical details (like how to choose the particular cutoff values for similarity) the problem of an approach purely based on similarity is that the similarities between all pairs of spectra have to be computed. This leads to quadratic run time in the number of spectra, which is prohibitive for large spectra databases.

Duplicates are often found by using a similarity measure. Such measures can be manually defined, but in case of strings suitable similarity measures can be learned automatically using a support vector machine [14], which improves the detection accuracy. Another example of very difficult duplicates are those found in the WHO drug safety database [107]. In this case, a classification problem was solved in order to find a measure for comparison of the records. As those duplicates themselves are very difficult to detect, it seems unlikely to find subquadratic algorithms for this problem class.

The detection of duplicates in images [80] is slightly related to our research, as 2D-NMR spectra could be thought as images as well. However, the used techniques in [80] ensure invariance wrt. scaling, shifting and rotation, which is not meaningful in case of 2D-NMR spectra.

The detection of duplicates is slightly related to collision detection in computer graphics [30]. The problem in this concern is to find 2D or 3D objects with overlapping boundaries in real time. The algorithms make the assumption, that only a few bounding boxes of the objects are overlapping. However, in our setting almost all bounding boxes of the spectra overlap. So, collision detection is not applicable to

our problem.

2.1.3 Near-duplicates of Documents

Various aspects of detecting duplicates have received a lot of attention in database and information retrieval research. The closest type of approaches is near-duplicate detection of documents. The efficient detection of near-duplicate documents has been studied by several authors [25, 139]. In particular, near-duplicate detection of web documents is a quite active research area [31, 60, 64]. The difference between near-duplicate documents and fuzzy duplicates of 2D-NMR spectra is that documents are composed of discrete entities, namely words or index terms, but 2D-NMR spectra consists of continuous 2D points. The crucial difference is that the matching operation is transitive for words but not for 2D points. That mean if word A matches word B and word B matches C then A matches also C. This does not hold for two-dimensional points, when matching means that two points are closer than a fixed threshold. An extension of near-duplicate documents are duplicates in XML documents [135], where the set of terms is organized as tree.

2.2 Locality Sensitive Hashing

A general approximation scheme is locality sensitive hashing (LSH) [76], which is a distribution on a family of hash functions F on a collection of objects, such that for two objects x, y

$$Pr_{h \in F}[h(x) = h(y)] = sim(x, y) \tag{2.1}$$

The idea is to construct k hash functions h on the set of objects according to the family F. The percentage of collisions among the k pairs of hash values for two objects estimates the probability of a collision and gives an approximative similarity score. In general, the outcome of a hash function can be thought of as an integer. So, the LSH-scheme maps each object to a k-dimensional integer vector.

In case, two objects x, y are very similar, their integer vectors agree on all k coordinates with high probability. Let be $s = sim(x, y)$, $s \in [0, 1]$ the similarity between x, y, then the probability is s^k that $h_i(x) = h_i(y)$ agree for all $1 \leq i \leq k$. To amplify that probability, the sampling process is repeated L times [55]. So, after L repetitions the probability that their integer vectors agree on all k coordinates at least once is

$$Pr[1 \leq i \leq k\colon h_i(x) = h_i(y) \text{ at least once}] = 1 - (1 - s^k)^L \tag{2.2}$$

Thus, the duplicate detection consist of finding L times the duplicates among integer vectors and union the results. Finding groups of equal integer vectors can be done by sorting, which has lower run time complexity than the naive algorithm.

There are locality sensitive hashing schemes known for the following similarity functions, Manhattan distance between fixed length integer vectors [57], approx-

imative cosine similarity between discrete sets [24], and Jaccard coefficient for set similarity [21,29]. We briefly review the hashing schemes for the similarity measures.

2.2.1 Manhattan Distance

Given a set of d-dimensional integer vectors with coordinates in the set $\{1, \ldots, C\}$, the Manhattan distance between two vectors is $x, y \in X$, $d_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|$. Let be $x = (x_1, \ldots, x_d)$ a vector from X and $u(x) = \text{Unary}_C(x_1) \ldots \text{Unary}_C(x_d)$ a transformation of x into a bit string, where $\text{Unary}_C(a)$ is the unary representation of a with C bits, i.e. a sequence of a ones followed by $C - a$ zeros. For any two vectors $x, y \in X$ there is $d_a(x, y) = d_H(u(x), u(y))$ with d_H is the Hamming distance, which gives the number of different bits between bit strings. An appropriate family of hash functions with the LSH property consists of $h_i(b)$, $1 \leq i \leq \text{length}(b)$, where $h_i(b)$ returns the ith bit from b.

Sampling uniformly from those hash functions and testing for collisions reduces to probabilistically counting the number of equal bits:

$$d_1(x, y) = d_H(u(x), u(y)) = dC(1 - Pr[h_i(u(x)) = h_i(u(y))]) \tag{2.3}$$

with random h_i, $1 \leq i \leq dC$.

For the implementation of this LSH scheme, k random indices $i_1, \ldots i_k$ are picked. The transformation into the Hamming space, which can be quite large, is in practice not necessary. In order to find the value of $h_i(u(x))$ we have to look to which coordinate of the integer vector the index i belongs and if $(i - 1 \mod C) + 1$ is larger than the integer value of that coordinate. So the hash function for index i is

$$h_i(u(x)) = \begin{cases} 1 & \text{if } (i - 1 \mod C) + 1 \leq x_{\lfloor \frac{i}{C} \rfloor + 1} \\ 0 & \text{else} \end{cases} \tag{2.4}$$

2.2.2 Approximate Cosine Similarity

Cosine similarity is used in information retrieval to compare documents which are represented by term frequency vectors. Given a subset $A \subset U$ of a universe U the term frequency vector $\vec{t}_A$ has $|U|$ components, each representing the number of occurrences of a particular element in A. The cosine similarity of A, B is

$$sim_C(A, B) = \frac{\vec{t}_A \cdot \vec{t}_B}{\|\vec{t}_A\| \cdot \|\vec{t}_B\|} \tag{2.5}$$

The hash functions are constructed by randomly mapping each element of U to $\{-1, 1\}$. Lets represent such a mapping $m \colon U \to \{-1, 1\}^{|U|}$ as a vector $\vec{m}$, then the hash function induced by m is

$$h_{\vec{m}}(A) = \begin{cases} 1 & \text{if } \vec{m} \cdot \vec{t}_A \geq 0 \\ 0 & \text{if } \vec{m} \cdot \vec{t}_A < 0 \end{cases} \tag{2.6}$$

The LSH scheme is then

$$Pr[h_{\vec{m}}(A) = h_{\vec{m}}(B)] = 1 - \frac{\theta(\vec{t_a}, \vec{t_b})}{\pi} \approx sim_c(A, B) \tag{2.7}$$

with $\theta(\vec{t_a}, \vec{t_b})$ is the angle between $\vec{t_a}$ and $\vec{t_b}$. The probability is estimated by sampling from the set of possible mappings $\vec{m}$.

2.2.3 Jaccard Coefficient

Given two subsets $A, B \subset U$ of a universe U the Jaccard coefficient is

$$sim_J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.8}$$

The hash functions for the LSH scheme are constructed by random orderings of the universe U. Such a random ordering can by viewed as a random permutation π of the elements of U, where $\pi(\cdot)$ delivers the position of an element according to π. The hash function $h_\pi(A) = \min\{\pi(x) \colon x \in A\}$ returns the smallest position of an element of A with respect to the ordering π. Then for two sets A, B :

$$Pr[h_\pi(A) = h_\pi(B)] = sim_J(A, B) \tag{2.9}$$

The probability is estimated by sampling from the set of possible permutations.

2.2.4 Theoretical Conditions

A LSH-scheme is an attractive tool for similarity-based duplicate detection. Thus, it natural to ask, whether there might exists a LSH-scheme for any similarity function. This is not the case. General conditions for the existence of a LSH-scheme for a given similarity function are proofed in [24].

Lemma 1 *For any similarity function $sim(x, y)$ that admits a locality sensitive hash scheme as defined by equation 2.1, the distance function $1-sim(x, y)$ satisfies triangle inequality, that is for all x, y, z:*

$$1 - sim(x, z) + 1 - sim(z, y) \geq 1 - sim(x, y) \tag{2.10}$$

The proof is given in [24].

As the Jaccard coefficient admits a LSH-scheme it follows from Lemma 1 that the corresponding distance obeys the triangle equation. This was independently proofed in [88, 90] using direct techniques. An example of a commonly used set similarity function with a corresponding distance function that does not obey triangle inequality is the dice coefficient [24]:

$$sim_D(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{2.11}$$

This is shown by the following counter example: Let the sets $A = \{a\}, B = \{b\}$ and $C = \{a, b\}$. Then $sim_D(A, C) = 2/3$, $sim_D(C, B) = 2/3$ and $sim_D(A, B) = 0$ and consequently $1 - sim_D(A, C) + 1 - sim_D(C, B) < 1 - sim_D(A, B)$. Thus, using lemma 1 shows that there exists no LSH-scheme for the dice coefficient.

In case, similarity-based duplicates are defined by $sim(x, y) = 1$, we can derive from lemma 1 the following corollary:

Corollary 1 *For any similarity function $sim(x, y)$ that admits a locality sensitive hash scheme as defined by equation 2.1 and that defines similarity-based duplicates by $sim(x, y) = 1$, the duplicate relation must be transitive, that is for all x, y, z: if x and z are duplicates and z and y are duplicates, then x and y are duplicates.*

Proof: Let the pairs x and z, and z and y are duplicates. By definition of similarity-based duplicates we have $1 - sim(x, z) + 1 - sim(z, y) = 0$. Since due to lemma 1 triangle inequality holds, we have $0 \geq 1 - sim(x, y)$ and because $sim(x, y) \in [0, 1]$ it follows $sim(x, y) = 1$. Thus, x and y are duplicates. □

This tools are helpful, when defining duplicates for two-dimensional NMR-spectra.

2.3 Duplicate Detection of 2D-NMR-Spectra

In this section, we define 2D-NMR spectra and fuzzy duplicates of 2D-NMR spectra in a formal way. Furthermore, based on the properties of the duplicate definition, we discuss the complexity of the problem of detecting all fuzzy duplicates in a given database.

2.3.1 Definition of Fuzzy Duplicates of 2D-NMR Spectra

A 2D-NMR spectrum of an organic compound captures characteristics of the chemical structure like rings and chains. As the shape of the measured peaks varies between experiments (even with the same substance!), we use centroid peak positions for the representation of the spectra. So, we define a spectrum as a set of two-dimensional points:

Definition 1 *A 2D-NMR spectrum A is defined as a set of points $\{x_1, \ldots, x_n\} \subset \mathbb{R}^2$. The $|\cdot|$ function denotes the size of the spectrum $|A| = n$.*

The number of peaks per spectrum is typically between 4 and 60. Our definition of duplicates is based on the idea that peaks can be matched. As spectra are measured experimentally, peak positions can differ even between technical replicates[3]. For that reason, peaks cannot be matched by their exact positions, but rather some slight deviations have to be allowed. A simple but effective approach is to match peaks only within a small spatial neighborhood, The neighborhood is defined by the ranges α and β:

[3] A technical replicate is the same substance/molecule under the same experimental conditions subjected to the measurement device at least twice.

Definition 2 *A peak x from spectrum A* ***matches*** *a peak y from spectrum B, iff $|x.c - y.c| < \alpha$ and $|x.h - y.h| < \beta$, where .c and .h denote the NMR measurements for carbon and hydrogen respectively.*

Based on the notion of matching peaks, we are ready to define a set-oriented similarity measure, from which in turn we derive the definition of duplicates as a special case. Note, that a single peak of a spectrum can match several peaks from another spectrum. Given two spectra A and B, the subset of peaks from A which find matching partners in B is denoted as $matches(A, B) = \{x : x \in A, \exists y \in B : x \text{ matches } y\}$. The function *matches* is not symmetric, but helps to define a symmetric similarity measure

Definition 3 *Let be A and B two spectra and $A' = matches(A, B)$ and $B' = matches(B, A)$, so* ***similarity*** *is defined as*

$$sim(A, B) = \frac{|A'| + |B'|}{|A| + |B|}$$

The measure is close to one if most peaks of both spectra are matching peaks. Otherwise, the similarity drops towards zero.

An important special case of similarity search is the detection of duplicates to increase the data quality of a collection of 2D-NMR-spectra. In addition to the measurement inaccuracies, in case a substance is measured twice with a high and low resolution, it may happen that neighboring peaks are merged to a single one. A restriction to one-to-one relationships between matching peaks can not handle such cases. This means that a single peak from spectrum A can be matching partner for two close peaks from spectrum B.

We propose a definition of fuzzy duplicates based on the similarity measure which can deal with the problems mentioned, namely deviances in peak measurements as well as splitted/merged peaks.

Definition 4 *A pair of 2D-NMR-spectra A and B are* ***fuzzy duplicates****, iff $sim(A, B) = 1$.*

By that definition it is only required that every peak of a spectrum finds at least one matching peak in the other spectrum. The parameters α and β can be set with the application knowledge of typical variances of single peak measurements. For our application, we chose $\alpha = 3$ ppm (^{13}C coordinate) and $\beta = 0.3$ ppm (^{1}H coordinate) if not stated otherwise.

2.3.2 Complexity of the Problem

The duplicate definition 3 is not transitive, that means if A is duplicate of B and B is duplicate of C then A is not necessarily duplicate of C. An example for this fact is sketched in figure 2.2. The reason is the nature of continuous measurements of the peak coordinates. Transitivity of the duplicate relation is in case of similarity-based

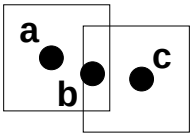


Figure 2.2: The peak a from spectrum A matches peak b from spectrum B and b matches c from spectrum C. However a and c are not matching.

duplicates a necessary condition for the existence of an LSH-scheme. Thus, we can conclude that the similarity function as defined in Definition 3 does not admit an LSH-scheme.

Intuitively, the lack of transitivity has the consequence that a set of duplicate spectra (where all spectra are pairwise duplicates) cannot be represented by a single spectrum. Such a representative would ease the detection of duplicates, since all duplicates of the representative are also pairwise duplicates. Because fuzzy duplicates of 2D-NMR spectra do not have this property, all pairs of the set have to be checked in order to calculate a set of duplicates. Thus, the complexity of an algorithm that finds all duplicates in a set of spectra has a quadratic worst case runtime in the number of spectra. Therefore, we have to resort to heuristics which reduce the experimental runtime on typical data sets.

2.4 Exact Methods for Fuzzy Duplicates

The straight forward implementation of the duplicate definition is to check every pair of spectra, whether the pair is covered by the duplicate definition.

To reduce the complexity of the straightforward method, we need to develop heuristics which (i) quickly check necessary conditions of a pair of spectra to be a duplicate and (ii) avoid false negatives so that no duplicate pairs are missed. A single heuristic may not reduce the number of candidate pairs significantly. Thus, several heuristics can be combined by requiring that all heuristic filter tests are true.

In general, the output of a heuristic is a set of candidate pairs which passed the test of some necessary condition for duplicates. Given the set of spectra S containing m spectra, such an output can be represented by a bit matrix $F \in \{0,1\}^{m \times m}$. In order to reduce the number of candidate pairs several heuristics can be combined. The result of a combination of several heuristics with the respective outputs $F^1, \ldots F^k$ is a bit matrix F with the elements $F_{ij} = F^1_{ij} \wedge \ldots \wedge F^k_{ij}$. Such a bit matrix usually contains many zeros and a few ones, so it can be stored as a sparse matrix. Intermediate results during the combination of outputs of several heuristics should be kept small. Therefore, the processing order should go from the matrix with the smallest number of ones to the matrix with the largest number of ones.

How to find heuristics that implement necessary conditions for a pair being a duplicate in a systematic way? A simple strategy to generate such heuristics is to select a peak x from a spectrum A and search for neighboring peaks of the selected

peak. The spectra that have a peak in the neighborhood of x are candidates for being a duplicate of A. All other spectra cannot be duplicates for A and are therefore excluded from further examination. We call such heuristics peak selecting heuristics.

Our definition of duplicates is symmetric, that means when A is duplicate of B then B is also a duplicate of A. In general, the output of a peak selecting heuristic is not symmetric, because the peak selected from A is not necessarily in the neighborhood of the peak selected from B. There are two options to deal with symmetry. First, symmetry can be used as additional filter. In case, $F_{ij} = 1$ and $F_{ji} = 0$ the bit of F_{ij} can be zeroed. The second option is to ignore symmetry and compute only one half of the bit matrix, such as the upper triangle. The questions is whether the costs to check the additional candidates being generated by ignoring symmetry is lower than the costs to compute the lower triangle bit matrix. We argue that it is beneficial to check a few more candidates to save the costs for the lower triangle bit matrix.

If half of the bit matrix is not computed, the question arises whether we can decrease the number of ones in the upper triangle matrix by selecting a special permutation of rows and columns. The costs to find such a permutation should be small, as it is only beneficial, if those costs pay off by the costs for the saved candidates. Two permutations are interesting candidates: the first option is to sort the spectra by the number of peaks in increasing order. Thus, rows corresponding to spectra with many peaks are at the bottom of the matrix and do not contribute much to the upper triangle of the matrix. The second option is to sort the spectra by the sum of the densities of their peaks in increasing order: rows corresponding to spectra with peaks in high density regions in the peak space are at the bottom of the matrix and do not contribute much to the upper triangle of the matrix. The density in the peak space can be estimated by a two-dimensional histogram.

In case of peak selecting heuristics, the optimal peak that we could select for a spectrum is the peak with the minimal number of matching peaks from other spectra. The set of spectra having a matching peak for the peak $x \in A$ is denoted by

$$N(x) = \{B \in S \colon A \neq B, \exists x' \in B \text{ with } |x'.c - x.c| \leq \alpha \text{ and} |x'.h - x.h| \leq \beta\} \quad (2.12)$$

An optimal peak $x \in A$ has the property $|N(x)| = \min_{x' \in A}\{|N(x')|\}$. Thus, in order to be able to select an optimal peak, it is required to know the total set of spectra S. In contrast to the optimal peak selecting heuristic, the simplest peak selecting heuristic is to select a peak at random from each spectrum. This heuristic will serve as baseline. Other possible heuristics include

Minimal C-Shift Select the peak $x \in A$ with $x.c = \min_{x' \in A}\{x'.c\}$

Minimal H-Shift Select the peak $x \in A$ with $x.h = \min_{x' \in A}\{x'.h\}$

Maximal C-Shift Select the peak $x \in A$ with $x.c = \max_{x' \in A}\{x'.c\}$

Maximal H-Shift Select the peak $x \in A$ with $x.h = \max_{x' \in A}\{x'.h\}$

2.5 Approximate Methods for Fuzzy Duplicates

The exact methods [69], which are guaranteed to have no false negatives, do not scale to very large data sets, even when using peak selecting heuristics. Therefore, we investigate methods which have significantly lower run time. The price for the lower runtime is the possibility of false negatives, that means some duplicate pairs could be missed.

The problem of finding fuzzy duplicates of 2D-NMR spectra is, that the duplicate relation lacks transitivity. The reason is the continuous nature of the peak measurements. So, the idea is to map the peaks to some discrete objects. Among the many possibilities to do that, we will explore two principal alternatives of those mappings. First, the peak coordinates are discretized and then those integers are concatenated to a fixed length vector. Second, the peaks of a spectrum are mapped to discrete objects so that a spectrum is represented by a set of those objects.

The task of finding duplicate spectra is then reduced to finding duplicates of integer vectors and duplicate sets of discrete objects respectively. Both of the latter duplicate relations are transitive, so that a set of duplicates can be specified by a single representative vector or set. In order to check whether a new mapped spectrum belongs to a set of duplicates, it suffices to test the duplicate relation with the representative of the set.

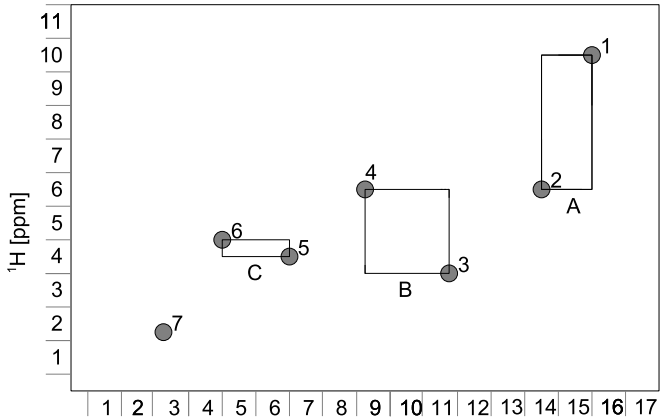
False negatives occur in this approach, when duplicate spectra are mapped to different discrete objects. We propose mappings which map duplicate spectra to discrete objects which are – if not identical – at least very similar.

2.5.1 LSH with Manhattan Distance

The first proposed mapping of 2D-NMR spectra maps transformed peaks to coordinates of the discrete integer vectors. This allows to find identical and similar integer vectors using the LSH-scheme for Manhattan distance. Such a mapping involves three issues, namely (1) how to handle possible splits/merges of peaks, (2) how to order the transformed peaks to a vector, and (3) how to chose the overall dimensionality of the vectors.

Robustification: In order to handle the problem of peak splitting, a peak x of a spectrum is selected and those peaks y are deleted from the same spectrum that are in the neighborhood of x. The neighborhood is given by $N(x) = \{y\colon y \neq x, |x.c - y.c| \leq \alpha \text{ and } |x.c - y.c| \leq \beta\}$. The peaks are selected in decreasing order of $|N(x)|$, so that the peak with the largest number of neighbors is selected first. The iteration stops when each peak in the spectrum is a singleton, i.e. the neighborhoods of the remaining peaks are empty. The remaining peaks are called the *representative peak set* of a spectrum. After this step, we assume a one to one relation between between peaks of duplicate spectra. Except a few pathological cases the assumption typically holds.

Peak Ordering: The coordinates of the representative peaks of a spectrum are discretized by binning. The question remains how to order the discretized peak



Peak	1		2		3		4		5		6		7			
Window	A				B				C				D			
Axis	C		H		C		H		C		H		C	H		
Bin#	15	14	10	6	11	9	6	4	6	4	5	4	3	2		

Figure 2.3: Mapping of peaks from a spectrum to integer vectors for $w = 2$. The blocks of the peaks are indicated by rectangles. The resulting integer vector of the discretized spectrum is shown in the table underneath (last row). The windows and C and H blocks within a window are shown in the second and third row respectively.

coordinates to form a vector, so that the order is not affected by small measurement errors. The most robust order is to sort ^{13}C- and ^{1}H-coordinates independently and discretize afterwards. Then, the vector consists of a block of ^{13}C-coordinates followed by a block of ^{1}H-coordinates. However, this procedure would entirely ignore the joint distribution of ^{13}C- and ^{1}H-measurements but representing the marginal distributions only. So, quite different spectra could be mapped to the same integer vector.

The other extreme is to sort the peaks by a single coordinate – say ^{13}C – and form a vector of alternating discretized ^{13}C- and ^{1}H-coordinates. The information of the joint distribution of ^{13}C- and ^{1}H- coordinates is retained in this mapping. In case of two peaks with close ^{13}C-coordinates but different ^{1}H-coordinates, measurement errors in the ^{13}C-coordinate of a duplicate spectrum could result in swaped order of the two peaks, which in effect also swaps the positions of the ^{1}H-coordinates. In case of two spectra being duplicates their integer vectors could be quite dissimilar, because of the difference in the swaped ^{1}H-coordinates.

We propose an intermediate approach that combines the robustness of the first with the discrimination power of the second. The representative peaks of a spectrum are sorted by one coordinate, say ^{13}C. Starting with the peak of the largest ^{13}C-coordinate, we use a jumping window of w consecutive peaks. We sort the ^{13}C- and

^{1}H- coordinates independently for the w peaks inside a window, and arrange them in blocks as in the first approach. The last window might contain less than w peaks if $\#peaks \mod w \neq 0$. The important aspect of this technique is, that close peaks from different spectra are mapped to the same sorted block, regardless of their order in the ^{13}C- axis. The problem of the second extreme approach can only occur at the jump positions of the window. Thus, by choosing w we can search for a tradeoff between robustness and retained information. The process is illustrated in figure 2.3.

Although some peaks of duplicate spectra might map to different integer vectors due to the binning process, i.e. close peaks coordinates are mapped to different bins, the difference is at most one bin per coordinate. Those duplicates can still be found by lowering the required similarity threshold for the integer vectors below one.

Overall dimensionality: The overall dimensionality D of the set of resulting spectra vectors S is determined by the spectrum having the largest set of representative peaks $D = max(\#peaks(S_i))$. Since the spectra have different numbers of representative peaks, we need to pad their integer vectors up to the fixed dimensionality D. Padding the vectors with zeroes increases their overall similarity, whereas padding by random values would decrease their overall similarity. Therefore we pad a vector by repeating the vector itself until the the length of the maximal vector is reached, thereby retaining the similarity of the original vectors.

2.5.2 LSH with Jaccard Coefficient

We introduce grid-based mappings to transform a spectrum to a set of discrete objects. Duplicates can be efficiently found in the set of the transformed spectra can using the LSH-scheme for set similarity (Jaccard coefficient).

Simple Grids A simple grid-based method is to partition each of the both axis of the two-dimensional peak space into intervals of same size. Thus, an equidistant grid is induced in the two-dimensional peak space and a peak is mapped to exactly one grid cell it belongs to. When a grid cell is identified by a discrete integer vector consisting of the cells coordinates the mapping of a peak $x \in \mathbb{R}^2$ is formalized as

$$g(x) = (g_c(x.c), g_h(x.h)) \text{ with } g_c(x.c) = \left\lfloor \frac{x.c}{\alpha} \right\rfloor,\ g_h(x.h) = \left\lfloor \frac{x.h}{\beta} \right\rfloor$$

The quantities α and β are the extensions of a cell in the respective dimensions. The grid is centered at the origin of the peak space. The cells of the grid act as words. The vocabulary generated by the mapped peaks consists of those grid cells which contain at least one peak. Empty grid cells are not included in the vocabulary. A word consists of a two-dimensional discrete integer vector.

Shifted Grids A problem of the simple grid-based method is that peaks which are very close in the peak space may be mapped to different grid cells, because a cell

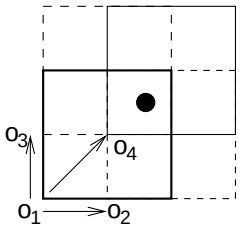


Figure 2.4: The four grids are marked as follows: base grid is bold, $(1,0)$, $(0,1)$ are dashed and $(1,1)$ is normal.

border is between them. So proximity of peaks does not guaranty that they are mapped to the same discrete cell.

Instead of mapping a peak to a single grid cell, we propose to map it to a set of overlapping grid cells. This is achieved by several shifted grids of the same granularity. In addition to the base grid additional grids are used each of them is shifted into one of the three directions $(1,0)(0,1)(1,1)$. An illustration of the idea is sketched in figure 2.4. In figure 2.4, one grid is shifted in each of the directions by half of the extent of a cell. In general, there may be $s-1$ grids shifted by fractions of $1/s, 2/s, \ldots, {}^{s-1}/s$ of the extent of a cell in each direction respectively. For the mapping of the peaks to words which consist of cells from the different grids, two additional dimensions are needed to distinguish (a) the $s-1$ grids in each direction and (b) the directions themselves. The third coordinate represents the fraction by which a cell is shifted and the fourth one represents the directions by the following coding: value 0 is (0,0), 1 is (1,0), 2 is (0,1) and 3 is (1,1). So, each peak is mapped to a finite set of four-dimensional integer vectors. A nice property of the mapping is that for two matching peaks there exists at least one grid cell in one of the four grids both peaks are mapped to.

2.6 Experiments

In this section we evaluate the proposed definition of duplicates and conduct experiments to investigate the tradeoff between costs for candidate filtering of the approximative methods and candidate checking of the exact methods.

2.6.1 2D-NMR Database

The substances included in the database are mostly secondary metabolites of plants and fungi. They cover a representative area of naturally occurring compounds and originate either from experiments or from simulations[4] based on the known structure of the compound. The database includes 1524 spectra with 2 to 60 peaks each, for a total of about 20,000 peaks. The density in the peak space for all peaks in the

[4]ACD/2D NMR predictor, version 7.08, http://www.acdlabs.com/

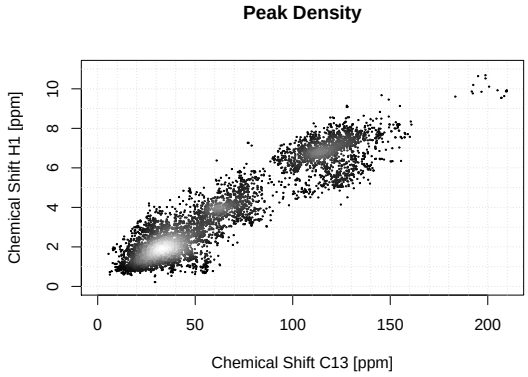


Figure 2.5: Density of the peaks of all spectra. Light gray means higher density. Note that when plotting a spectrum with ^{13}C as x-axis (0-220)ppm and ^{1}H as y-axis (0-12)ppm, aromatic structures are located in the upper right region and aliphatic structures are located in lower left region.

database is shown in figure 2.5. The database is a preliminary sample of a much larger database of 2D-NMR spectra that is currently build by our collaborators at the Leibniz Institute of Plant Biochemistry.

We generated larger data sets based on the real 2D-NMR data set while keeping the overall distribution of peaks. First, we estimated the mean of a Poisson distribution from the number of peaks per spectrum. The average number of peaks per spectrum is 12.3. The size of a newly generated spectrum is a random integer drawn from the Poisson distribution. In order to generate a new spectrum, we select a real spectrum at random and copy a fixed percentage p of randomly selected peaks from that spectrum into the new one. This step is repeated until the spectrum has reached the predetermined size. In order to avoid identical peaks, Gaussian noise is added to the copied peaks. In the experiments we have set the percentage p to different values $10, 20, 40, 60$ without obtaining significant differences in the results. Therefore, we excluded this variable from our analysis.

2.6.2 Performance of Exact Methods

We conducted experiments with artificially generated data of sizes 1000, 1500, 2000, 2500, and 5000 spectra. All data sets include the set of real spectra as well as generated spectra. The set of duplicates was the same as for the real data in all cases, because it is unlikely that a duplicate is generated at random by our procedure.

First, we investigate how the number of candidate pairs found by a heuristic grows with the size of the data set. The results are shown in figure 2.6 (left). The simple straightforward method checks all $n(n-1)/2$ pairs, where n is the number of spectra.

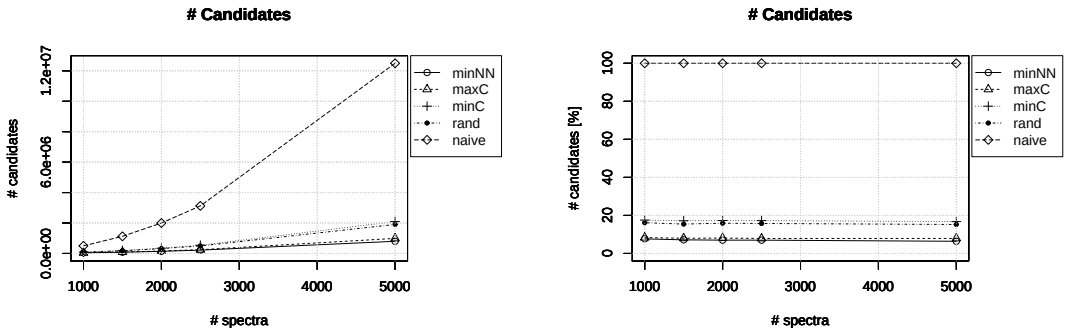


Figure 2.6: Number of candidate pairs versus number of spectra (left), percentage of candidate pairs versus number of spectra (right).

The optimal peak selecting heuristic minNN generates the smallest set of candidates. Because the shift-C and the shift-H coordinates are strongly correlated, the minC and minH as well as maxC and maxH respectively produce very similar results. Therefore, we present only the results for minC and maxC. The maxC heuristic produces only slightly more candidates as minNN. Random and minC are similar but have considerably more candidates than the optimal minNN heuristic. The better performance of maxC compared to minC and rand can be explained by the fact that the point density in the upper right corner of the peak space is lower than in the rest of the space (see figure 2.5). We normalized the number of candidates by the heuristics to be precentages of the number of pairs checked by the straightforward method, shown in figure 2.6 (right). The figure shows that the fractions stays constants as the data set grows. This indicates that the advantage of the heuristics will not degrade as the peak space becomes more and more densely populated with peaks.

In figure 2.7 we report the total run times of the heuristics. As the costs to find the candidates are different for the heuristics, we cannot expect that the situation for the plot with the number of candidates directly carries over to the run times. The figure shows that the optimal heuristic has a larger runtime than the other three heuristics, because the costs to generate the candidates are much higher. The maxC shows the best run time results because the number of generated candidates is low and the costs to generate those candidate are comparable to minC and rand. So, maxC with the given SQL implementation (see [69] for details) is the best choice to find duplicates in 2D-NMR data.

2.6.3 Performance of Approximate Methods

We implemented the approximate methods as SQL statements using the SQL 1999 standard (see [43] for details). The used data are the 1524 original spectra, which contain 118 fuzzy duplicates. The run times of the approximate methods are below

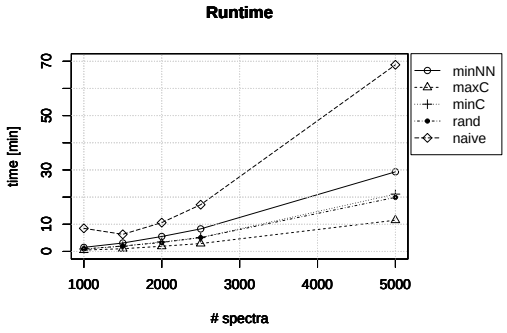


Figure 2.7: Total run time in minutes versus number of spectra.

20 seconds for all methods. That is a large speedup with respect to the exact methods as well as the heuristics proposed in [69], since those methods run several minutes on that data. The actual speedup depends on the size of the used data set, since the methods of the two classes have different runtime complexities (n^2 versus $n \log n$).

For the approximate methods, we investigate the number of false positives and false negatives for different numbers k of sampled hash functions. First, the parameter $L = 5$ is fixed. For small k more spectra are likely to be reported as similar. The larger k, the more the reported integer vectors as well as the discrete sets have to be identical. Since our mapping to discrete integer vectors and discrete sets respectively may cause false negatives, we want to allow a some variability of the detected spectra.

A relevant performance measure is the number of false positives for very small false negatives. At this point, the reported similar spectra can be subsequently checked with the naive exact method to exclude the false positives. In that respect, the approximate method acts as a strong filter while only few true duplicates are missed. The results for Manhattan distance with LSH are shown in figure 2.8. Here the number of false positives is about 390 without any false negative. For Jaccard coefficient with Minhashing we tested the mapping to simple grids and shifted grids. The number of false positives are about 900 and 500 respectively, as shown in figure 2.9.

As Jaccard coefficient with Minhashing gives more false negatives than the Manhattan distance, additionally, we experimented with different values for L. The results are shown in table 2.1. The table shows (especially in the two blocks at the bottom) that increasing L produces more false positives while the number of false negatives is reduced at the same time.

All reported measurements are averages of five runs. The main point is that merely several hundreds of spectra must be explicitly checked as putative duplicates compared to two millions $(1524 \cdot (1524-1)/2)$ for the naive method. For comparison, the best exact heuristic reported in [69] still needs to check about 30,000 duplicate pairs with the naive method. So, approximate methods have a huge performance

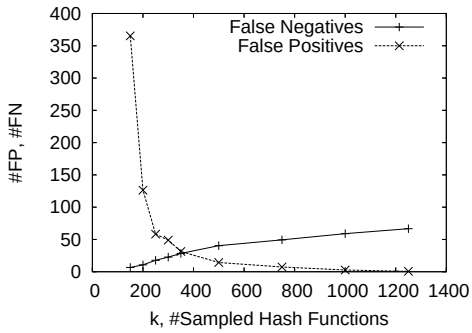


Figure 2.8: Number of false positives and false negatives FP,FN for Manhattan with LSH ($L = 5$) and different k for four repeated experiments.

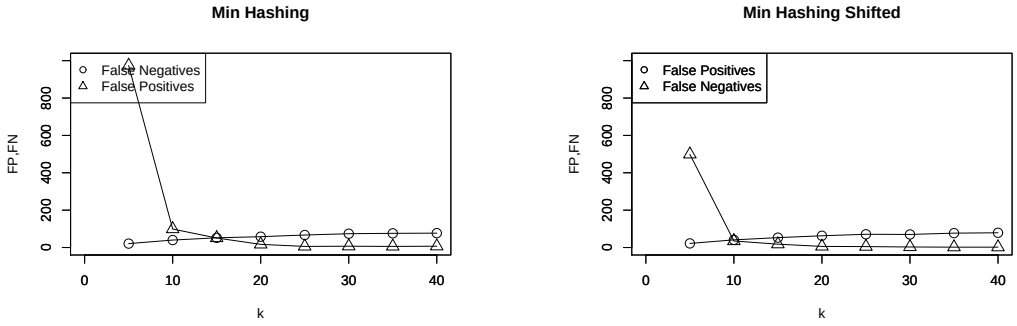


Figure 2.9: Number of false positives and false negatives for Jaccard coefficient with Minhashing ($L = 5$), simple grids (left) and shifted grids (right).

Table 2.1: Number of false positives and false negatives for Jaccard coefficient with Minhashing for different setting for L and k.

k	L	Minhashing		Minhashing+Shift	
		FN	FP	FN	FP
2	1	42	9352	46	2918
3	1	59	252	55	558
4	1	67	170	57	168
5	1	69	57	66	47
2	5	19	15167	11	13828
3	5	32	2626	31	1540
4	5	39	514	36	547
5	5	46	199	47	183
5	10	35	444	31	285
5	15	26	654	17	481
5	20	25	836	16	584
5	50	20	1445	12	1119

gain.

In conclusion, the mapping to integer vector in combination with Manhattan distance and LSH turned out to be the best method, delivering the least number of false positives and no false negatives. The mapping to shifted grids is better than the mapping to simple grids, but the number of false positives is higher. However, the minhashing method has a slight runtime advantage, since less hash functions need to be sampled. This might be useful in case of very large data sets.

2.6.4 Detected Duplicates

There were no duplicates intentionally included in the database. With a setting of $\alpha = 3$ppm and $\beta = 0.3$ppm, which are reasonable tolerances, 118 of 2,322,576 possible pairs are reported as fuzzy duplicates.

The found duplicate pairs revealed the following types of classes of duplicates occurring in practice: (i) accidental entry of the same spectra/substance with different names, (ii) spectra prediction software ignoring stereochemical quaternary carbon configurations, (iii) some pairs consist of an experimental and a simulated spectrum (see figure 2.10) of the same substance (which speaks for both our duplicate definition and the simulation software), (iv) same chemical compound in different measurement conditions (measurement frequency, solvent).

The results included also a surprise, namely the pair Thalictrifoline/Cavidine. Both structures differ only in the stereochemical orientation of one methyl group. Evidently, in this case the commercial software package used in the simulation is unable to reflect the different stereochemistry in calculated spectra. In the future, fuzzy duplicates will be used to improve the quality of collections of 2D-NMR spectra.

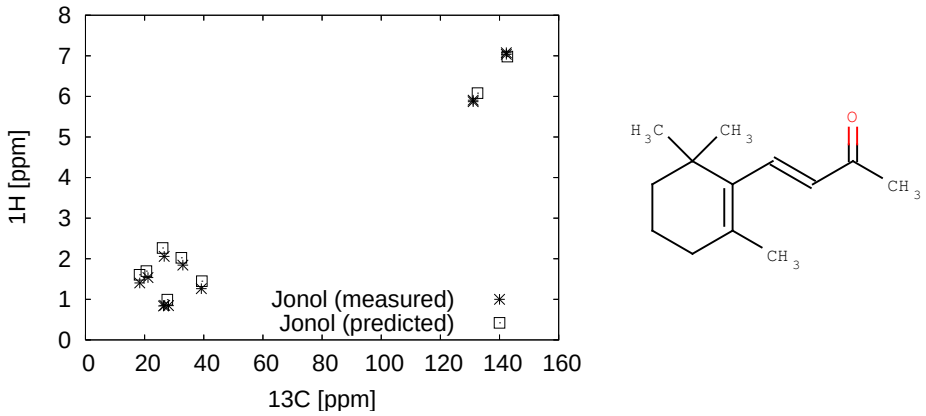


Figure 2.10: Two spectra as an example for a detected duplicate in our database: Peaks as simple points from an experimental and predicted spectrum of β –Jonol. Note, that each peak in A has matching peak in B according to $\alpha = 3.0ppm$ and $\beta = 0.3ppm$.

2.7 Summary

We proposed a simple and robust definition for fuzzy duplicates of 2D-NMR spectra on the basis of co-matching peaks. Considering peak splitting as well as inherent measurement errors is crucial to find duplicates in 2D-NMR spectra collections. We described ideas and heuristics to embed 2D-NMR spectra data into vector spaces and discrete objects to suitably interface NMR-data to text mining algorithms. A scale up to large data volumes is achieved by applying approximate and fast algorithms as preliminary filters prior to the computation of the exact duplicates, avoiding the quadratic nature of searching for duplicates in sets of spectra. The developed methods are the foundation to start and manage a large collection of NMR spectra, which is part of an ongoing metabolomics project at the IPB in Halle (Saale).

3 Similarity Search and Latent Semantic Analysis

Information retrieval and similarity search has been applied to find relevant documents in large document collections. The effectiveness of the retrieval approach mainly depend on the representation of the documents in the collection. The dominant approach is to model documents as bag of words. This model does not represent the order of words in a document nor are any substructures such as sections and paragraphs retained. The vector space model for documents, which is the mainly used one among the instances of the bag-of-word-model, represents only the frequency of a term (word) in a document. The material presented in this section are revised versions of our publications on 2D-NMR similarity search [73, 137] and those on latent semantic analysis of proteomics experiments [82,83].

A document collection can be thought of as a term-document matrix. Each document is a column vector with a dimensionality of the size of the vocabulary. An entry in a document vector is the word frequency of the term in the respective document. Thus, the matrix has as many rows as there are words in the vocabulary and as many columns as there are documents in the collection. Such a matrix is only sparsely populated with non-zero entries, e.g. assume there are one million documents each having about thousand different words out of a vocabulary o five hundred thousand words. The total number of matrix entries is $0.5 \cdot 10^{12}$, however, the number of non-zero entries is about 10^9, which is three magnitudes smaller than the total number of entries. Typically a document collection is stored as sparse matrix. i.e. only the non-zero elements are represented.

Documents are compared by similarity functions among which the cosine similarity is a prominent one. Cosine similarity compares two documents by multiplying element-wise the respective document vectors and summing up all products to a score that is normalized by the sizes of the documents. In order to get a large similarity score, the documents need to have many words in common that is in many element-wise products of the respective term frequencies both factors need to be non-zero.

Because the use of language in documents is divers and governed by many factors, the words used to express the same or similar facts differ from document to document. As a consequence, similar documents may appear to be not similar according to cosine similarity just because they use different words.

One approach to remedy this effect is to represent documents by topics instead by words. The presence of a topic in a document implies the presences of a set of words that belong to a specific semantic context. Topics are similarly represented as

documents in the vector space model, namely by a vector with a dimensionality of the size of the vocabulary. The topic representation of a document is also a vector but now the number of dimensions is given by the number of topics. Such a document specific vector is a list coefficients that govern how the underlying topics vectors are combined to approximate the original document vector of word frequencies. When comparing documents by measuring cosine similarity of document specific vectors of topic coefficients, the diversity of word usage in natural language is hidden behind the topics. Similar documents appear similar according to this new measure, when they are similarly represented by the topics. The particular choices of words used in the documents to indicate a topic have not any longer a direct effect on the similarity measure. By this way, noise contained in the documents is reduced.

An open question is, how to find the topics. There are several theories used to this effect that give rise to methods to automatically determine topics from a given collection. Early methods base on the theory of matrix factorization. The goal is to approximate the term document matrix by a product of two much smaller matrices. The computed topics, the columns of one of the two matrices, are termed latent factors, therefore, the method is called latent semantic analysis (LSA). Despite the method is based on basic theory of linear algebra, which introduces its own restrictions like orthogonality of the latent factors (topics) and gives certain theoretical guaranties for the optimality of the result, it can be difficult to interpret the found topics in a semantic way.

Alternatively, finding topics from a given document collection can be formulated as a problem of learning a probabilistic mixture model. Mixture models have been used to cluster multi-dimensional data. In this context, topics are seen as clusters to which pairs consisting of identifiers of both documents and words are probabilistically assigned. The model learns topics as multinomial distributions over the vocabulary. A topic covers semantically related words by assigning them high probability, while words that are semantically unrelated to the topic should get low probability. From the view of machine learning, the model captures correlations between words that often occur together in documents. The interpretation of topics as probability distributions over words can help to find a semantic interpretation of a topic.

The abstract view on documents as bags of words in addition to the developed methods to search large collections of documents in that kind of representation makes the methodology an attractive framework for other types of data. In this chapter, we analyze the application of methods for information retrieval and latent semantic analysis to search 2D-NMR spectra as well as to analyze proteomics experiments.

Nuclear magnetic resonance (NMR)-spectra are an important fingerprinting method to investigate the chemical structure of organic compounds from plants or other tissues. Two-dimensional-NMR spectroscopy is able to capture the influences of two different atom types at the same time (e.g. ^{1}H, hydrogen and ^{13}C carbon). The result of an 2D-NMR experiment can be seen as an intensity function measured over two variables[1]. Regions of high intensity are called peaks, which contain the real

[1] The measurements are in parts per million (ppm).

information about the underlying molecular structure. The usual visualizations of 2D-NMR spectra are contour plots as shown in figure 2.1. An ideal peak would register as a small dot, however, due to the limited resolution available (dependent on the strength of the magnetic field) multiple peaks may appear as a single merged object with non-convex shape. In the literature peaks are noted by their two-dimensional positions without any information about the shapes of the peaks. Content-based similarity search of 2D-NMR spectra would be a valuable tool for structure investigation by comparing spectra of unknown compounds with a set of spectra, for which the structures are known. While the principle is already in use for 1D-NMR spectra [1,10,85,91,129], to the best of our knowledge, no effective similarity search method is known for 2D-NMR-spectra.

Simplified, a 2D-NMR spectrum is a set of two-dimensional points. There is an analogy to text retrieval, where documents are usually represented as sets of words. Latent space models [18, 75, 113] were successfully used to model documents and thus improved the quality of text retrieval. Recently, a diversity of text mining approaches for different problems [22, 100, 131] have been proposed, which make use of probabilistic latent space models. The goal of this work is to show by example how to apply text retrieval and mining methods to biological data originating from experiments.

The contribution of this work are methods to map 2D-NMR spectra to discrete text-like data, which can be analyzed and searched by any text retrieval method. We evaluate on real data the performance of two text retrieval methods, namely the standard vector space model [117] and probabilistic latent semantic analysis [75] in combination with our mapping methods for 2D-NMR spectra. Additionally, we investigate a simple similarity function proposed for duplicate detection, which operates directly on the peaks of the spectra and serves as bottom line benchmark in the experimental evaluation. Our results indicate at a larger scope that text retrieval and mining methods, designed for text data created by humans, in combination with appropriate mapping functions may yield the potential to be also successful for experimental data from naturally occurring objects. In this paper we consider exemplarily ^{1}H, ^{13}C one-bond heteronuclear shift correlation 2D-NMR spectra.

Our second application is similarity search in proteomics experiments. The field of proteomics has undergone several dramatic changes over the past few years. Advances in instrumentation and separation technologies [2,39] have enabled the advent of high-throughput analysis methods that generate large amounts of proteomics identifications per experiment. Many of these datasets were initially only published as supplementary information in PDF format and, while available, were not readily accessible to the community. Obviously, this situation led to large-scale data loss and was perceived as a major problem in the field [66, 114].

Several public proteomics data repositories, including the Global Proteome Machine (GPM) [33], the Proteomics Identifications Database (PRIDE) [78, 97] and PeptideAtlas [38] were constructed to turn the available data into accessible data, thereby reversing the trend of increasing data loss.

As a case in point, several large-scale proteomics projects that have recently been

undertaken by the Human Proteome Organization (HUPO), including the Plasma Proteome Project (PPP) [109] and the Brain Proteome Project (BPP) [62], have published all of their assembled data in one or more of these repositories. As a result, their findings are readily accessible to interested researchers. It is therefore remarkable to see that very little additional information has so far been extracted from the available data. One of the rare examples where the analysis of large proteomics datasets resulted in a practical application is the recent effort by Mallick and co-workers in which several properties of a large amount of identified peptides were used to fine-tune an algorithm that can predict proteotypic peptides from sequence databases [92].

We here present a novel way to reveal the information that lies hidden in large bodies of proteomics data, by analyzing them for latent semantic patterns. We used the original peptide sequences to evaluate experiment similarity by performing a latent semantic analysis. Our results suggest that LSA as well as the probabilistic variant PLSA can be considered useful analysis tools of such data yielding results which cannot easily be obtained by conventional means.

3.1 Information Retrieval and Latent Semantic Analysis

3.1.1 Vector Space Model

We briefly introduce the essentials of the vector space model [94]. In the vector space model, documents are represented by vectors which have as many dimensions as there are words in the used vocabulary of the document collection. Each component of a document vector reflects the importance of the corresponding word for the document. The typical quantity used is the raw term frequency (tf) of that word for the document, say the number of occurrences of that word in a document d.

Words are not equally selective. Some words appear in nearly every document, others are quite rare. In order to improve the retrieval quality, the weights of the words are reweighed by multiplying the term frequency with the inverse document frequency (idf) of a word. The inverse document frequency measure is large, if a word is used in a small percentage of the documents in the collection. This is implemented by the following formula:

$$idf(w) = \log \frac{N}{N_w} \tag{3.1}$$

The quantity N is the total number of documents and N_w is the number of documents that contain the word w. The effect of idf-weighting is that poorly differentiating, often-occurring terms will have a much lower weight than highly specific, rare terms.

Formally, we denote the set of documents by $D = \{d_1, ..., d_N\}$ and the vocabulary by $W = \{w_1, ..., w_M\}$. The term frequency of a word $w \in W$ in a document $d \in D$ is denoted as $n_{d,w}$ and the reweighed quantity is $\hat{n}_{d,w} = n_{d,w} \cdot idf(w)$. The similarity

between a query document q and a document d from the collection is

$$sim(d,q) = \frac{\sum_{w \in W} \hat{n}_{d,w} \cdot \hat{n}_{q,w}}{\sqrt{\sum_{w \in W} \hat{n}_{d,w}^2} \cdot \sqrt{\sum_{w \in W} \hat{n}_{q,w}^2}} \tag{3.2}$$

This can be interpreted as the cosine of the angles between the two vectors.

The vector space model has several drawbacks. First, it relies on exact term matching (because of the scalar product in (3.2)), thus making it impossible for the model to detect and use similarity information expressed by synonyms. Second, correlations between words are neglected. Two documents appear dissimilar, if they have only a few words in common, even when many words in one document are strongly correlated to respective words in the other document.

3.1.2 Latent Semantic Analysis

Let A be the term document matrix, where each document is represented as a column. In order to reduce the sparseness of the term document matrix and to detect hidden (latent) term relations, LSA projects the original documents vectors into a lower dimensional semantic space where documents that contain repeatedly co-occurring terms will have a similar vector representation. This effectively overcomes the fundamental deficiencies of the exact term-matching employed in a VSM [87]. As such, LSA might predict that a given term should be associated with a document, even though no such association was observed in the original matrix [35]. The core principle for achieving this is the application of singular value decomposition (SVD), a type of matrix factorization that can be applied to any rectangular matrix in the form of:

$$A = U \Sigma V^T \tag{3.3}$$

where $U \in \mathbb{R}^{M \times M}$ and $V \in \mathbb{R}^{N \times N}$ are orthogonal matrices (i.e. $UU^T = I$ and $VV^T = I$) and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$ is a diagonal matrix with $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r \geq 0$ and $r = \min(M, N)$. LSA makes use of the matrix approximation theorem, which states that

$$\underset{A_k \text{ has rank } k}{\text{argmin}} \|A - A_k\|_F = U_k \Sigma_k V_k^T \tag{3.4}$$

with U_k and V_k consist of the first k columns of U and V respectively and $\Sigma_k = \text{diag}(\sigma_1, \ldots, \sigma_k)$. Thus, for $k \leq r$, A_k is the best rank k approximation of A in the sense of the Forbenius norm. The approximation error is bounded with respect to the Frobenius norm by $\|A - A_k\|_F \leq \sigma_{k+1}$. The column vectors of $(\Sigma_k V_k^T)$ are the new document vectors in the latent space. The mapping of some original document vector $\vec{d}$ into the latent space is described by $U^t \vec{d}$.

The SVD alters the original values in the matrix A by new estimates, based on the observed co-occurrences of terms and their 'true semantic meaning' within the whole corpus of documents [53]. The latter is achieved because terms with a common meaning are roughly mapped to the same direction in the latent space. By leaving

out the smallest singular values, 'weak patterns' or noise are filtered out. The choice of k determines the degree of reduction, and it is therefore important to note that a high k value (corresponding to a weak reduction) might not be able to filter out noise or unimportant fluctuations in the source data, while a very small k value (strong reduction) will retain too little information from the original data structure [40].

3.1.3 Probabilistic Latent Semantic Analysis

Probabilistic latent semantic analysis (PLSA) introduced in [75] extends the vector space model by learning topics hidden in the data. The training data consists of a set of document-word pairs $(d^{(i)}, w^{(i)})_{i=1,\ldots,N}$ with $w^{(i)} \in W$ and $d^{(i)} \in D$. The joint probability of such a pair is modeled according to the employed aspect model. The model introduces the hidden variable $\vec{z} \in \{0,1\}^K$, which is a K-dimensional bit vector. The variable follows an 1-out-of-K scheme that means only one of the K bits is set, i.e. $\sum_{k=1}^{K} z_k = 1$. In the context of text retrieval $\vec{z} = (z_1, \ldots, z_K)$ is interpreted as an indicator for a topic. Two assumptions are made by the aspect model. First, it assumes that document word pairs (d, w) are statistically independent and identically distributed. Second, conditional independence between w and d is assumed for a given value of $\vec{z}$.

$$p(d,w) = \sum_{\vec{z}} p(d,w,\vec{z}) \tag{3.5}$$

$$= \sum_{\vec{z}} p(d,w|\vec{z})p(\vec{z}) \tag{3.6}$$

$$= \sum_{\vec{z}} p(d|\vec{z})p(w|\vec{z})p(\vec{z}) \tag{3.7}$$

$$= p(d) \sum_{\vec{z}} p(\vec{z}|d)p(w|\vec{z}) \tag{3.8}$$

$$= p(d) \sum_{k=1}^{K} p(z_k = 1|d) \cdot p(w|z_k = 1) \tag{3.9}$$

The second step applies the conditional independence assumption and the third step transforms the symmetric form of the aspect model into the asymmetric, document-centric form, which is the commonly used one.

The probabilities necessary to compute the joint probability $p(d, w)$, namely $p(d)$, $p(w|\vec{z})$ and $p(\vec{z}|d)$, are all multinomial distributions. The parameter of those distribution can be thought of as matrices, i.e. $p(d)$ has a N-dimensional parameter vector $\vec{\delta}$, $p(w|\vec{z})$ and $p(\vec{z}|d)$ have $(M \times K)$- and $(K \times N)$-dimensional parameter matrices $\vec{\omega}$ and $\vec{\theta}$ respectively. Those parameter matrices are estimated by an expectation maximization (EM) algorithm. The idea is to find values for unknown parameters

that maximize the log-likelihood

$$\log p(D,W) = \sum_{d\in D}\sum_{w\in W} n_{d,w}\cdot\left[\log p(d) + \log\sum_{k=1}^{K} p(\vec{z}_k = 1|d)\cdot p(w|\vec{z}_k = 1)\right] \tag{3.10}$$

$$= \sum_{d\in D}\sum_{w\in W} n_{d,w}\cdot\left[\log \delta_d + \log\sum_{k=1}^{K} \theta_{kd}\cdot\omega_{wk}\right] \tag{3.11}$$

The latter form writes the log-likelihood in term of the parameter matrices. Because of the sum inside the logarithm, no closed form for the maximum configuration of the parameter matrices can be found. In contrast, the complete data log-likelihood, which additionally depends on the particular allocation of the hidden variables $\vec{z}$, does not have this difficult form. Let be $\vec{Z} = \{\vec{z}_{dw}|d\in D, w\in W\}$ a bit-matrix that describes particular states of the hidden variables for all $|D|\cdot|W|$ possible document-word pairs. The complete data log-likelihood is

$$\log p(D,W,Z) = \sum_{d\in D}\sum_{w\in W}\sum_{k=1}^{K} n_{d,w}\cdot\left[\log p(d) + z_{dwk}\Big(\log p(\vec{z}_k = 1|d) + \log p(w|\vec{z}_k = 1)\Big)\right] \tag{3.12}$$

$$= \sum_{d\in D}\sum_{w\in W}\sum_{k=1}^{K} n_{d,w}\cdot\left[\log \delta_d + z_{dwk}\Big(\log\theta_{kd} + \log\omega_{wk}\Big)\right] \tag{3.13}$$

The variable z_{dwk} is the k-th component of the bit-vector $\vec{z}_{dw}$ for the document-word pair (d,w). Following the EM-framework [36, 99], maximizing the expectation of the complete log-likelihood is equivalent to maximizing the data log-likelihood (3.11). The expectation of the function (3.13) is taken over the posterior distribution $p(\vec{Z}|D,W)$, thus, we are maximizing

$$\mathbb{E}_{\vec{Z}}[\log p(D,W,Z)] = \sum_{\vec{Z}} p(\vec{Z}|D,W)\cdot\log p(D,W,Z) \tag{3.14}$$

$$= \sum_{d\in D}\sum_{w\in W}\sum_{k=1}^{K} n_{d,w}\cdot\left[\log\delta_d + \mathbb{E}[z_{dwk}]\Big(\log\theta_{kd} + \log\omega_{wk}\Big)\right] \tag{3.15}$$

$$= \sum_{d\in D}\sum_{w\in W}\sum_{k=1}^{K} n_{d,w}\cdot\left[\log\delta_d + p(z_k = 1|d,w)\Big(\log\theta_{kd} + \log\omega_{wk}\Big)\right] \tag{3.16}$$

This is a function depending only on the parameters $\vec{\delta}$, $\vec{\theta}$ and $\vec{\omega}$. The expectation (3.15) is over a sum of random variables, namely the z_{dwk}. Thus, it can be broken down into a sum of individual expectations of the z_{dwk}. Because, the z_{dwk} are bit variables, the expectations are the posteriors.

The idea of the EM-algorithm is to maximize (3.16) iteratively with two steps per iteration. After initialization of the parameters, in the first step, the expectations of the indicator variables z_{dwk}, the posteriors, are computed with respect to fixed parameter values (E-step). In the second step, the posteriors are fixed and (3.16) is maximized with respect to the parameters (M-step). The two steps are iterated until convergence. Starting from a random initialization, the EM-algorithm is guarantied to reach a local maximum of (3.16).

In detail, the E-step computes the individual expectations of the z_{dwk}, the posteriors, given $\vec{\theta}$ and $\vec{\omega}$:

$$\mathbb{E}[z_{dwk}] = p(z_k = 1|d, w) = \frac{p(z_k = 1|d) \cdot p(w|z_k = 1)}{\sum_{k'=1}^{K} p(z_{k'} = 1|d) \cdot p(w|z_{k'} = 1)} \tag{3.17}$$

$$= \frac{\theta_{kd} \cdot \omega_{wk}}{\sum_{k'=1}^{K} \theta_{k'd} \cdot \omega_{wk'}} = \gamma_{dwk} \tag{3.18}$$

The posteriors γ_{dwk} are used to update the parameters in the M-step. The update equations are derived by maximizing (3.16) with respect to θ_{kd} and ω_{wk}:

$$\theta_{kd} = \frac{\sum_{w \in W} n_{d,w} \gamma_{dwk}}{\sum_{w \in W} n_{d,w}} \tag{3.19}$$

$$\omega_{wk} = \frac{\sum_{d \in D} n_{d,w} \gamma_{dwk}}{\sum_{d \in D} \sum_{w' \in W} n_{d,w'} \gamma_{dw'k}} \tag{3.20}$$

The parameter vector $\vec{\delta}$ for $p(d)$ has a maximum likelihood estimator, which is independent of the hidden variables, namely

$$\delta_d = \frac{\sum_{w \in W} n_{d,w}}{\sum_{d' \in D} \sum_{w \in W} n_{d',w}} \tag{3.21}$$

There are serval ways to answer similarity queries using the trained model. Because of its simplicity, we adopt the PLSA-U variant from [75]. The idea is to extend the cosine similarity measure from the tf-idf vector space model. The extension by Hofmann treats the learned multinomials $p(w|d)$ as term frequencies (tf). Note that $p(w|d) = p(d, w)/p(d)$. The multinomials $p(w|d)$ are smoothed variants of the original term frequencies $\tilde{p}(w|d) = n_{d,w}/(\sum_{w' \in W} n_{d,w'})$. The proposed tf-weights are linear combinations of the multinomials $p(w|d)$ and $\tilde{p}(w|d)$. Thus, the new tf-idf weights used for the documents within the similarity calculation (3.2) are

$$\hat{n}_{d,w} = (\lambda \cdot p(w|d) + (1 - \lambda) \cdot \tilde{p}(w|d)) \cdot idf(w)$$

with $\lambda \in [0, 1]$. Hofmann suggests in [75] to set $\lambda = 0.5$. The tf-idf weights for the query are determined as in the standard vector space model. The smoothed tf-weight for a word, which actually does not appear in a document, may be still non-zero when the word belongs to a topic that is active in the particular document.

In that way a more abstract similarity search becomes possible.

3.2 Similarity Search of 2D-NMR Spectra

Searching and mining nuclear magnetic resonance (NMR)-spectra of naturally occurring substances is an important task to investigate new potentially useful chemical compounds. Multi-dimensional NMR-spectra are objects like documents, but consists of continuous multi-dimensional points called peaks instead of words. We develop several mappings from continuous NMR-spectra to discrete text-like data. With the help of those mappings any text retrieval method can be applied. We evaluate the performance of two retrieval methods, namely the standard vector space model and probabilistic latent semantic analysis (PLSA). PLSA learns hidden topics in the data, which is in case of 2D-NMR data interesting in its owns rights. Our experiments show that the vector space model as well as PLSA, which are both designed for text data created by humans, can effectively handle the mapped NMR-data originating from natural products. Additionally, PLSA is able to find meaningful "topics" in the NMR-data.

In section 2.3, we introduced a method to directly compute similarity between pairs of 2D-NMR spectra. This method will be used in the experiments as a bottom line benchmark. The matching criterion used 2.3 in checks only local properties of the spectra. Therefore, the direct similarity function cannot account for typical chemical substructures described by typical constellations of multiple peaks. For that it would be necessary to check if several peaks are present at the same time. To capture those more abstract properties more sophisticated methods are needed.

3.2.1 Mapping

In this section, we propose different methods to map the peaks of an 2D-NMR spectrum from the continuous space of measurements to a discrete space of words. With the help of such a mapping, methods for text retrieval like PLSA can be directly applied. However, the quality of the similarity search depend on how the peaks are mapped to discrete words.

Like a 2D-NMR spectrum consists of a set of peaks, a document consists of many words. Typically, a document is modeled as a bag of words. Assuming a 2D-NMR spectrum can be transformed into a text-like object by mapping the continuous 2D peaks to discrete variables, a variety of text retrieval models can be applied. However, it is an open question, whether models designed for quite different data, namely texts created by humans, are effective on data that comes from naturally occurring compounds and thus does not exhibit human design patterns. For 2D-NMR spectra similarity search, it is not clear, what is the best way to map the peaks of a spectrum to discrete words. We develop methods for this task in the next subsections. That will enable us to tackle the question, whether methods like the vector space model or PLSA, which are designed for text data, remain effective for experimental data from natural products.

3.2.1.1 Grid-based Mapping

We introduce a simple grid-based method, on which we will build more sophisticated methods. A simple grid-based method is to partition each of the both axes of the two-dimensional peak space into intervals of same size. Thus, an equidistant grid is induced in the two-dimensional peak space and a peak is mapped to exactly one grid cell it belongs to. When a grid cell is identified by a discrete integer vector consisting of the cells coordinates, the mapping of a peak $x \in \mathbb{R}^2$ is formalized as

$$g(x) = (g_c(x.c), g_h(x.h)) \text{ with } g_c(x.c) = \left\lfloor \frac{x.c}{w_c} \right\rfloor,\ g_h(x.h) = \left\lfloor \frac{x.h}{w_h} \right\rfloor$$

The quantities w_c and w_h are the extensions of a cell in the respective dimensions, which are parameters of the mapping. The grid is centered at the origin of the peak space. The cells of the grid act as words. The vocabulary generated by the mapped peaks consists of those grid cells that contain at least one peak. Empty grid cells are not included in the vocabulary. A word consists of a two-dimensional discrete integer vector.

Unfortunately, the grid-based mapping has two disadvantages. First, close peaks may be mapped to different grid cells. This may lead to poor matching of related peaks in the discrete word space. Second, peaks of new query spectra would be ignored when they are mapped to grid cells not included in the vocabulary. Thus, some information from the query would not be used for the similarity search, which may weaken the performance.

3.2.1.2 Redundant Mappings

We propose three mappings which introduce certain redundancies by mapping a single peak to a set of grid cells. The redundancy in the new mappings shall compensate for the drawbacks of the simple grid-based mapping.

3.2.1.2.1 Shifted Grids The first disadvantage of the simple grid-based method is that peaks which are very close in the peak space may be mapped to different grid cells, because a cell border is between them. So proximity of peaks does not guaranty that they are mapped to the same discrete cell. We draw on the idea of average shifted histograms [121] to propose a new mapping method to overcome the disadvantage.

Instead of mapping a peak to a single grid cell, we propose to map it to a set of overlapping grid cells. This is achieved by several shifted grids of the same granularity. In addition to the base grid some grids are shifted into the three directions $(1,0)(0,1)(1,1)$. An illustration of the idea is sketched in figure 3.1. In figure 3.1, one grid is shifted in each of the directions by half of the extent of a cell. In general, there may be $k-1$ grids shifted by fractions of $1/k, 2/k, \ldots, k-1/k$ of the extent of a cell in each direction respectively. For the mapping of the peaks to words which consist of cells from the different grids, two additional dimensions are needed to dis-

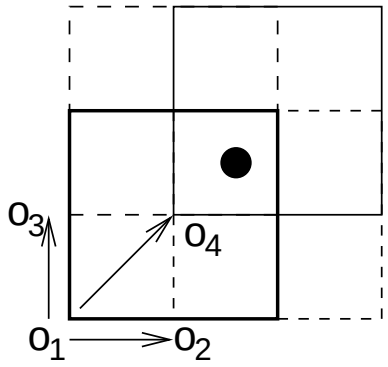


Figure 3.1: The four grids are marked as follows: base grid is bold, $(1,0)$, $(0,1)$ are dashed and $(1,1)$ is normal.

tinguish (a) the $k-1$ grids in each direction and (b) the directions themselves. The third coordinate represents the fraction by which a cell is shifted and the fourth one represents the directions by the following coding: value 0 is (0,0), 1 is (1,0), 2 is (0,1) and 3 is (1,1). So each peak is mapped to a finite set of four-dimensional integer vectors. The mapping of a peak $x \in \mathbb{R}^2$ is

$$\begin{aligned} s(x) \quad = \quad & \{(g_c(x.c), g_h(x.h), 0, 0)\} \cup \\ & \bigcup_{i=1}^{k-1} \Big\{ (g_c(x.c + {}^{i}\!/_{k} \cdot w_c), g_h(x.h), i, 1), \\ & \quad (g_c(x.c), g_h(x.h + {}^{i}\!/_{k} \cdot w_h), i, 2), \\ & \quad (g_c(x.c + {}^{i}\!/_{k} \cdot w_c), g_h(x.h + {}^{i}\!/_{k} \cdot w_h), i, 3) \Big\} \end{aligned}$$

Thus, a single peak is mapped to $3(k-1)+1$ words. A nice property of the mapping is that there exists at least one grid cell for every pair of matching peaks both peaks are mapped to.

3.2.1.2.2 Different Resolutions The second disadvantage of the simple grid-based mapping comes from the fact that empty grid cells (not occupied by at least one peak from the set of training spectra) do not contribute to the representation to be learned for similarity search. Therefore, peaks of new query spectra mapped to those empty cells are ignored. That effect can be diminished by making the grid cells larger. However, this is counterproductive for the precision of the similarity search due to the coarser resolution. Thus, there are two contradicting goals, namely (a) to have a fine resolution to handle subtle aspects in the data and (b) to cover at the same time the whole peak space by a coarse resolution grid so that no peaks of a new query spectrum have to be ignored.

Instead of finding a tradeoff for a single grid, both goals can be served by combining

simple grids with different resolutions. Given l different resolutions $\{(w_c^{(1)}, w_h^{(1)}), \ldots, (w_c^{(l)}, w_h^{(l)})\}$ a peak is mapped to l grid cells of different sizes. In order to distinguish between the different grids, an additional discrete dimension is needed. So the mapping function is

$$r(x) = \bigcup_{i=1}^{l} \{(g_c^{(i)}(x), g_h^{(i)}(x), i)\}$$

with $g_c^{(i)}$ and $g_h^{(i)}$ use $w_c^{(i)}$ and $w_h^{(i)}$ respectively. Note that a hierarchical quad-tree like partitioning is a special case of the proposed mapping function with $w_c^{(i)} = 2^{i-1} w_c$ and $w_h^{(i)} = 2^{i-1} w_h$.

3.2.1.2.3 Combining shifted Grids with different Resolutions Both methods are designed to compensate for different drawbacks of the simple grid mapping. Thus, it is natural to combine both mappings. The parameters of such a mapping are the number of shifts k, the number of different grid cell sizes l and the actual sizes $\{(w_c^{(1)}, w_h^{(1)}), \ldots, (w_c^{(l)}, w_h^{(l)})\}$. Beside the two coordinates for the grid cells, additional discrete dimensions are needed for the shift, the direction and the grid resolution. Using the definitions from above the mapping function of the combined mapping of a peak is

$$\begin{aligned} c(x) = \quad & \bigcup_{i=1}^{l} \left\{ \left(g_c^{(i)}(x.c), g_h^{(i)}(x.h), 0, 0, i\right) \right\} \cup \\ & \bigcup_{j=1}^{k-1} \Big\{ \left(g_c^{(i)}(x.c + j/k \cdot w_c^{(i)}), g_h^{(i)}(x.h), j, 1, i\right), \\ & \qquad \left(g_c^{(i)}(x.c), g_h^{(i)}(x.h + j/k \cdot w_h^{(i)}), j, 2, i\right), \\ & \qquad \left(g_c^{(i)}(x.c + j/k \cdot w_c^{(i)}), g_h^{(i)}(x.h + j/k \cdot w_h^{(i)}), j, 3, i\right) \Big\} \end{aligned}$$

Thus, a single peak is mapped to $l(3(k-1)+1)$ words.

3.2.2 Experiments

In this section we present the results of a comparison of the effectiveness of the mappings for similarity search, and mining aspects of 2D-NMR-data.

3.2.2.1 2D-NMR-Data

The substances included in the database are mostly secondary metabolites of plants and fungi. They cover a representative area of naturally occurring compounds and originate either from experiments or from simulations[2] based on the known structure

[2] ACD/2D NMR predictor, version 7.08, http://www.acdlabs.com/

Group	#Spectra	#Peaks
Pregnans	11	17–26
Anthrquinones	8	3–6
Aconitanes	8	22–26
Triterpenes	17	24–31
Flavonoids	18	5–8
Isoflavonoids	16	5–7
Aflatoxins	8	8–10
Steroids	12	16–23
Cardenolides	15	18–25
Coumarins	19	3–8

Table 3.1: Groups with number of spectra and range of peaks

of the compound. The database includes about 587 spectra, each has about 3 to 35 peaks. The total number of peaks is 7029. Ten small groups of chemically similar compounds are included in the database for controlled experiments. The groups with the number of spectra and number of peaks are listed in table 3.1. The peak space with all peaks in the database is shown in figure 3.2. Two groups, steroids and flavonoids, are selected as examples and shown with their peak distribution within figure 3.2.

Natural steroids occur in animals, plants and fungi. They are vitamins, hormones or cardioactive poisons like digitalis or oleander. The steroids in the database ar mostly hormones like androgens and estrogens. Flavonoids are aromatic substances (rings). Some flavonoids decrease vascular permeability or possess antioxidant activity which can have an anticarcinogenic effect.

3.2.2.2 Performance Evaluation

The different methods for similarity search of 2D-NMR-spectra are compared using recall-precision curves [94]. The search quality is high, when both – recall and precision – are high. Therefore, the upper curves are the best.

First, a series of experiments is conducted using our proposed mapping functions in combination with the vector space model. Each spectrum from the ten groups is used as a query while the rest of the respective group should be found as answers. The plots in figure 3.3 and 3.4 show averages over all queries. The results for the simple grid-based mapping are shown in figure 3.3a. The sizes of the grid cells are varied over $w_c = 4, 6, 8, 10$ and $w_h = 0.4, 0.6, 0.8, 1.0$ respectively. Small sizes give the best results.

The use of shifted grids improves the performance substantially over simple grids, as shown in figure 3.3b,c. The plots show the experiments for $k = 2, 3$. The results for $k = 2$ and $k = 3$ are almost identical. However, the vocabulary for $k = 2$ is much smaller. In practise, the smaller model with $k = 2$ shifts is favored.

Also, the mapping based on grids with different grid cell sizes are assessed. Due to

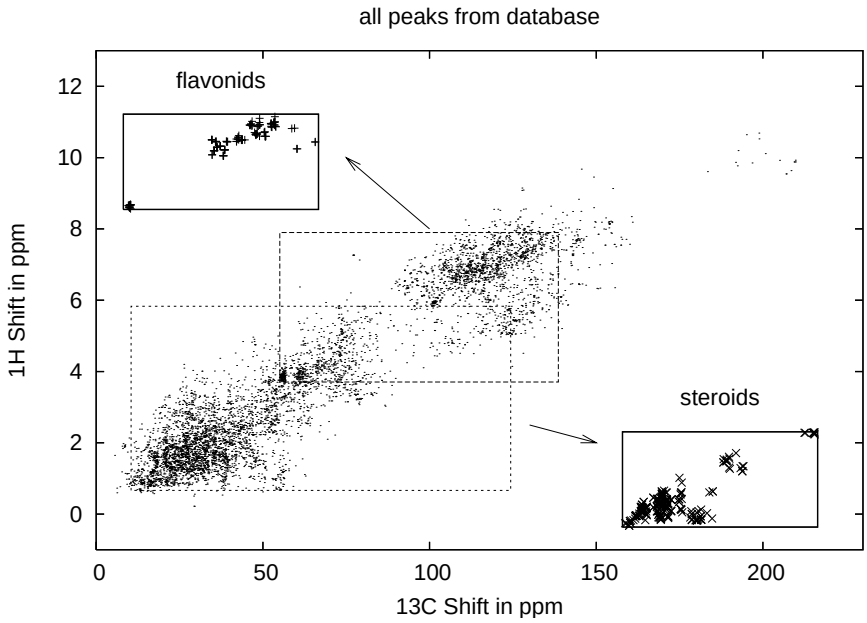


Figure 3.2: Distribution of the peaks of all spectra with the distribution within the groups of flavonoids and steroids.

lack of space, only the results from combinations of $w_c^{(1)} = 4, w_h^{(1)} = 0.4$ with other sizes are reported, because those performed best among all combinations. Figure 3.3d shows that also the mapping based on different grid cell sizes outperforms the simple grid-based mapping. But the improvement is not as much as for shifted grids. The set of resolutions $\{(w_c^{(1)} = 4, w_h^{(1)} = 0.4), (w_c^{(2)} = 10, w_h^{(2)} = 1.0)\}$ performs best.

Further, experiments are performed with the combination of the previous two mappings, namely a combination of shifted grids with those of different resolutions. The performance results are shown in figure 3.3e which indicates that the best combination, namely the resolution set $\{(w_c^{(1)} = 4, w_h^{(1)} = 0.4), (w_c^{(2)} = 10, w_h^{(2)} = 1.0)\}$ with $k = 2$ shifts, outperforms both previous mappings. This is more clearly seen in figure 3.3f that compares the best performing settings from the above experiments.

Next, a series of similar experiments is conducted using our proposed mapping functions in combination with PLSA. Random initialization is used for the EM training algorithm. All curves are averages from cross validation over all groups. As PLSA is trained on the data beforehand, we used cross validation where the current query is not included in the training data. As the groups are very small, the leave-one-out cross validation scheme is employed. The results for PLSA are shown in figure 3.4a-f. PLSA requires to chose the number of hidden aspects. For the experiments reported so far, the PLSA model is used with 20 hidden aspects. Furthermore, different numbers of aspects are tested using the best combination of mappings. Figure 3.4g shows that the performance with 10 aspects drops a bit. The increase in the numbers of

aspects from 20 to 32 is only marginally reflected in increase of search performance. Therefore, $K = 20$ is a reasonable number of aspects for the given data.

In summery, the experiments with both text retrieval methods show, that the mappings based on shifted grids and those with different resolutions perform significantly better than the simple grid-based mapping. In both cases, the combination of shifted grids and grids with different resolutions is even better than the individual mappings. The comparison between PLSA and the vector space model (figure 3.4h) shows that both have similar performance for small recall but for large recall PLSA has a better precision.

Last, the direct similarity function is tested (figure 3.4i). The size of the matching neighborhood is varied over $\alpha = 4, 6, 8, 10$ and $\beta = 0.4, 0.6, 0.8, 1.0$ respectively. The search quality is quite low. In fact on average, it fails to deliver a spectrum from the answer set in the top ranks, which is indicated by the hill-like shape of the curves.

In conclusion, the results prove experimentally that the vector space model as well as the PLSA model, which are designed for text retrieval, are indeed effective for similarity search of 2D-NMR spectra from naturally occurring products.

3.2.2.3 Analysis of the latent Aspects

We analyzed the latent aspects learned by the PLSA model using the mapping based on the combination of shifted grids with different resolutions. The grid cells (words) with high probability for a given aspect are plotted together to describe the aspects meaning. Some aspects specialized in certain regions in the peak space that are typical for distinct molecule fragments like aromatic rings or alkane skeletons. However, also more subtle details of the data are captured by the model. For example, the main aspect for the group of flavonoids specializes not only in the region for aromatic rings that are the main part of flavonoids. It also includes a smaller region that indicates oxygen substitution. A closer inspection of the database revealed that indeed many of the included flavonoids do have several oxygen substitutes. The main aspect for flavonoids with the respective peak distribution of the flavonoid group is shown in figure 3.5a. We believe a detailed analysis of the aspects found by the model may help to investigate unknown structures of new substances when their NMR-spectra are included in the training set.

3.2.3 Summary

We proposed redundant mappings from continuous 2D-NMR spectra to discrete text-like data that can be processed by any text retrieval method. We demonstrated experimentally the effectiveness of the our mappings in combination with the vector space model and PLSA. Further analysis revealed that the aspects found by PLSA are chemically relevant. The study is a preliminary step towards analyzing large sets of 2D-NMR spectra that are currently collected by our collaboration partners at the Leibniz institute of plant biochemistry.

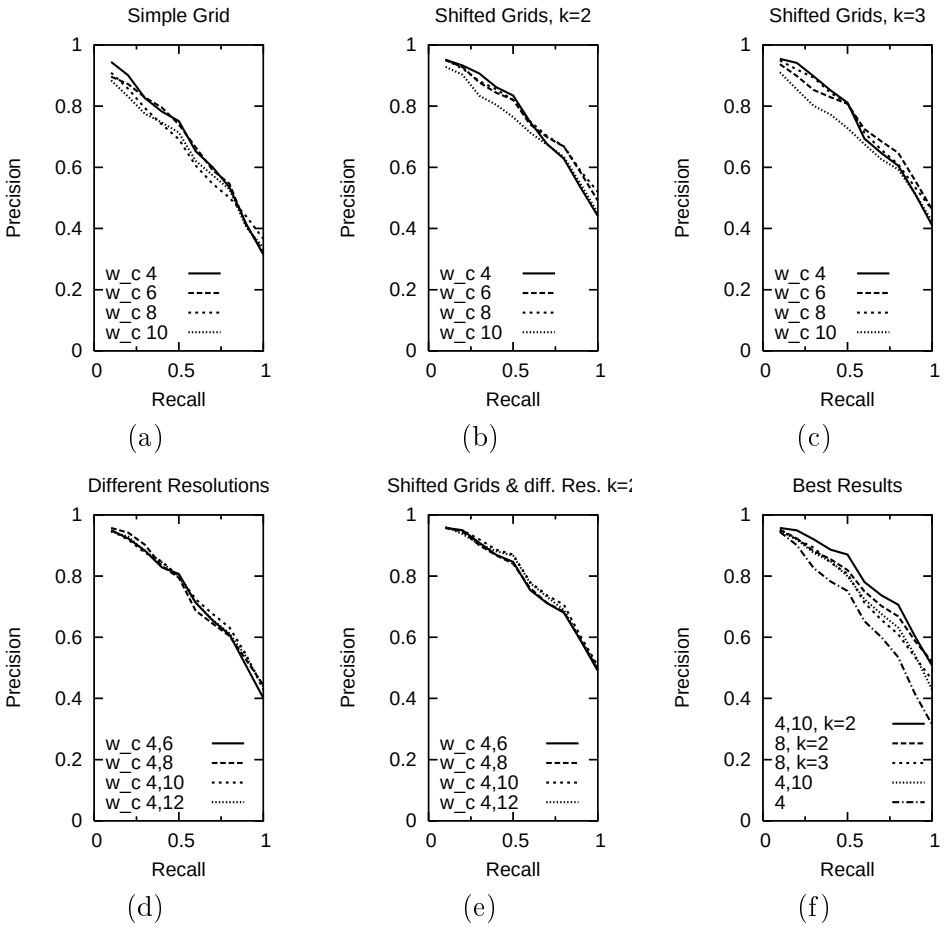


Figure 3.3: Average recall-precision curves using the vector space model

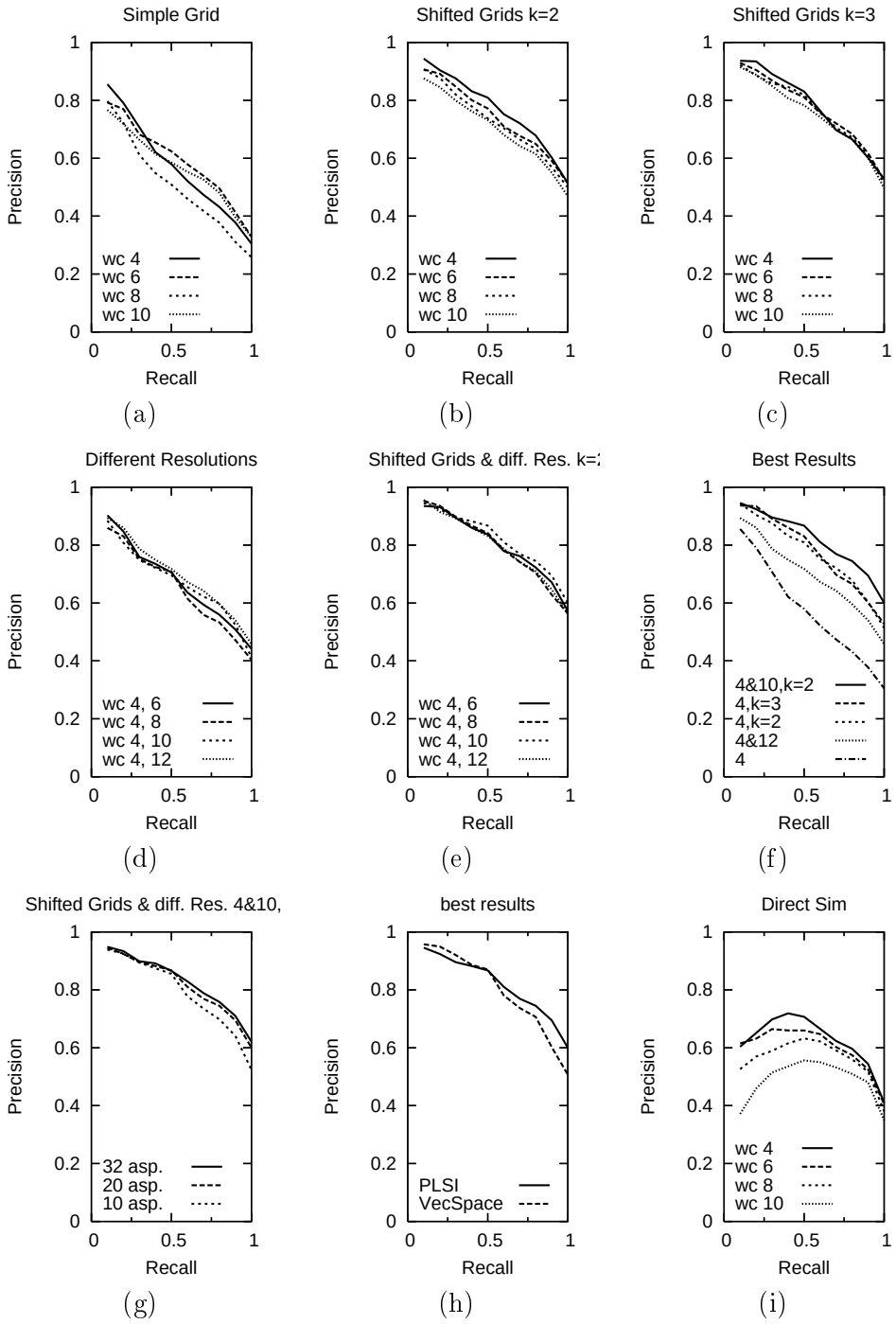
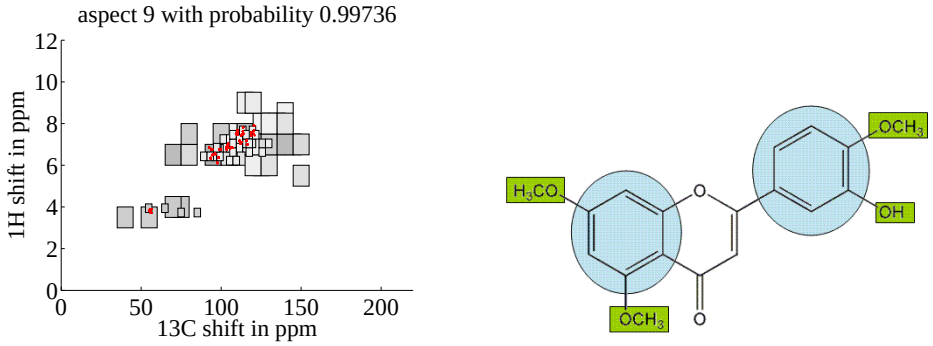


Figure 3.4: Average recall-precision curves from leave-one-out cross validation experiments with the PLSI model (a-g), best results of PLSI and vector space model (h) and results for the direct similarity (i).



(a) (b)

Figure 3.5: (a) Main aspect of the flavonoid group which includes the region of aromatic rings (upper right cluster) and the region for oxygen substituents (lower left cluster). The gray shades indicate the strength of the association between grid cell and aspect. (b) An example of an flavonoid (3'-Hydroxy-5,7,4'-trimethoxyflavone) where the aromatic rings and the oxygen substituents (methoxy groups in this case) are marked.

3.3 Latent Semantic Analysis of Proteomics Experiments

In order to assess inter-experiment similarity in an all-against-all comparison, latent semantic analysis (LSA; also referred to as latent semantic indexing, LSI) is employed. The main task is, to reduce the documents from a word-based representation to a topic-based representation, which reduces the influence of noise (random words) during the similarity computation between pairs of documents. Applied to the context of proteomics, experiments take the role of documents while peptides identified in one experiment (more precisely their amino acid sequence) act as terms. The algorithm reports a similarity score for each pair of experiments, based on the latent topics in peptide representation of the experiments.

3.3.1 Data Preparation

The HUPO PPP dataset was obtained from the PRIDE database[3], under accession numbers 4 to 98. These data sets are also accessible as PRIDE XML files via FTP[4].

All 95 Hupo PPP experiments and their corresponding sets of peptides are directly taken from the PRIDE database and give a term-document matrix with dimensions $25,052 \times 95$. Entries of the term-document matrix are weighted using *idf*. In contrast to IR-applications on natural language, no further pruning of unique terms was performed. A close look at the term-document matrix reveals the following differences compared to natural language datasets: while in a corpus of natural language text documents (for example the TREC Spanish AFP collection or the TREC Volume

[3] `http://www.ebi.ac.uk/pride`
[4] `ftp://ftp.ebi.ac.uk/pub/databases/pride`

3 corpus) the amount of unique words is below 2% [34], the Hupo PPP data set has 14,808 unique peptides (59%). This finding seems to contradict the intuitive assumption that proteomics experiments from the same tissue should yield highly similar results. The lack of reproducibility across proteomics experiments plays a considerable role in this divergence of the results [116]. This is illustrated for the HUPO PPP data by the fact that not even a single peptide is seen in every experiment. Moreover, only 37 peptides out of the 25,052 peptides are found in at least half of the experiments. In contrast, more than 70% of the reported proteins are only found in one or two experiments. This effect can be further explained by the wide array of techniques applied by the HUPO PPP contributors, with the purpose of enhancing coverage and allowing subsequent method evaluation [109]. Furthermore, a shotgun proteomics approach to analyze a complex mixture typically results in approximately 30% of all proteins identified by only a single peptide [106].

As a result, non-zero entries constitute less than 4% of the term-document matrix, which, although better than a typical natural language set, is still quite poor, especially considering the fact that we analyzed a small experiment corpus and that this data set describes the proteome of a single tissue, namely plasma. In order to analyze the ability of LSA to compensate for this sparseness of the term-document matrix, the similarity of pair of the 95 experiments is computed using different values for K and the standard cosine measure, which gives $95(95-1)/2 = 4,465$ similarity scores. The choice of K depends on the distribution of the singular values of the original document/term matrix. A rapid drop of the values in the sorted sequence of singular values (i.e. $\sigma_l - \sigma_{l+1}$ is large and σ_{l+1} is small) indicates that A_l is good approximation with low error. In our case, we used $K = 75$ for small degree of compression and $K = 15$ for high compression. For comparison, similarity scores are also directly computed from the vector space representation.

3.3.2 Application of LSA and PLSA to Proteomics Data

As expected, the distribution of the similarity scores obtained for the HUPO PPP experiments with the vector space model (VSM) shows a low overall similarity (93.5% pairs of experiments have a similarity less or equal than 0.1). Since the VSM approach to calculate experiment similarity suffers from the limitations of the exact term matching employed, the sparseness of the term-document matrix cannot be compensated.

Even the similarities for $K = 75$ show no drastic improvements; the similarity scores rise only slightly (88.5% pairs of experiments have a similarity less or equal than 0.1). However, with a strong dimensionality reduction ($K = 15$) of the HUPO PPP data, latent semantic relationships between terms as well as co-occurrences of peptides within the replicate experiments are amplified. Consequently, the inter-experiment similarities rise, resulting in the fact that now only 35% (in contrast to 93.5% for the VSM) of the experiments pairs have a similarity of 0.1 or less. This effectively compensates for the sparseness of the term-document matrix with all entries are non-zero. To validate the results and show that LSA indeed resulted

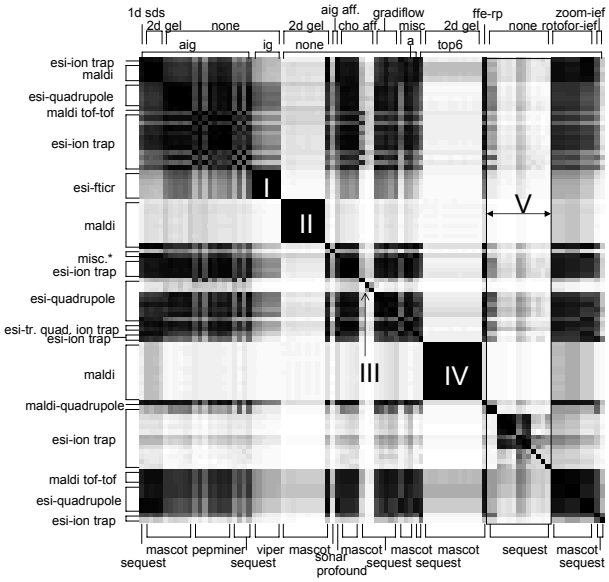


Figure 3.6: A visualization of the inter-experiment HUPO PPP similarity matrix, obtained from an LSA with $K = 15$. Dark indicates high similarity. The 95 experiments have been grouped by depletion technique, search engine, separation method and mass spectrometer (annotated above, to the left and below).

in meaningful experiment similarity, the obtained scores were visualized in a gray-scale map with white representing a score of 0, black a score of 1. The experiments are grouped by metadata, i.e. used technology (depletion step(s) applied; protein fractionation technique; search engine and finally mass spectrometer type). Since all 95 experiments originate from the same tissue, it is reasonable to expect LSA to amplify the intra-similarities within the same technology group.

3.3.2.1 Influence of Proteomics Technologies in the Hupo PPP dataset

A gray-scale map of the similarity scores between all 95 experiments, obtained after LSA with $K = 15$, is shown in figure 3.6. The experiments have been grouped according to four technologies that have been annotated above, below and to the left-hand side of the figure. Obviously, an experiment is identical to itself, which is why the top-left to lower-right diagonal is black. A great amount of experiments have a high similarity (dark areas) as expected when looking on one single tissue and using a low K, although some experiments are less similar to the other experiments than expected. Those experiments form distinct clusters, each with high internal

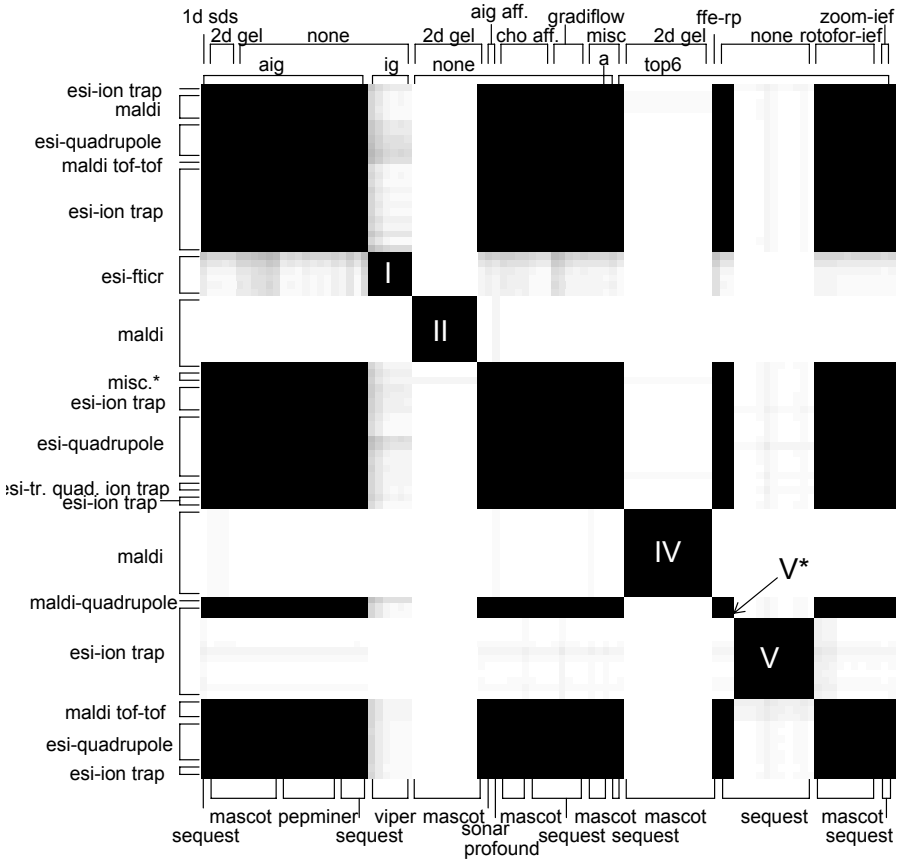


Figure 3.7: A visualization of the inter-experiment HUPO PPP similarity matrix, obtained from an PLSA with $K = 5$. Dark indicates high similarity. The 95 experiments have been grouped by depletion technique, search engine, separation method and mass spectrometer (annotated above, to the left and below).

similarity (I, II and IV). The first cluster (I) represents only ESI FT-ICR[5] experiments. This uniqueness in the instrument used, together with the use of proprietary VIPER search engine most probably contributes to the dissimilarity from the rest of the experiments. Cluster (II) is derived from a set of 2D-PAGE experiments, and these can be compared to cluster (IV), because both represent experiments performed by the same lab with the same technology. The only difference is the use of the top-6 protein depletion[6] on the biological sample in cluster (IV), while no depletion was employed in cluster (II). The very low similarity between these two clusters (nearly white overlap regions) shows that removal of the six most abundant proteins in plasma resulted in the detection of an almost completely different part of the plasma proteome. Three experiments (III) have very low similarity with the other experiments combined with a low similarity between each other. The reason for this could be the combination of a CHO-affinity (aldehyde affinity) fractionation and the SEQUEST search engine which no other experiment employed. As all three experiments are from the same laboratory (which only contributed these three experiments) and they have a rather low similarity among themselves, it seems plausible that these experiments are outliers and even might indicate suspect results. Another group of experiments also sticks out (band V). This group comprises experiments that employed a peptide shotgun approach (with no protein separation technique) on top-6 depleted samples. The shotgun experiments thus reveal very little similarity, both within the repeated experiments as well as compared to the rest of the experiments. The low similarities of the three clusters (I,II,IV) and the shotgun experiments (none for separation technique) with all other experiments indicates that they contribute unique peptide identifications (i.e.: they cannot be semantically connected to other peptide identifications). However, in contrast to the shotgun experiments, the 2D-PAGE cluster (II,IV) are strongly internally consistent, hinting at a high reproducibility of the method. The total number of peptide identifications for the HUPO PPP data set reveal that shotgun experiments contributed a major part of the overall unique peptide identifications. Finally, all observed clusters in this analysis derive from differences introduced by the various methodologies and technology platforms employed, rather than from differences between the samples which shows that a strong bias is introduced.

3.3.2.2 Sample Analysis

A second experimental setup was used to evaluate the performance of LSA to compute meaningful similarities on proteomics data: instead of the natural grouping of peptides by experiment, all peptides found by any number of experiments of the same biological sample (plasma or serum) and anticoagulation treatment (EDTA, citrate or heparin) where selected and grouped, which resulted in a $5 \times 25,052$ document term matrix. Again, a VSM approach is not able to produce meaningful similarities, whereas LSA with $K = 2$ (we would expect the term-document matrix to capture

[5] EletroSpray Ionization Fourier-Transform Ion Cyclotron Resonance instrument
[6] The six most frequent, known proteins are removed.

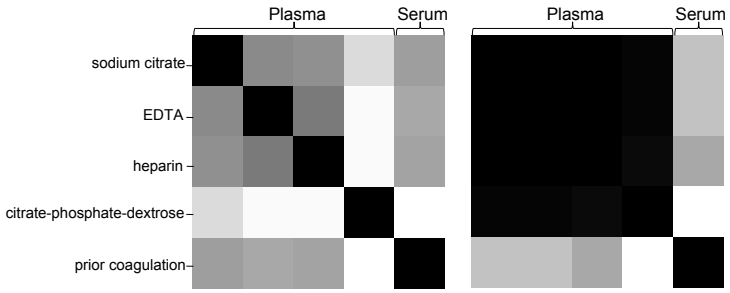


Figure 3.8: : HUPO sample similarity matrices, for the VSM on the left and LSA with $K = 2$ on the right. This grayscale map visualizes the differences in similarities resulting from a VSM and LSA ($k = 2$) analysis on all peptides from the HUPO PPP dataset. Peptides found by any of the 95 experiments are grouped by the biological sample type (plasma or serum) - annotated above - and the anticoagulation method used - annotated to the left.

two semantic topics: plasma and serum) yields easily interpretable results. Those similarity scores are visualized and annotated in figure 3.8: VSM (on the left) only shows the (trivial) high similarities of one sample/anticoagulation group with itself, whereas LSA is able to resolve the similarity of the 4 groups of peptides originating from the plasma samples. Obviously the very low similarity between plasma and serum is caused by the fact that the serum samples do not contain any proteins (and therefore peptides) associated with clotting, such as Fibrin.

3.3.2.3 Interpretation of the peptide-based LSA

Literature suggests that a comparison based on exact matching of peptide sequences dramatically underestimates the overlap between experiments [98]. Therefore, it is particularly interesting to see how LSA groups peptides to latent topics, which are the dimensions of the latent space.

This analysis can be carried out by studying the term representation $U_K\Sigma_K$. A k-means clustering [126, 130] performed on the rows of $U_K\Sigma_K$ allows detection of these related terms. Upon analysis of the resulting groups, two distinct patterns

emerge. First of all, LSA resolves peptides distinguished only by the occurrence of isobaric amino acids (e.g.: isoleucine/leucine). Since these amino acids are indistinguishable to the mass spectrometer, their substitution does not affect the semantic representation of the containing peptide sequence. Second, peptides that represent subsequences of longer peptides, either through missed cleavages (e.g.: YLGNATAIFFLPDEGK and YLGNATAIFFLPDEGKLQHLENELT), in-source decay or in vivo proteolytic degradation (e.g.: YLGNATAIFFLPDEGKLQHLENELT and YLGNATAIFFLPDEGKLQHLENELTHD) are all grouped together with the longer sequence. These two effects can be compared to synonyms in natural language.

3.3.2.4 Comparison with PLSA

We used also PLSA to determine the inter-experiment similarities of the Hupo PPP data set. This allows to validate the results obtained by LSA as well as to compare the performance of different models. Since PLSA cannot benefit from terms that appear only in a single document (unique peptides), the data set was reduced to peptides which occur at least in two experiments.

The number of latent topics for PLSA is set to $K = 5$, because the results of LSA showed 5 strong semantic topics, namely the two 2D-Page experiment-clusters, the Viper-search engine results, the shotgun-experiments, and the rest of the data set. The similarities between all pairs of experiments are visualized in a greyscale map in figure 3.7. The observed similarities cluster in an analogous way compared to LSA – the 2D-Page, viper search engine and shotgun experiments are dissimilar while the rest of the experiments have a high similarity among each other. This confirms the LSA results. A major improvement of PLSA is the clearer distinction between the clusters of experiments. Additionally, experiments within a cluster have a very high similarity in common (illustrated by the dark black coloring of the corresponding areas and absence of grey areas in figure 3.7). This is due to the facts that PLSA is more flexible in associating experiments with topics.

On the plasma proteome data, this results in a clear association of an experiment to exactly one latent topic. Thus, an experiment has a high probability (> 0.9) to belong to one of the five topics, while the probability for other topics (other entries in $p(z|d)$) is very low. Note, that this clear association to a single topic is generally not the case for PLSA. Usually, an experiment is represented by a specific mixture of topics. The observed clear distinction between clusters of experiments once more illustrates the complementary nature of the proteomics technologies employed.

The second difference between PLSA and LSA is shown by the results concerning the misclassified experiment annotated as III in figure 1: PLSA associates those experiments to the remainder of the other experiments. This might disprove the previously assumption made after LSA that these experiments could be outliers. Also, the pruning of unique peptides could be responsible for the high similarity of the previously misclassified experiments to the rest. Interestingly, in repeated experiments with different starting configurations for the EM-algorithm, those 3 experiments where sometimes separated. This leaves the issue as an open question.

The third difference is constituted by the shotgun experiments: with LSA those experiments had a low similarity between each other, resulting from the fact that each of those experiments contributed a substantial high amount of new peptide identifications compared to other experiments while having only a small set of peptides in common. The unique peptides lead to a low similarity between the shotgun experiments, but now with PLSA those experiments show high similarity. The reasons are two-fold. The first is that unique peptides were pruned. This lead to a shortened vector space representation with regards to the number of rows of the document term matrix resulting in a higher percentage of peptides the experiments have in common. The second cause is again based on the nature of the PLSA method. Those experiments are represented by similar mixture proportions of the latent topics, even though their representation in vector space differs.

Interestingly two shotgun experiments (annotated as V*) are not similar to the other shotgun experiments, in contrast they are similar to the majority of the remaining Hupo PPP experiments. There are several explanations for this finding. Most likely this is caused by the underlying separation technique employed in these two experiments. While all experiments from the shotgun-experiments cluster (annotated as V) employed an additional scx-separation step on the peptides after tryptic digestion, the two dissimilar experiments just relied on RP-HPLC as a protein separation method. This findings correlate to the results in [89]. However, another explanation could be the fact that those two experiments originate from a different lab than the rest of the shotgun experiments. Therefore it seems plausible, that internal lab-specific optimisations/variations of protocol-steps which are not annotated or even falsely annotated in the source database yielded different results.

The comparison of LSA and PLSA illustrates the advantages of PLSA on proteomics data. The more principled approach of PLSA based on the statistical latent class model has a sound statistical foundation. The higher flexibility allows PLSA to generate more realistic association of experiments to topics. Furthermore, the choice of an appropriate value for K, the number of latent space dimension, directly correlates to the semantic topics captured by PLSA, whereas with LSA the choice of K was based more or less on trial and error and ad hoc heuristics.

However, PLSA also has some disadvantages. Namely the training procedure of the EM-Step could get stuck in a local optimum of the likelihood function which leads to distributions for $p(z|d)$, $p(d)$ and $p(w|z)$ with less quality and therefore a model of low accuracy. In our experiments, the EM-step of PLSA had major difficulties on the peptide-experiment matrix without pruning the unique peptides. In combination with LSA nevertheless, PLSA can be an important tool for analyzing proteomics data.

3.3.3 Discussion and Conclusion

We have demonstrated a novel application of LSA by comparing peptide lists derived from many different proteomics experiments performed on the same tissue. By applying LSA to the data from the Hupo PPP study, we were able to show that this

method can handle the very diverse and heterogeneous data arising from proteomics experiments and compute meaningful similarities.

A large amount of experiments have a high similarity after LSA, which shows the strength of the method. However, some experimental setups, namely 2D-PAGE and shotgun approaches, strongly bias the set of observed peptides, which LSA cannot compensate for. Our results confirm visually, that if the goal of a project is to achieve maximal proteome coverage for a particular sample, shotgun proteomics experiments, repeated over multiple replicates achieve the most gain. 2D-PAGE analysis should not be disregarded as an analytical tool however, since it can complement a substantial fraction of unique identifications. Due to the high internal reproducibility of 2D-PAGE analyses as performed in the HUPO PPP, it seems that carrying out many replicates of this technology does not necessarily lead to a proportional increase in novel peptide identifications. In the specific case of plasma, the influence of various depletion techniques is also of interest. While methods employing top-6 depletion contributed more than 50% of the identifications, about 10% of all proteins were only found when no depletion was used at all.

The relatively simple task of comparing different sample types demonstrates that the fundamental difficulties arising from the origin of the data could be overcome through the utilization of an LSA analysis and its key principle of peptide/experiment association data representation in a lower dimensional 'latent space'. It is important to consider that the latent semantic analysis employed here greatly benefits from the large number of varying experimental repetitions on the same sample.

3.3.3.1 Interpretation of the semantic associations

An interesting finding is the ability of LSA to detect semantic relationships between apparently unrelated sets of peptide sequences, based solely on co-occurrences within experiments. We have found that at least some of these semantic links can be explained by underlying methodological or biological concepts and can be compared to synonyms found in natural language. The application of LSA to replicated shotgun experiments might help to alleviate one of the primary caveats of peptide-centric proteomics: the protein inference problem.

Since the semantic structures underlying protein lists (at least in part) represent entities of biological interest, the nature of the semantic relationships that occur at that level are also of considerable interest. Potential candidates of biological importance include protein complexes or protein components of the same pathway.

3.3.3.2 Future perspectives

It is clear from these findings that large collections of heterogeneous proteomics datasets can be mined relatively easily to obtain valuable information with LSA. The analysis carried out opens many paths for further investigations. By extending the analysis to include other tissue data sets (for instance the HUPO Brain Proteome Project (HUPO BPP) [62], and eventually any available proteomics data) and

by carefully choosing an appropriate value for k, the focus of investigation could be shifted from the fine-grained effects resulting from the application of different technology platforms, to the coarse-grained distinctions derived from differences in tissue type, disease state or developmental stage.

It is of significant interest to get a better understanding of the semantic similarities peptide share in the latent semantic space resulting from singular value decomposition. Other methods, especially probabilistic latent semantic analysis described in [75] that has a solid statistical foundation should be examined.

3.4 Summary

We presented new methods to transform 2D-NMR spectra into a bag of words representation consisting of discrete words. PLSA has been shown to improve the search quality on this data. Furthermore, topics found by PLSA have meaningful interpretations and reveal interesting statistical properties of the data. The result of this pilot study were encouraging. Subsequently, the compilation of a large collection of over 100.000 2D-NMR spectra of natural products was started. The collection was finished by the end of 2009 and is currently subject to initial analyses.

The latent semantic analysis of proteomics experiments also brought fruitful results. The analysis of relevant parts of the PRIDE database showed that LSA and PLSA can be used to define a meaningful similarity measure between proteomics experiments on the peptide level. Our first analysis shows a new and valuable way [32] to use data collections of proteomics experiments like PRIDE.

4 Probabilistic Modelling and Kernel Density Estimation

Many text-mining methods rely on probabilistic models, e.g. the multinomial mixture model, the Bernoulli Mixture model, probabilistic latent semantic analysis (PLSA) [75] and latent Dirichlet allocation (LDA) [18]. A probabilistic model defines a probability distribution $p(\vec{x}|\vec{\theta})$ that assigns a probability, a non-negative real number, to a given data object $\vec{x}$ by using values of a parameter vector $\vec{\theta}$. A typical way to learn the parameters of probabilistic models (parameter inference) is to find a parameter setting that maximizes some joint probability that is defined in terms of a set of observed data objects $\vec{X} = \{\vec{x}_1, \ldots, \vec{x}_N\}$, e.g. the likelihood $p(\vec{X}|\vec{\theta})$ or the posterior $p(\vec{\theta}|\vec{X})$. The material presented in this chapter is a revised version of our publication on kernel density based clustering [70] and Bayesian folding-in for PLSA [71].

All of the mentioned probabilistic text-mining models use hidden variables to represent internal model assumptions. Hidden variables can be seen as non-observable virtual data that help to rewrite the optimization function used for parameter learning in a simpler, mathematically equivalent form. The expectation maximization algorithm [36, 99] provides a general framework for this simplification of the optimization procedure and many inference methods for the mentioned models rely on it.

Another method to estimate a probability density from a set of data objects is kernel density estimation (KDE). KDE works when a data object is described by one or multiple continuous attributes that constitute a data point in a multi-dimensional space. KDE places a kernel function, which is a unimodal function with a unique maximum, at each data point. The estimated probability density is the normalized sum of all kernel functions and is defined over the whole multi-dimensional space that covers the data points. Such a probability density has in general more than one local maximum. A useful task is to compute all local maxima of the estimated density. A data point $\vec{x}$ is assigned to the local maximum to that a hill climbing procedure started at $\vec{x}$ does converge. This concept is used for cluster analysis: The local maxima define clusters to which data points are assigned by a hill climbing procedure.

A novel result presented in this chapter is a new, faster and more accurate hill climbing procedure. We derive this procedure by casting the task of hill climbing a kernel density estimate as an EM algorithm. Beside the newly found theoretical connection between KDE and EM theory, the new method offers dramatic speed-up for cluster analysis due to faster convergence of the hill climbing. Furthermore,

the connection to the EM algorithm breaks the way to apply a wealth of general optimization strategies for the EM algorithm to the specific problem setting of kernel density based clustering.

The use of the EM algorithm for hill climbing a kernel density estimate is not limited to cluster analysis only. The technique can be combined with probabilistic models for text mining. We present a novel use of a kernel density estimate as a prior distribution for the parameters that govern the hidden variables in the PLSA model. The use of prior distributions for parameters helps to avoid overfitting the model to the training data. The general idea is that the prior distribution penalizes parameter settings of the model that correspond to known pathological cases. On the mathematical side, the EM procedure for hill climbing the kernel density estimate is seamlessly combined with the EM algorithm to learn parameters of the PLSA model. This example shows how a new general class of prior distributions, namely kernel density estimates, can be used in probabilistic models.

In the remainder of the chapter, we first give a brief overview about the basics of the EM algorithm and kernel density estimation. In Section 4.2, we develop our new technique in the context of kernel density based clustering, show the new connection between KDE and EM theory and demonstrate experimentally the advantages of the new method. The usage of the new technique as a prior distribution in a text mining model is described in Section 4.3.

4.1 Underpinnings of Expectation Maximization Algorithm and Kernel Density Estimation

We give a brief overview about the expectation maximization algorithm and kernel density estimation. As our proof to show the convergence of the new hill climbing method for kernel density estimation relies on the reduction to an EM algorithm, we sketch here the idea of the proof of the EM algorithms convergence to a local maximum. We follow the proof described in [15], pages 450-455.

We assume that the goal of the EM algorithm is to find a parameter setting $\vec{\theta}$ that maximizes the likelihood $p(\vec{X}|\vec{\theta})$ for given training data $\vec{X} = \{\vec{x}_1, \ldots, \vec{x}_N\}$. It is often more convenient to maximize the log-likelihood instead of the likelihood. In case of mixture models with a set of hidden variables $\vec{Z}$, the log-likelihood takes the form $\ln p(\vec{X}|\vec{\theta}) = \ln \sum_{\vec{Z}} p(\vec{X}, \vec{Z}|\vec{\theta})$. The mixture models we a dealing with have only discrete hidden variables, therefore the likelihood can be written as a sum over all possible instances of $\vec{Z}$. Argumentation in case of continuous hidden variables is analogous with the sum replaced by integrals. Because of the sum inside the logarithm there is no closed-form solution to this maximization problem.

The idea of the EM algorithm is to rewrite the log-likelihood with respect to an arbitrary probability distribution $q(\vec{Z})$. We do not need to require $q(\vec{Z})$ to have a specific form, we just need that q is a probability distribution with $\sum_{\vec{Z}} q(\vec{Z}) = 1$.

Then the log-likelihood can be rewritten as

$$\ln p(\vec{X}|\vec{\theta}) = \mathcal{L}(q,\vec{\theta}) + \mathrm{KL}(q||p) \text{ with} \tag{4.1}$$

$$\mathcal{L}(q,\vec{\theta}) = \sum_{\vec{Z}} q(\vec{Z}) \ln \frac{p(\vec{X},\vec{Z}|\vec{\theta})}{q(\vec{Z})} \tag{4.2}$$

$$\mathrm{KL}(q||p) = -\sum_{\vec{Z}} q(\vec{Z}) \ln \frac{p(\vec{Z}|\vec{X},\vec{\theta})}{q(\vec{Z})} \tag{4.3}$$

The first term $\mathcal{L}(q,\vec{\theta})$ is a functional depending on the distribution $q(\vec{Z})$ and the parameter vector $\vec{\theta}$. The second term is the Kullback-Leibler (KL)-divergence between the distributions $q(\vec{Z})$ and the posterior distribution $p(\vec{Z}|\vec{X},\vec{\theta})$. The decomposition can be verified as follows:

$$\ln p(\vec{X}|\vec{\theta}) = \sum_{\vec{Z}} q(Z) \ln p(\vec{X}|\vec{\theta}) \tag{4.4}$$

$$= \sum_{\vec{Z}} q(Z) \ln p(\vec{X}|\vec{\theta}) + \sum_{\vec{Z}} q(\vec{Z}) \ln \frac{p(\vec{Z}|\vec{X},\vec{\theta})}{q(\vec{Z})} - \sum_{\vec{Z}} q(\vec{Z}) \ln \frac{p(\vec{Z}|\vec{X},\vec{\theta})}{q(\vec{Z})} \tag{4.5}$$

$$= \sum_{\vec{Z}} q(Z) \left[\ln p(\vec{X}|\vec{\theta}) + \ln \frac{p(\vec{Z}|\vec{X},\vec{\theta})}{q(\vec{Z})} \right] + \mathrm{KL}(q||p) \tag{4.6}$$

$$= \sum_{\vec{Z}} q(Z) \ln \frac{p(\vec{Z}|\vec{X},\vec{\theta}) \cdot p(\vec{X}|\vec{\theta})}{q(\vec{Z})} + \mathrm{KL}(q||p) \tag{4.7}$$

$$= \sum_{\vec{Z}} q(Z) \ln \frac{p(\vec{X},\vec{Z}|\vec{\theta})}{q(\vec{Z})} + \mathrm{KL}(q||p) \tag{4.8}$$

$$= \mathcal{L}(q,\vec{\theta}) + \mathrm{KL}(q||p) \tag{4.9}$$

The decomposition (4.2) is important because the KL-divergence has the property that it is always non-negative $\mathrm{KL}(q||p) \geq 0$. The equality to zero is realized if and only if the two distributions q and p are identical. Therefore, $\mathcal{L}(q,\vec{\theta})$ is a lower bound of the log-likelihood $\ln p(\vec{X}|\vec{\theta})$.

$$\mathcal{L}(q,\vec{\theta}) \leq \ln p(\vec{X}|\vec{\theta}) \tag{4.10}$$

The key idea of the EM algorithm is to maximize the lower bound instead of the log-likelihood. The maximization of $\mathcal{L}(q,\vec{\theta})$ is the iterative two step procedure. After initialization of $\vec{\theta}$, the lower bound is maximized with respect to q in the first step. Second, the lower bound is maximized with respect to $\vec{\theta}$. The iteration runs until convergence. As the lower bound is increased in each step and the log-likelihood is a finite function, the procedure converges in a finite number of steps.

The maximization of $\mathcal{L}(q, \vec{\theta})$ with respect to q is done by minimizing the KL-divergence $\mathrm{KL}(q||p)$. For fixed parameters $\vec{\theta}^{old}$, the KL-divergence $\mathrm{KL}(q||p)$ takes its minimum, namely zero, when $q(\vec{Z}) = p(\vec{Z}|\vec{X}, \vec{\theta}^{old})$. In the second step, the lower bound is maximized with respect $\vec{\theta}$ while fixing $q(\vec{Z})$ to the posterior distribution derived in the previous step. Note that when assigning the posterior $p(\vec{Z}|\vec{X}, \vec{\theta}^{old})$ to $q(\vec{Z})$ the lower bound can be simplified as follows:

$$\mathcal{L}(q, \vec{\theta}) = \sum_{\vec{Z}} p(\vec{Z}|\vec{X}, \vec{\theta}^{old}) \ln p(\vec{X}, \vec{Z}|\vec{\theta}) - \sum_{\vec{Z}} p(\vec{Z}|\vec{X}, \vec{\theta}^{old}) \ln p(\vec{Z}|\vec{X}, \vec{\theta}^{old}) \tag{4.11}$$

$$= \mathbb{E}_{\vec{Z}}[\ln p(\vec{X}, \vec{Z}|\vec{\theta})] + \text{const} \tag{4.12}$$

The second term is constant with respect to $\vec{\theta}$ because it depend only on the previous parameter setting $\vec{\theta}^{old}$ through the posterior. The expectation of the complete data log-likelihood $\mathbb{E}_{\vec{Z}}[\ln p(\vec{X}, \vec{Z}|\vec{\theta})]$ with respect to the posterior $p(\vec{Z}|\vec{X}, \vec{\theta}^{old})$ is exactly the quantity we maximized in the M-step in the derivation of the EM-algorithm for PLSA in chapter 3.1.3.

So far, we concentrated on maximizing the log-likelihood $\ln p(\vec{X}|\vec{\theta})$ using the EM algorithm. The EM algorithm can be also used to maximize the log-posterior $\ln p(\vec{\theta}|\vec{X})$, which will be helpful when we show the use of KDE-priors in probabilistic text-mining models. Note that the posterior can be rewritten in terms of the likelihood $p(\vec{\theta}|\vec{X}) = p(\vec{X}|\vec{\theta}) \cdot p(\vec{\theta})/p(\vec{X})$ using the Bayesian rule. Therefore, the log-posterior can be written in terms of the lower bound and the KL-divergence:

$$\ln p(\vec{\theta}|\vec{X}) = \ln p(\vec{X}|\vec{\theta}) + \ln p(\vec{\theta}) - \ln p(\vec{X}) \tag{4.13}$$

$$= \mathcal{L}(q, \vec{\theta}) + \mathrm{KL}(q||p) + \ln p(\vec{\theta}) - \ln p(\vec{X}) \tag{4.14}$$

$$\geq \mathcal{L}(q, \vec{\theta}) + \ln p(\vec{\theta}) - \ln p(\vec{X}) \tag{4.15}$$

The optimization of the lower bound 4.15 of the log-posterior gives the same E-step as for the log-likelihood, because the terms $\ln p(\vec{\theta})$ and $\ln p(\vec{X})$ do not depend on $q(\vec{Z})$. The M-step is modified by the term of the prior $\ln p(\vec{\theta})$, which in case of a conjugated prior distribution is only a minor change in the mathematical formula. The term $\ln p(\vec{X})$ is constant with respect to both $q(\vec{Z})$ and $\vec{\theta}$ and therefore does not influence the maximization procedure. Thus, the EM-algorithm for finding parameters $\vec{\theta}$ of a probabilistic model by maximizing the posterior $\ln p(\vec{\theta}|\vec{X})$ differs not much from the EM-algorithm for maximizing the likelihood.

The kernel density framework estimates the probability density in a data space as a function of all data instances $\vec{x}_n \in \vec{X}$ with $n = 1, \ldots, N$. Now $\vec{X}$ is a finite subset of $\mathbb{R}^d, d \in \mathbb{N}$ of a d-dimensional continuous vector space. The influences of the data instances in the continuous data space are modeled via a simple kernel function, e.g. the Gaussian kernel

$$K(\vec{u}) = (2\pi)^{-\frac{d}{2}} \cdot \exp\left(-\frac{\vec{u}^2}{2}\right) \tag{4.16}$$

The sum of all kernels (with suitable normalization) gives an estimate of the proba-

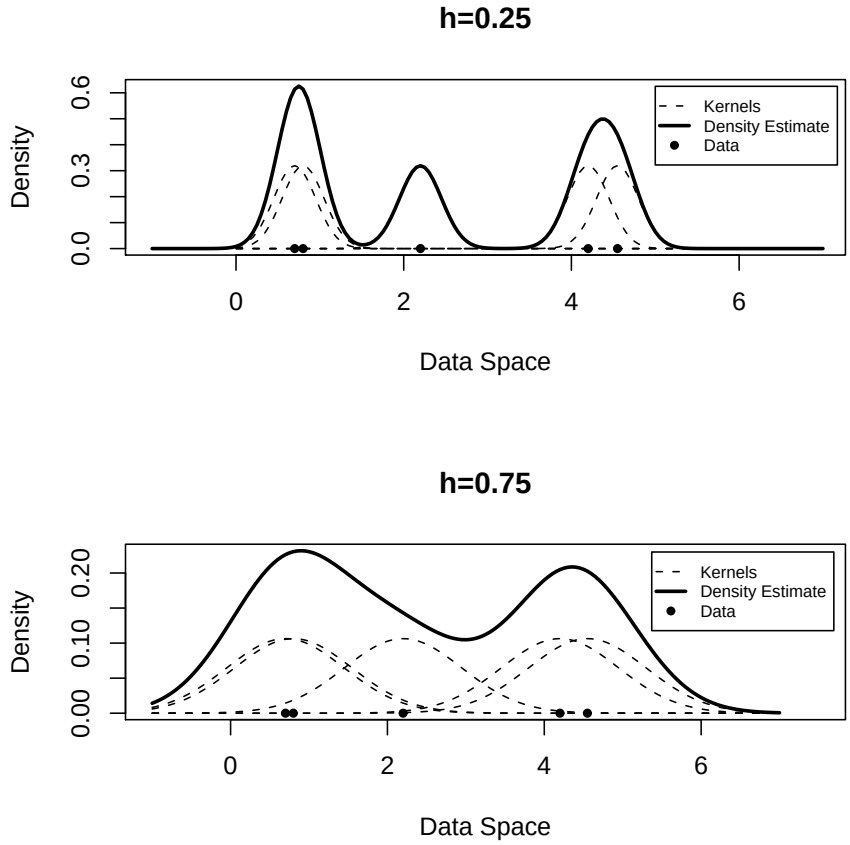


Figure 4.1: Kernel density estimate for one-dimensional data and different values for the smoothing parameter h.

bility density at any point $\vec{x}$ in the data space

$$\hat{p}(\vec{x}) = \frac{1}{(Nh^d)} \sum_{t=1}^{N} K\left(\vec{x}-\vec{x}_t/h\right) \tag{4.17}$$

The estimate $\hat{p}(\vec{x})$ enjoys all properties as the original kernel function, e.g. differentiability. The quantity $h > 0$ specifies to what degree the influence of a data point is smoothed over data space. When h is large, an instance stretches its influence up to more distant regions. When h is small, an instance effects only the local neighborhood. We illustrate the idea of kernel density estimation of one-dimensional data in figure 4.1.

4.2 KDE-Clustering as EM-Algorithm

Clustering can be formulated in many different ways. Non-parametric methods are well suited for exploring clusters, because no generative probabilistic model of the data needs to be assumed. Instead, the probability density in the data space is directly estimated from data instances. Kernel density estimation [122, 124] is a principled way of doing that task. There are several clustering algorithms, which exploit the adaptive nature of a kernel density estimate. Examples are the algorithms by Schnell [120] and Fukunaga [52] which use the gradient of the estimated density function. The algorithms are also described in the books by Bock [20] and Fukunaga [51] respectively. The DENCLUE framework for clustering [68,72] builds upon Schnells algorithm. There, clusters are defined by local maxima of the density estimate. Data points are assigned to local maxima by hill climbing. Those points which are assigned to the same local maximum are put into a single cluster.

However, the algorithms use directional information of the gradient only. The step size remains fixed throughout the hill climbing. This implies certain disadvantages, namely the hill climbing does not converges towards the local maximum, it just comes close, and the number of iteration steps may be large due to many unnecessary small steps in the beginning. The step size could be heuristically adjusted by probing the density function at several positions in the direction of the gradient. As the computation of the density function is relatively costly, such a method involves extra costs for step size adjustment, which are not guaranteed to be compensated by less iterations.

Our contribution is a new hill climbing method for kernel density estimates that bases on the EM algorithm. We develop it here in the context of Gaussian kernels as this is one of the most relevant kernels in the context of cluster analysis. However, our method is not limited to the use of this specific kernel as we demonstrate in section 4.3 The new hill climbing method adjusts the step size automatically at no additional costs and converges towards a local maximum. We prove this by casting the hill climbing as a special case of the expectation maximization algorithm. Depending on the convergence criterium, the new method needs less iterations as fixed step size methods. Since the new hill climbing can be seen as an EM algorithm, general acceleration methods for EM, like sparse EM [105] can be used as well. We also explore acceleration by sampling. Fast Density estimation [141] can be combined with our method as well but is not tested in this first study.

Other density based clustering methods beside DENCLUE, which would benefit from the new hill climbing, have been proposed by Herbin et al [65]. Variants of density based clustering are DBSCAN [118], OPTICS [7], and followup versions, which, however, do not use a probabilistic framework. This lack of foundation prevents the direct application of our new method there.

Related approaches include fuzzy c-means [13], which optimized the location of cluster centers and uses membership functions in a similar way as kernel functions are used by DENCLUE. A subtle difference between fuzzy c-means and DENCLUE is, that in c-means the membership grades of a point belonging to a cluster are nor-

malized, s.t. the weights of a single data point for all clusters sum to one. This additional restriction makes the clusters competing for data points. DENCLUE does not have such restriction. The mountain method [138] also uses similar membership grades as c-means. It finds clusters by first discretizing the data space into a grid, calculates for all grid vertices the mountain function (which is comparable to the density up to normalization) and determines the grid vertex with the maximal mountain function as the center of the dominant cluster. After effects of the dominant cluster on the mountain function are removed, the second dominant cluster is found. The method iterates until the heights of the clusters drop below a predefined percentage of the dominant cluster. As the number of grid vertices grow exponentially in high dimensional data spaces, the method is limited to low dimensional data. Niche clustering [104] uses a non-normalized density function as fitness function for prototype-based clustering in a genetic algorithm. Data points with high density (larger than a threshold) are seen as core points, which are used to estimate scale parameters similar to the smoothing parameter h introduced in the next section.

The rest of the section is structured as follows. In section 4.2.1, we briefly introduce the old DENCLUE framework and in section 4.2.2 we propose our new improvements for that framework. In section 4.2.3, we compare the old and the new hill climbing experimentally.

4.2.1 DENCLUE 1.0 framework for clustering

The DENCLUE framework [72] builds on non-parametric methods, namely kernel density estimation. Non-parametric methods are not looking for optimal parameters of some model, but estimate desired quantities like the probability density of the data directly from the data instances. This allows a more direct definition of a clustering in contrast to parametric methods, where a clustering corresponds to an optimal parameter setting of some high-dimensional function. In the DENCLUE framework, the probability density in the data space is estimated as a kernel density estimate $\hat{p}(\vec{x})$ (see 4.1). A clustering in the DENCLUE framework is defined by the local maxima of the estimated kernel density function. Data points are assigned to local maxima by a hill climbing function. A hill-climbing procedure is started for each data instance. After the procedure found a local maximum, the instance is assigned to this local maximum. In case of Gaussian kernels, the hill climbing is guided by the gradient of $\hat{p}(\vec{x})$, which takes the form

$$\nabla\hat{p}(\vec{x}) = \frac{1}{h^{d+2}N}\sum_{t=1}^{N} K\left(\frac{\vec{x}-\vec{x}_t}{h}\right)\cdot(\vec{x}_t-\vec{x}). \tag{4.18}$$

The hill climbing procedure starts at a data point and iterates until the density does not grow anymore. The update formula of the iteration to proceed from $\vec{x}^{(l)}$ to $\vec{x}^{(l+1)}$ is

$$\vec{x}^{(l+1)} = \vec{x}^{(l)} + \delta\frac{\nabla\hat{p}(\vec{x}^{(l)})}{\|\nabla\hat{p}(\vec{x}^{(l)})\|_2}. \tag{4.19}$$

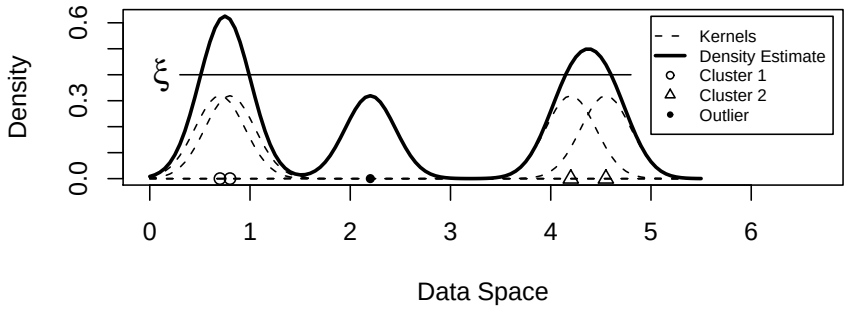


Figure 4.2: Example of a DENCLUE clustering based on a kernel density estimate and a noise threshold ξ.

The step size δ is a small positive number. In the end, those end points of the hill climbing iteration, which are closer than 2δ are considered, to belong to the same local maximum. Instances, which are assigned to the same local maximum, are put into the same cluster.

A practical problem of gradient based hill climbing in general is the adaptation of the step size. In other words, how far to follow the direction of the gradient? There are several general heuristics for this problem. All of them need to calculate $\hat{p}(\vec{x})$ several times to decide a suitable step size.

In the presence of random noise in the data, the DENCLUE framework provides an additional parameter $\xi > 0$ that defines all points assigned to local maxima $\vec{\hat{x}}$ with $\hat{p}(\vec{\hat{x}}) < \xi$ as outliers. Figure 4.2 sketches the idea of a DENCLUE clustering.

4.2.2 DENCLUE 2.0

In this section, we propose significant improvements of the DENCLUE 1.0 framework for Gaussian kernels. Since the choice of the kernel type does not have large effects on the results in the typical case, the restriction on Gaussian kernels is not very serious. First, we introduce a new hill climbing procedure for Gaussian kernels, which adjust the step size automatically at no extra costs. The new method does really converge towards a local maximum. We prove this property by casting the hill climbing procedure as an instance of the expectation maximization algorithm. Last, we propose sampling based methods to accelerate the computation of the kernel density estimate.

4.2.2.1 Fast Hill Climbing

The goal of a hill climbing procedure is to maximize the density $\hat{p}(\vec{x})$. An alternative approach to gradient based hill climbing is to set the first derivative of $\hat{p}(\vec{x})$ to zero and solve for $\vec{x}$. Setting (4.18) to zero and rearranging we get

$$\vec{x} = \frac{\sum_{t=1}^{N} K\left(\frac{\vec{x}-\vec{x}_t}{h}\right)\vec{x}_t}{\sum_{t=1}^{N} K\left(\frac{\vec{x}-\vec{x}_t}{h}\right)} \tag{4.20}$$

Obviously, this is not a solution for $\vec{x}$, since the vector is still involved into the righthand side. Since $\vec{x}$ influences the righthand side only through the kernel, the idea is to compute the kernel for some fixed $\vec{x}$ and update the vector on the lefthand side according to formula (4.20). This give a new iterative procedure with the update formula

$$\vec{x}^{(l+1)} = \frac{\sum_{t=1}^{N} K\left(\frac{\vec{x}^{(l)}-\vec{x}_t}{h}\right)\vec{x}_t}{\sum_{t=1}^{N} K\left(\frac{\vec{x}^{(l)}-\vec{x}_t}{h}\right)} \tag{4.21}$$

The update formula can be interpreted as a normalized and weighted average of the data points and the weights of the data points depend on the influence of their kernels on the current $\vec{x}^{(l)}$. In order to see that the new update formula makes sense it is interesting to look at the special case $N = 1$. In that case, the estimated density function consists just of a single kernel and the iteration jumps after one step to $\vec{x}^1$, which is the maximum.

The behavior of DENCLUEs 1.0 hill climbing and the new hill climbing procedure is illustrated in figure 4.3. The figure shows that the step size of the new procedure is adjusted to the shape of the density function. On the other hand, an iteration of the new procedure has the same computational costs as one of the old gradient based hill climbing. So, adjusting the step size comes at no additional costs. Another difference is, that the hill climbing of the new method really converges towards a local maximum, while the old method just comes close.

Since the new method does not need the step size parameter δ, the assignment of the instances to clusters is done in a new way. The problem is to define a heuristic, which automatically adjusts to the scale of distance between the converged points.

A hill climbing is started at each data point $\vec{x}_t \in X$ and iterates until the density does not change much, i.e. $[\hat{f}(\vec{x}_t^{(l)}) - \hat{f}(\vec{x}_t^{(l-1)})]/\hat{f}(\vec{x}_t^{(l)}) \leq \epsilon$. An end point reached by the hill climbing is denoted by $\vec{x}_t^* = \vec{x}_t^{(l)}$ and the sum of the k last step sizes is $s_t = \sum_{i=1}^{k} \|\vec{x}_t^{(l-i+1)} - \vec{x}_t^{(l-i)}\|_2$. The integer k is parameter of the heuristic. We found that $k = 2$ worked well for all experiments. Note that the number of iterations may vary between the data points, however, we restricted the number of iterations to be larger than k. For appropriate $\epsilon > 0$, it is safe to assume that the end points $\vec{x}_t^*$ are close to the respective local maxima. Typically, the step sizes are strongly shrinking before the convergence criterium is met. Therefore, we assume that the true local maximum is within a ball around $\vec{x}_t^*$ of radius s_t. Thus, the points belonging to the same local maximum have end points $\vec{x}_t^*$ and $\vec{x}_{t'}^*$, which are closer than $s_t + s_{t'}$.

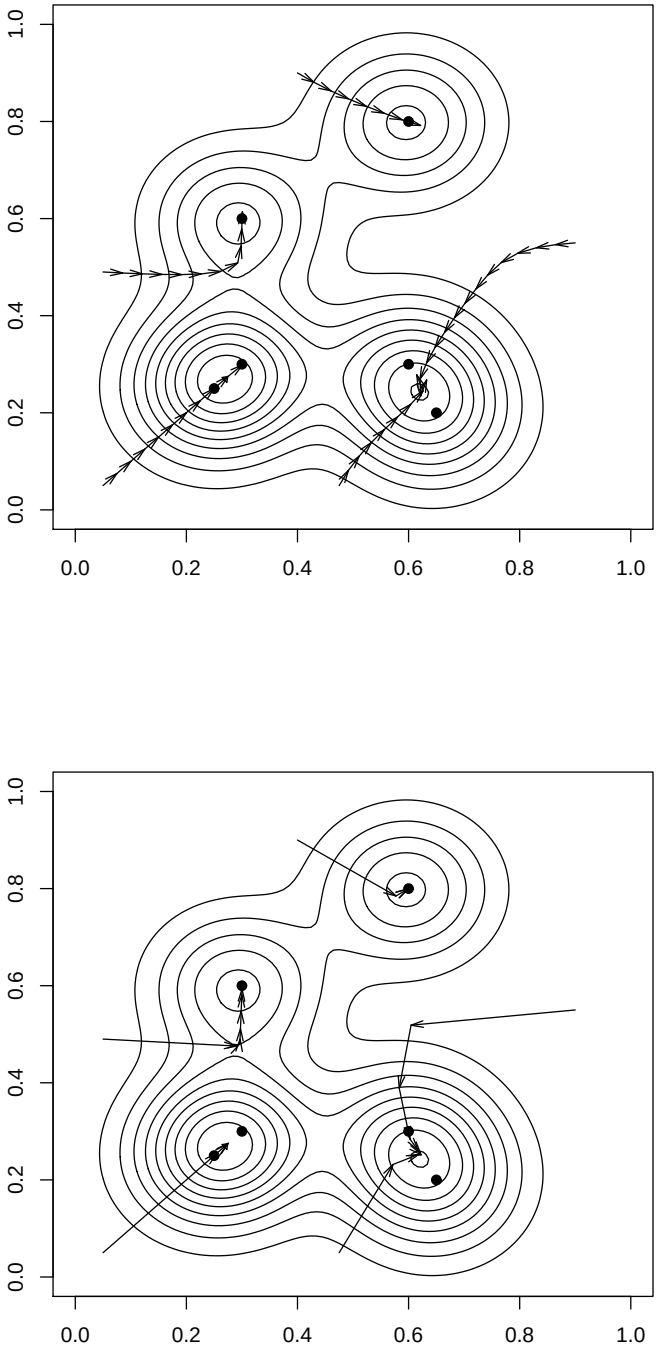


Figure 4.3: (top) Gradient hill climbing as used by DENCLUE 1.0, (bottom) Step size adjusting hill climbing used by DENCLUE 2.0.

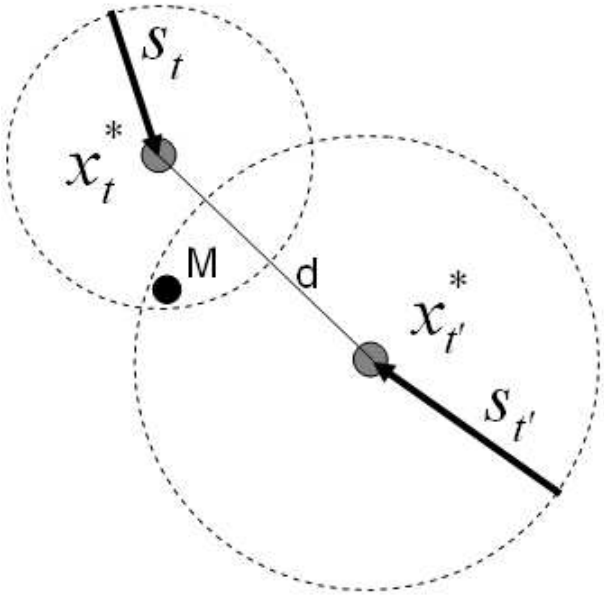


Figure 4.4: Assignment to a local maximum.

Figure 4.4 illustrates that case.

However, there might exists rare cases, when such an assignment is not unique. This happens when the following conditions hold for three end points $\vec{x}_t^*, \vec{x}_{t'}^*$ and $\vec{x}_{t''}^*$

$$\|\vec{x}_t^* - \vec{x}_{t'}^*\| \leq s_t + s_{t'} \text{ and} \tag{4.22}$$

$$\|\vec{x}_t^* - \vec{x}_{t''}^*\| \leq s_t + s_{t''} \text{ but not} \tag{4.23}$$

$$\|\vec{x}_{t'}^* - \vec{x}_{t''}^*\| \leq s_{t'} + s_{t''} \tag{4.24}$$

In order to solve the problem, the hill climbing is continued for all points, which are involved in such situations, until the convergence criterium is met for some smaller ϵ (a simple way to reduce ϵ is multiply it with a constant between zero and one). After convergence is reached again, the ambiguous cases are rechecked. The hill climbing is continued until all such cases are solved. Since further iterations causes the step sizes to shrink the procedure will stop at some point. The idea is illustrated in figure 4.5.

However, until now it is not clear why the new hill climbing procedure converges towards a local maximum. In the next section, we prove this claim.

4.2.2.2 Reduction to Expectation Maximization

We prove the convergence of the new hill climbing method by casting the maximization of the density function as a special case of the expectation maximization framework [36,99]. When using the Gaussian kernel we can rewrite the kernel density

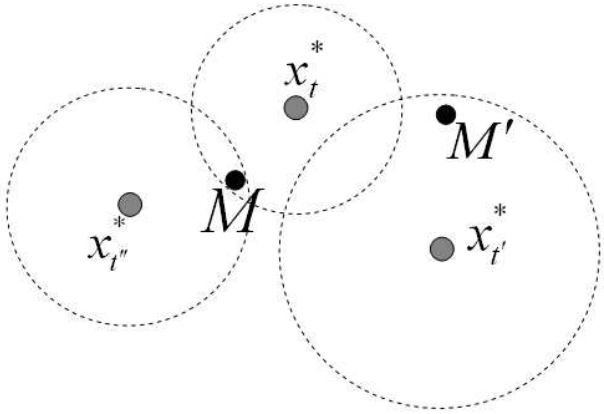


Figure 4.5: Ambiguous assignment: The points M and M' denote the true but unknown local maxima.

estimate $\hat{p}(\vec{x})$ in the form of a constrained mixture model with Gaussian components

$$p(\vec{x}|\vec{\mu},\sigma) = \sum_{n=1}^{N} \pi_n \mathcal{N}(\vec{x}|\vec{\mu}_n,\sigma) \tag{4.25}$$

and the constraints $\pi_n = 1/N$, $\vec{\mu}_n = \vec{x}_t$ ($\vec{\mu}$ denotes a vector consisting of all concatenated $\vec{\mu}_n$), and $\sigma = h$. We can think of $p(\vec{x}|\vec{\mu},\sigma)$ as a likelihood of a single data point $\vec{x}$ given the model defined by $\vec{\mu}$ and σ. Maximizing the log-likelihood $\ln p(\vec{x}|\vec{\mu},\sigma)$ with respect to $\vec{x}$ is not possible in a direct way, because of the sum inside the logarithm. Therefore, we resort to the EM framework by introducing a hidden variable $\vec{z} \in \{0,1\}^N$, a N-dimensional bit vector, with the constraints $\sum_{n=1}^{N} z_n = 1$ and

$$z_t = \begin{cases} 1 & \text{if the density at } \vec{x} \text{ is explained by } \mathcal{N}(\vec{x}|\vec{\mu}_t,\sigma) \text{ only} \\ 0 & \text{else} \end{cases}. \tag{4.26}$$

The complete log-likelihood is

$$\ln p(\vec{x},\vec{z}|\vec{\mu},\sigma) = \ln p(\vec{x}|\vec{z},\vec{\mu},\sigma) p(\vec{z}) \text{ with} \tag{4.27}$$

$$p(\vec{z}) = \prod_{n=1}^{N} \pi_t^{z_n} \text{ and} \tag{4.28}$$

$$p(\vec{x}|\vec{z},\vec{\mu},\sigma) = \prod_{n=1}^{N} \mathcal{N}(\vec{x}|\vec{\mu}_n,\sigma)^{z_n} \tag{4.29}$$

In contrast to generative models, which use EM to determine parameters of the model, we maximize the complete likelihood with respect to the vector $\vec{x}$. The EM-framework ensures that maximizing the expectation of complete log-likelihood

maximizes the original log-likelihood as well. Therefore, we define the quantity

$$\mathcal{Q}(\vec{x}|\vec{x}^{(l)}) = \mathbb{E}_{\vec{z}}[\log p(\vec{x}, \vec{z}|\vec{\mu}, \sigma)|\vec{\mu}, \sigma, \vec{x}^{(l)}] \tag{4.30}$$

In the E-step the expectation $\mathcal{Q}(\vec{x}|\vec{x}^{(l)})$ is computed with respect to the posterior distribution of $p(\vec{z}|\vec{x}^{(l)}, \vec{\mu}, \sigma)$. In the subsequent M-step the posterior distribution is fixed and $\mathcal{Q}(\vec{x}|\vec{x}^{(l)})$ is taken as a function of $\vec{x}$ and maximized. The E-step boils down to compute the posterior probability for the individuals components z_n:

$$\begin{aligned} E[z_n|\vec{\mu}, \sigma, \vec{x}^{(l)}] &= p(z_n = 1|\vec{x}^{(l)}, \vec{\mu}, \sigma) && (4.31)\\ &= \frac{p(\vec{x}^{(l)}|z_n = 1, \vec{\mu}, \sigma)p(z_n = 1|\vec{\mu}, \sigma)}{\sum_{n'=1}^{N} p(\vec{x}^{(l)}|z_{n'} = 1, \vec{\mu}, \sigma)p(z_{n'} = 1|\vec{\mu}, \sigma)} && (4.32)\\ &= \frac{1/N \cdot \mathcal{N}(\vec{x}^{(l)}|\vec{\mu}_n, \sigma)}{\sum_{n'=1}^{N} 1/N \cdot \mathcal{N}(\vec{x}^{(l)}|\vec{\mu}_{n'}, \sigma)} && (4.33)\\ &= \frac{1/N \cdot K(\frac{\vec{x}^{(l)} - \vec{x}_n}{h})}{\hat{p}(\vec{x}^{(l)})} =: \theta_n && (4.34) \end{aligned}$$

In the M-step, the posterior θ_t is held fixed, which yields

$$\mathcal{Q}(\vec{x}|\vec{x}^{(l)}) = \sum_{n=1}^{N} \theta_n [\ln 1/N + \ln \mathcal{N}(\vec{x}|\vec{\mu}_n, \sigma)] \tag{4.35}$$

Computing the derivative with respect to $\vec{x}$ and setting it to zero yields $\sum_{n=1}^{N} \theta_n \sigma^{-2}(\vec{x} - \vec{\mu}_n) = 0$ and thus

$$\vec{x}^{(l+1)} = \frac{\sum_{n=1}^{N} \theta_n \mu_n}{\sum_{n=1}^{N} \theta_n} = \frac{\sum_{n=1}^{N} K(\frac{\vec{x}^{(l)} - \vec{x}_n}{h})\vec{x}_n}{\sum_{n=1}^{N} K(\frac{\vec{x}^{(l)} - \vec{x}_n}{h})} \tag{4.36}$$

By starting the EM algorithm with $\vec{x}^{(0)} = \vec{x}_n$ the method performs an iterative hill climbing starting at data point $\vec{x}_n$.

4.2.2.3 Sampling based Acceleration

As the hill climbing procedure is a special case of the expectation maximization algorithm, we can employ different general acceleration techniques known for EM to speed up the the DENCLUE clustering algorithm.

Known methods for the EM algorithm try to reduce the number of iterations needed until convergence [99]. Since the number of iterations is typically quite low that kind of techniques yield no significant reduction for the clustering algorithm.

In order to speed up the clustering algorithm, the costs for the iterations itself should be reduced. One option is sparse EM [105], which still converges to the true local maxima. The idea is to freeze small posteriors for several iterations, so only the $p\%$ largest posteriors are updated in each iteration. As the hill climbing typically

needs only a few iterations we modify the hill climbing starting at the single point $\vec{x}^{(0)}$ as follows. All kernels $K(\frac{\vec{x}^{(0)}-\vec{x}_n}{h})$ are determined in the initial iteration and $\vec{x}^{(1)}$ is determined as before. Let be U the index set of the $p\%$ largest kernels and L the complement. Then, in the next iterations the update formula is modified to

$$\vec{x}^{(l+1)} = \frac{\sum_{n\in U} K(\frac{\vec{x}^{(l)}-\vec{x}_n}{h})\vec{x}_n + \sum_{n\in L} K(\frac{\vec{x}^{(0)}-\vec{x}_n}{h})\vec{x}_n}{\sum_{n\in U} K(\frac{\vec{x}^{(l)}-\vec{x}_n}{h}) + \sum_{n\in L} K(\frac{\vec{x}^{(0)}-\vec{x}_n}{h})} \tag{4.37}$$

The index set U and L can be computed by sorting. The disadvantage of the method is, that the first iteration is still the same as in the original EM.

The original hill climbing converges towards a true local maximum of the density function. However, we does not need the exact position of such a maximum. It is sufficient for the clustering algorithm, that all points of a cluster converge to the same local maximum, regardless where that location might be. In that light, it makes sense to simplify the original density function by reducing the data set to a set of $p\%$ representative points. That reduction can be done in many ways. We consider here random sampling and k-means. Thus, the number of points N is reduced to a much smaller number of representative points N', which are used to construct the density estimate.

Note that random sampling has much smaller costs as k-means. We investigate in the experimental section, whether the additional costs by k-means pay off by less needed iterations or by cluster quality.

4.2.3 Experimental Evaluation

We compared the new step size adjusting (SSA) hill climbing method with the old fixed step size hill climbing. We used synthetic data with normally distributed 16-dimensional clusters with uniformly distributed centers and approximately same size. Both methods are tuned to find the perfect clustering in the most efficient way. The total sum of numbers of iterations for the hill climbings of all data points is plotted versus the number of data points. SSA was run with different values for ϵ, which controls the convergence criterium of SSA. Figure 4.6 clearly shows that SSA ($\epsilon = 0.01$) needs only a fraction of the number of iterations of FS to achieve the same results. The costs per iterations are the same for both methods.

Next, we tested the influence of different sampling methods on the computational costs. Since the costs per iteration differ for sparse EM, we measure the costs in number of kernel computations versus sample size. Figure 4.7(left) shows that sparse EM is more expensive than random sampling and k-means based data reduction. The difference between the two latter methods is negligible, so the additional effort of k-means during the data reduction does not pay off in less computational costs during the hill climbing. For sample size 100% the methods converge to the original SSA hill climbing.

For random sampling, we tested sample size versus cluster quality measured by normalized mutual information (NMI is one if the perfect clustering is found). Figure

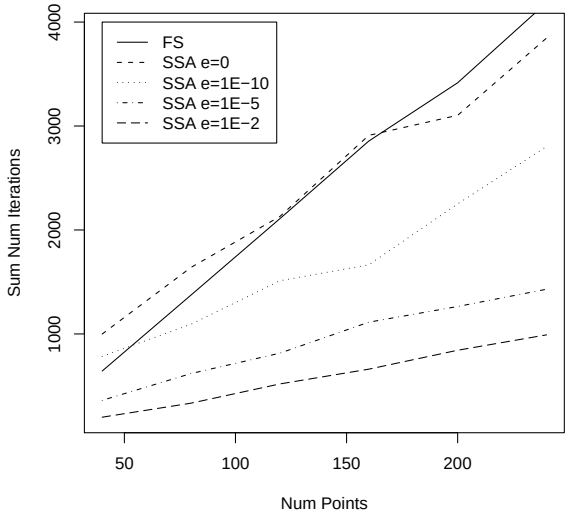


Figure 4.6: Number of data points versus the total sum of numbers of iterations.

4.7(right) shows that the decrease of cluster quality is not linear in sample size. So, a sample of 20% is still sufficient for a good clustering when the dimensionality is $d = 16$. Larger dimensionality requires larger samples as well as more smoothing (larger h), but the clustering can still be found.

In the last experiment, we compared SSA, its sampling variants, and k-means with the optimal k on various real data sets from the machine learning repository wrt. cluster quality. Table 4.1 shows average values of NMI with standard deviation for k-means and sampling, but not for SSA which is a deterministic algorithm.

SSA has better or comparable cluster quality as k-means. The sampling variants degrade with smaller sample sizes $(0.8, 0.4, 0.2)$, but k-means based data reduction

Table 4.1: NMI values for different data and methods, the first number in the three rightmost columns shows the sample size.

	k-means	**SSA**	**Random Sampling**	**Sparse EM**	**k-means Sampling**
iris	0.69±0.10	0.72	0.8: 0.66±0.05	0.8: 0.68±0.06	0.8: 0.67±0.06
			0.4: 0.63±0.05	0.4: 0.60±0.06	0.4: 0.65±0.07
			0.2: 0.63±0.06	0.2: 0.50±0.04	0.2: 0.64±0.07
ecoli	0.56±0.05	0.67	0.8: 0.65±0.02	0.8: 0.66±0.00	0.8: 0.65±0.02
			0.4: 0.62±0.06	0.4: 0.61±0.00	0.4: 0.65±0.04
			0.2: 0.59±0.06	0.2: 0.40±0.00	0.2: 0.65±0.03
wine	0.82±0.14	0.80	0.8: 0.71±0.06	0.8: 0.72±0.07	0.8: 0.70±0.11
			0.4: 0.63±0.10	0.4: 0.63±0.00	0.4: 0.70±0.05
			0.2: 0.55±0.15	0.2: 0.41±0.00	0.2: 0.58±0.21

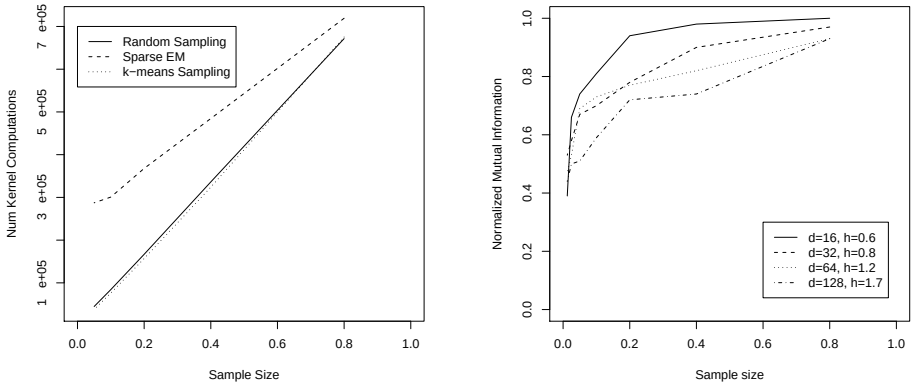


Figure 4.7: (left) Sample size versus number of kernel computations, (right) sample size versus cluster quality (normalized mutual information, NMI).

suffers much less from that effect. So, the additional effort of k-means based data reduction pays off in cluster quality.

In all experiments, the smoothing parameter h was tuned manually. Currently, we are working on methods to determine that parameter automatically. In conclusion, we proposed a new hill climbing method for kernel density functions, which really converges towards a local maximum and adjusts the step size automatically. Furthermore, our new technique can be combined with general probabilistic models as we show in the next section.

4.3 Bayesian Folding-In with KDE-Prior for PLSA

Probabilistic latent semantic analysis (PLSA) represents documents of a collection as mixture proportions of latent topics, which are learned from the collection by an expectation maximization (EM) algorithm. New documents or queries need to be folded into the latent topic space by a simplified version of the EM algorithm. During PLSA-Folding-in of a new document, the topic mixtures of the known documents are ignored. This may lead to a suboptimal model of the extended collection.

Our new approach incorporates the topic mixtures of the known documents in a Bayesian way during folding-in. That knowledge is modeled as prior distribution over the topic simplex using a kernel density estimate of Dirichlet kernels. We demonstrate the advantages of the new Bayesian folding-in in two scenarios using real text data.

The representation of documents as mixture proportions of latent topics is proven to be a useful tool for text mining. PLSA [74, 75] opened the way for probabilistic modeling of such representations. Applications and extensions of the PLSA model in the field of data mining include author-topic identification [131], document category detection [133], comparative and contextual textmining [101, 140], and web usage

mining [77].

A major drawback is, that PLSA is susceptible to overfitting [113]. The following research explored different Bayesian extensions of the PLSA model itself as well as alternative undirected models to overcome different drawbacks.

Bayesian extensions include latent Dirichlet allocation (LDA) [18], which models the topic mixture proportions of a document as hidden variables drawn from a single Dirichlet distribution, rather than as parameters of the model. PLSA has been shown to be a special case of LDA [58], when it uses an uninformative flat Dirichlet as prior. A more sophisticated variant are Dirichlet process priors [19]. Another Bayesian extension of PLSA are correlated topic models [16], which use a log-normal prior distribution. Such a distribution can capture correlations between topics, which cannot be expressed by a single Dirichlet. Dynamic topic models [17] extent this framework to model temporal changes in the latent topic space.

Alternative models from the class of undirected graphical models are undirected PLSA [136] and the Rate Adapting Poisson (RAP) model [54]. Those models are trained using contrastive divergence and avoid drawbacks of Bayesian directed models like the explaining away effect.

A general drawback of the proposed extensions and alternatives to PLSA is that the improvements come at the price of increased runtime costs for the inference algorithms, which hinders an applications to large data.

PLSA is not a fully generative model, so a special procedure called folding-in has to be used to determine topic mixture proportions of new documents or queries. Our approach extends PLSAs folding-in in a Bayesian way instead of extending the PLSA model itself. Instead of using a maximum likelihood estimator for folding-in, a maximum a posteriori estimator is employed, which uses a kernel density estimate as prior. The used kernel is a Dirichlet density. The advantage of a kernel density estimate as prior is, that only a very few model assumptions are made. Furthermore, such a prior can express correlations between topics similar to the correlated topic model.

Our contributions are

- we propose a new Bayesian model for folding-in
- a new inference technique is introduced which uses EM to maximize the posterior consisting of the word likelihood and a kernel density prior
- we propose new application scenarios for Bayesian folding-in

The reminder of the section is structured as follows, in section 4.3.1 we elaborate on the problems of PLSIs folding-in. In section 4.3.2, we introduce the new Bayesian model for folding-in and prove the convergence of the folding-in algorithm. Next, we present in section 4.3.3 two applications of how to use our new model. Last, we describe our experiments on real text data in section 4.3.4.

4.3.1 Problems of PLSIs Folding-In

Let D be a collection of documents, and each document is represented by a bag-of-words, which is a subset of the vocabulary W. The data set $\mathcal{D}$ modeled by PLSA [74, 75] is a set of pairs that consists of document- and word-identifiers: $\mathcal{D} \subseteq D \times W$. PLSA models the probability distribution of those pairs as a mixture of K latent classes $p(d, w) = p(d) \sum_{k=1}^{K} p(w|z_k = 1)p(z_k = 1|d)$. All probability distributions are multinomial distributions. The parameters of the model are the topic-word associations $\vec{\omega} = \{\omega_{wk} = p(w|z_k = 1) \colon w \in W \text{ and } k = 1, \ldots, K\}$ and the document-topic mixtures $\vec{\theta} = \{\theta_{kd} = p(z_k = 1|d) \colon d \in D \text{ and } k = 1, \ldots, K\}$, which are estimated by an EM-algorithm. A K-dimensional column $\vec{\theta}_d$ of the latter matrix denotes the mixture of topics for document d.

PLSA is not a generative model, thus the topic mixture proportions $p(\vec{z}|d_q)$ are not known for a new query document d_q that comes with its own set of document-word pairs $Q \subseteq \{d_q\} \times W$. The proposed folding-in procedure [74, 75] estimates those topic mixtures by running the original EM with fixed word-topic associations. Thus, the folding-in procedure reduces to (only the underlined probabilities are allowed to change during the algorithm):

$$\text{E-step: } p(z_k = 1|w, d_q) = \frac{p(w|z_k = 1)\underline{p(z_k = 1|d_q)}}{\sum_{k'=1}^{K} p(w|z_{k'} = 1)\underline{p(z_{k'} = 1|d_q)}} \tag{4.38}$$

$$\text{M-step: } p(z_k|d_q) = \frac{\sum_{w \in W} n(d_q, w)\underline{p(z_k = 1|w, d_q)}}{n(d_q)} \tag{4.39}$$

The quantities $n(d_q, w)$ and $n(d_q)$ are the number of occurrences of word w in d_q and the total number of words in d_q respectively.

Note, that the topic mixture for d_q is found independently from the mixtures of the other documents in the collection. The only influence comes through the fixed word-topic associations $p(w|\vec{z})$, which are involved in the righthand side of the E-step.

This can lead to problems in case of short queries, which do not contain a rich vocabulary as the documents in the collection. Since those queries have much fewer words with non-zero frequencies as the documents and the raw frequency counts are one in most cases, PLSAs folding-in tends to produce topic mixtures which are dominated by a single latent aspect.

The following example demonstrates this behavior. The data consists of $28 + 28 + 28 = 84$ documents, which are randomly sampled from three different newsgroups of the *20newsgroups collection*. Stopwords as well as infrequent words are eliminated and the other words are reduced to their stemmed form by Porters stemmer. PLSA run with $K = 3$ latent aspects maps the documents of the three different newsgroups (small circles, squares and triangles) into the latent space shown by the simplex in figure 4.8(a). The six big filled icons represent topic mixtures of documents from the respective groups found by PLSAs folding-in. The other six big empty icons simulate short queries each consists of four words sampled from one of the folded-in documents. Figures 4.8(b) and (c) show the likelihoods induced by the filled big

triangle nearest to the left margin and the big empty triangle nearest to the top margin respectively. While the long document induces a likelihood with a clear local maximum, the likelihood of the short version of that query has its maximum close to the upper corner of the simplex. Only the tempered version of EM [75] used for folding prevents that the short query is mapped to that border position. However, note the empty big circles and squares representing the other short queries in the left and right corners of the simplex in figure 4.8(a), where the tempered EM could not help. Such a corner position indicates that only a single latent aspect is present in such a query, which, however, in case of short queries is mainly caused by the small sample of words in the query. Therefore, PLSAs folding-in cannot account for alternative mappings of such a query, which might correspond to alternative semantic interpretations of the query.

4.3.2 Bayesian Folding-In

We present a new Bayesian way to estimate the mixture proportions of topics $\vec{\theta}_q$ for a new (query) document d_q with the document word pairs $Q \subseteq \{d_q\} \times W$. Instead of maximizing the likelihood $p(Q|\vec{\theta}_q, \vec{\theta}, \vec{\omega})$ of the set of query specific document-word pairs with respect to $\vec{\theta}_q$, the posterior $p(\vec{\theta}_q|Q, \vec{\theta}, \vec{\omega})$ is maximized. This maximum a posteriori (MAP) approach requires the definition of a prior distribution for the mixture of topics $\vec{\theta}_q$ of the new (query) document.

Because the topic mixtures of the documents in the collection shall have some influence on the topic mixture of the query, the prior is modeled as kernel density estimate using a Dirichlet distribution as kernel function. Kernel density estimation is a quite flexible method, which does not make strong assumptions about the distribution of the topic mixtures of the documents in the collections. Note that while the Dirichlet is a unimodal distribution, which assumes independence between the topics, a kernel density estimate using Dirichlet kernels can be multimodal and is able to capture correlations between topics.

In next the subsections, the details of the new folding-in procedure are explained and we prove the convergence of the method by casting the problem as a special case of the EM-algorithm.

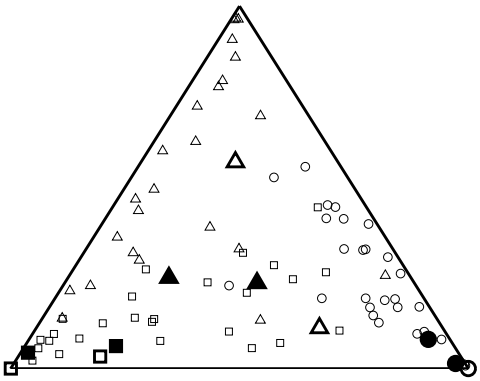
4.3.2.1 Model

We propose to derive the unknown topic mixture $\vec{\theta}_q$ of a new document using a MAP estimator. The following quantities are given for the MAP estimator, namely the set of query specific document-word pairs Q of the new document d_q, the topic mixtures of the documents in the training set $\vec{\theta}$, and the word topic associations $\vec{\omega}$.

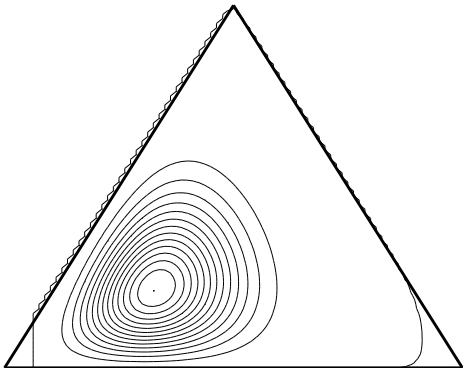
The maximum a posteriori estimator for Q is the topic mixture $\vec{\theta}_q^{MAP}$ that maximizes

$$p(\vec{\theta}_q|Q, \vec{\theta}, \vec{\omega}) \propto p(Q|\vec{\theta}_q, \vec{\theta}, \vec{\omega}) p(\vec{\theta}_q|\vec{\theta}, \vec{\omega}). \tag{4.40}$$

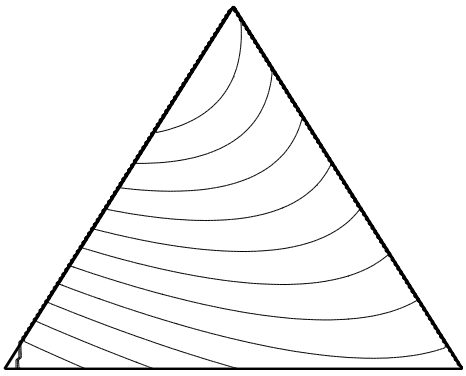
where $p(Q|\vec{\theta}_q, \vec{\theta}, \vec{\omega})$ is the likelihood of Q and $p(\vec{\theta}_q|\vec{\theta}, \vec{\omega})$ is the topic mixture prior. We



(a)



(b)



(c)

Figure 4.8: (a) topic mixtures (small icons) for documents learned by PLSA and for queries of different sizes (short: big empty icons, long: big filled icons) generated by PLSA folding-in, (b) typical likelihood in the topic simplex for a long query, (c) short query.

assume the query words to be independent, so the likelihood of Q can be decomposed as follows

$$p(Q|\vec{\theta}_q, \vec{\omega}) = \prod_{w \in W} p(w|\vec{\theta}_q, \vec{\omega})^{n(d_q,w)} \tag{4.41}$$

$$= \prod_{w \in W} \left(\sum_{k=1}^{K} p(w|z_k = 1) p(z_k = 1|d_q) \right)^{n(d_q,w)} \tag{4.42}$$

$$= \prod_{w \in W} \left(\sum_{k=1}^{K} \omega_{wk} \theta_{kq} \right)^{n(d_q,w)} \tag{4.43}$$

Since the likelihood of the query words does not depend on $\vec{\theta}$, for ease of writing that parameter is neglected in the list of given variables in $p(Q|\vec{\theta}_q, \vec{\omega})$.

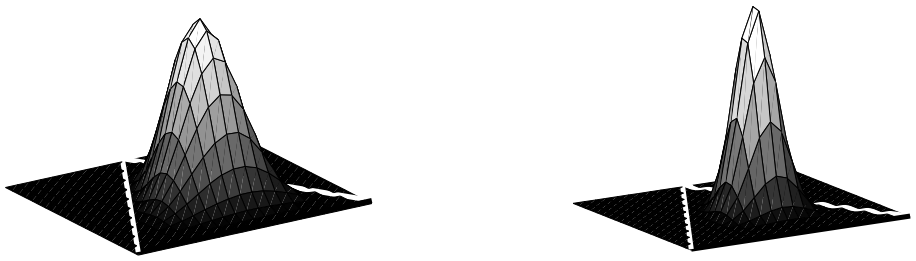
The prior is modeled by a kernel density estimate based on $\vec{\theta}$ using Dirichlet kernels. The topic mixtures are vectors, which have non-negative components and all components sum to one, $\forall d \in D\colon \sum_{k=1}^{K} \theta_{kd} = 1$. That means those vectors reside in a $K-1$-simplex which is embedded in the $\mathbb{R}^K$. The Dirichlet distribution is suitable for such data since the density function integrates to one over the simplex. The density of a Dirichlet is given by

$$Dir(\vec{x}|\vec{\alpha}) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_j)}{\prod_{k=1}^{K} \Gamma(\alpha_j)} \cdot \prod_{k=1}^{K} x_k^{\alpha_k - 1} \tag{4.44}$$

The $k$the coordinate of the mode of a Dirichlet is $\frac{\alpha_k - 1}{(\sum_{k'=1}^{K} \alpha_{k'}) - K}$ with $\alpha_k > 1$. The parameter vector $\vec{\alpha}$ controls both the location of the mode in the simplex and the sharpness of the mode. Examples of Dirichlet distributions with different parameter settings are shown in figure 4.9. Note that multiplying the parameter vector with a scalar larger one means to increase the sharpness of the mode but it does not change the location of the mode.

A kernel density estimate sums over the given data that are in our case the topic mixtures of the documents in the collection. The influence of each topic mixture θ_d is modeled by a single Dirichlet distribution, which has the mode located at θ_d. In order to control the sharpness of such a Dirichlet kernel the smoothing parameter h is introduced. Further, the function $\alpha(\vec{\theta}) = 1/h \cdot \vec{\theta} + \vec{1}$ is introduced, which takes a topic mixture vector and outputs the corresponding parameter vector of the Dirichlet, so that the mode is exactly at $\vec{\theta}$. Large h makes the kernels flat and stretches the influences of the individual topic mixtures over the simplex, whereas small values for h make the kernels like sharp peaks. Modeling the prior as kernel density estimate based on the topic mixtures of the documents in the collection gives the following formula:

$$p(\vec{\theta}_q|\vec{\theta}) = \frac{1}{|D|} \sum_{d \in D} \mathrm{Dir}(\vec{\theta}_q|\alpha(\vec{\theta}_d)) \tag{4.45}$$

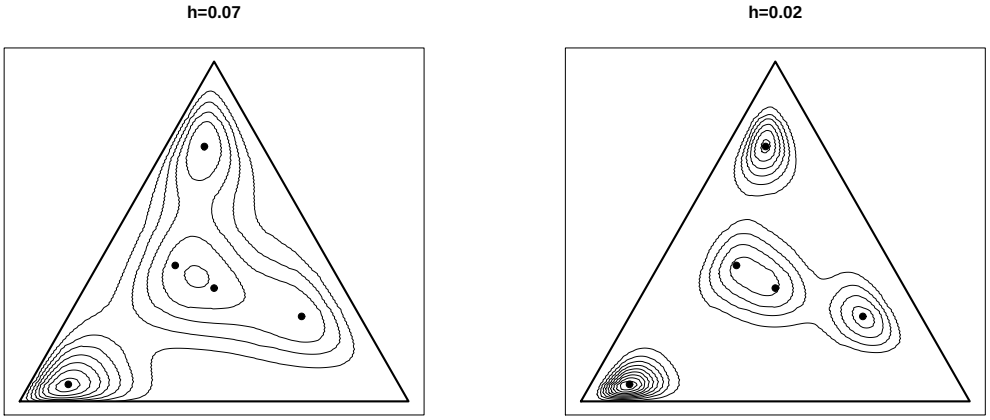


(a) (b)

Figure 4.9: (a) Example of a Dirichlet distribution over an $(3-1)$-simplex with $\vec{\alpha} = (4, 6, 3)$, (b) Another example with the same mode but different sharpness, $\vec{\alpha} = (8, 12, 6)$.

Since the prior does not depend on $\vec{\omega}$, for ease of writing that parameter is neglected in the list of given variables in $p(\vec{\theta}_q|\vec{\theta})$. Examples for priors in a $(3-1)$-simplex with different values for h are shown in figure 4.10.

The direct maximization of logarithm of the righthand side of (4.40) with respect to $\vec{\theta}_q$ is difficult due to the sums inside the logarithms. Therefore, hidden variables are introduced for both, the likelihood of Q and the prior distribution to make the maximization tractable with an EM algorithm. First, the likelihood of a single word $p(w|\vec{\theta}_q, \vec{\omega})$ can be seen as a mixture model of K topics. Thus, a hidden binary variable $\vec{y}_w \in \{0, 1\}^K$ following a 1-out-of-K scheme is introduced that indicates the particular topic z_k that is responsible for word $w \in W$. Second, the prior $p(\vec{\theta}_q|\vec{\theta})$ (eq. 4.45) also can be seen as a mixture model of $|D|$ Dirichlet components and equal component priors. Again, a hidden binary variable $\vec{x} \in \{0, 1\}^{|D|}$ following a 1-out-of-$|D|$ scheme is introduced, which indicates the particular Dirichlet component that is responsible for a specific setting of the topic mixture of the query document. All hidden variables are concatenated to the vectors $\vec{y}$ and $\vec{x}$ respectively. The details about the definitions of the hidden variables are explained in the next subsection. Instead of maximizing the posterior shown in (4.40), the expectation of logarithm of



(a) (b)

Figure 4.10: Examples of density estimates with Dirichlet kernels, (a) $h = 0.07$, (b) $h = 0.02$.

the posterior of the extended model is maximized.

$$\begin{aligned}
\mathbb{E}_{\vec{y},\vec{x}}[\ln(p(\vec{\theta}_q|Q,\vec{y},\vec{x},\vec{\omega}))] \propto & \mathbb{E}_{\vec{y},\vec{x}}\big[\ln(p(Q,\vec{y}|\vec{\theta}_q,\vec{\omega})p(\vec{\theta}_q,\vec{x}|\vec{\theta},\vec{\omega}))\big] \\
= & \mathbb{E}_{\vec{y}}\Big[\sum_{w\in W} n(d_q,w)\sum_{k=1}^{K} y_{wk}\big[\ln\omega_{wk}+\ln\theta_{kq}\big]\Big]+ \\
& \mathbb{E}_{\vec{x}}\Big[\sum_{d\in D} x_d\big[\ln\frac{1}{|D|}+\ln\mathrm{Dir}(\vec{\theta}_q|\vec{\alpha}(\vec{\theta}_d))\big]\Big]+c \\
= & \Big[\sum_{w\in W} n(d_q,w)\sum_{k=1}^{K} \mathbb{E}[y_{wk}]\big[\ln\omega_{wk}+\ln\theta_{kq}\big]\Big]+ \\
& \Big[\sum_{d\in D} \mathbb{E}[x_d]\big[\ln\frac{1}{|D|}+\ln\mathrm{Dir}(\vec{\theta}_q|\vec{\alpha}(\vec{\theta}_d))\big]\Big]+c \qquad (4.46)
\end{aligned}$$

The constant c comes from the normalization constant in equation (4.40). The maximization is done by an EM-algorithm, which starts with some settings for the wanted topic mixture of the query document $\vec{\theta}_q^{(0)}$ and iteratively computes posteriors for the hidden variables in the E-step and updates the topic mixture of the query document in the M-step. The posteriors for the hidden variables computed in the

E-step are given by the following formulas

$$\mathbb{E}[y_{wk}] = p(y_{wk} = 1 | w, \vec{\theta}_q^{(s)}, \vec{\omega}) = \frac{\omega_{wk} \cdot \theta_{kq}^{(s)}}{\sum_{k'=1}^{K} \omega_{wk'} \cdot \theta_{k'q}^{(s)}} = g_{wk} \tag{4.47}$$

$$\mathbb{E}[x_d] = p(z_d = 1 | \vec{\theta}_q^{(s)}, \vec{\theta}) = \frac{Dir\big(\vec{\theta}_q^{(s)} | \alpha(\vec{\theta}_d)\big)}{\sum_{d' \in D} Dir\big(\vec{\theta}_q^{(s)} | \alpha(\vec{\theta}_{d'})\big)} = h_d \tag{4.48}$$

In the M-step, the posteriors of the hidden variables are held fixed and plugged into (4.46). That equation is maximized with respect to $\vec{\theta}_q$ under the condition $\sum_{k=1}^{K} \theta_{kq} = 1$, which give the following update formula for the wanted topic mixture

$$\theta_{kq}^{(s+1)} = \frac{\sum_{w \in W} n(d_q, w) g_{wk} + 1/h \sum_{d \in D} h_d \theta_{kd}}{n(d_q) + 1/h} \tag{4.49}$$

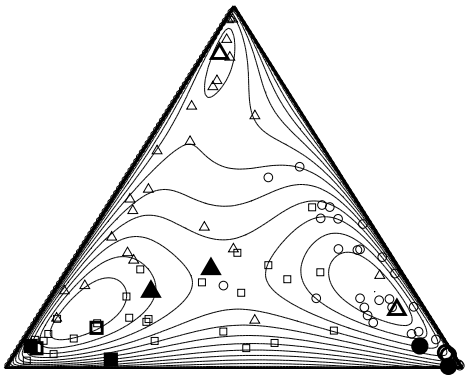
Note that Bayesian folding-in includes PLSAs folding-in as a special case, namely when $h = \infty$. In that case, the Dirichlet kernels become flat and the posteriors h_d of the prior vanish in the update formula 4.49. Therefore, the posteriors of the prior become irrelevant and the formulas (4.47) and (4.49) reduce to the equations (4.38) and (4.39) of PLSA folding-in respectively.

Figure 4.11 continues the small example from the previous section and shows the results for Bayesian folding-in using the same data as before. The contour lines in figure 4.11(a) show the prior, which is the same for all queries. Also note the multimodal posterior (figure 4.11c) for the short query (big empty triangle in the upper corner). The other less likely modes of that posterior may correspond to alternative semantic interpretations.

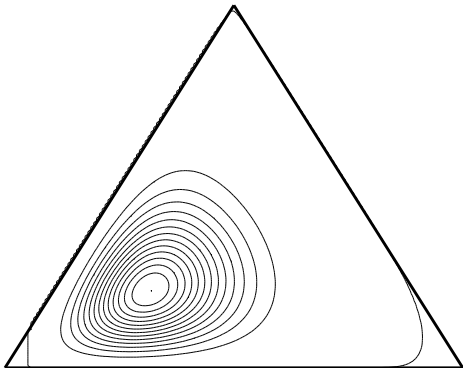
4.3.2.2 Convergence of Bayesian Folding-In

It is not straightforward to see that the alternating application of the formulas (4.47), (4.48) and (4.49) converges towards a local maximum of the posterior (4.40) used for the MAP estimator. The claim is proved in this subsection by casting the maximization procedure as a special case of the EM algorithm. This is not the typical use of the EM-algorithm. EM is often used to compute a maximum-likelihood estimation of parameters of some mixture model. In contrast, we used it here as a tool for hill climbing on a modified kernel density function. This kind of application of the EM technique is interesting in its own right, therefore we report it here in more detail.

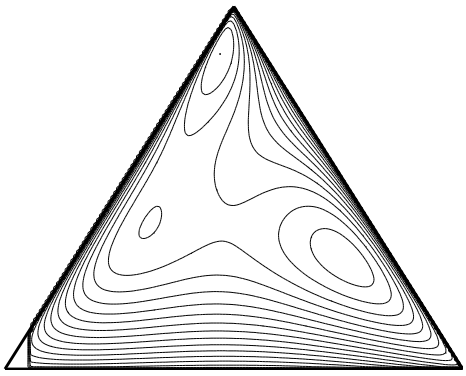
Hidden variables $\vec{y}, \vec{x}$ are introduced for both, the likelihood of Q and the prior distribution to make the maximization of the posterior (4.40) tractable with an EM algorithm. First, the likelihood of a single word $p(w|\vec{\theta}_q, \vec{\omega})$ is seen as a mixture model of K topics. A hidden binary vector $\vec{y}_w \in \{0,1\}^K$ is introduced for each word $w \in W$ that appears in the new document d_q. We do not need to introduce hidden variables for words that do not appear in the new document because those terms gets zeroed by $n(d_q, w)$ in eq.(4.46). The vector $\vec{y}_w$ follows a 1-out-of-K representation that



(a)



(b)



(c)

Figure 4.11: (a) topic mixtures for documents learned by PLSI and for queries of different sizes (short, long) generated by Bayesian folding-in, (b) typical posterior in the topic simplex for a long query, (c) short query.

means all components of $\vec{y}_w$ are zero except the one that indicates the topic z_k that explains word w. Thus, the complete likelihood of word w is

$$p(w, \vec{y}_w|\vec{\theta}_q, \vec{\omega}) = p(w|\vec{y}_w, \vec{\theta}_q, \vec{\omega})p(\vec{y}_w|\vec{\theta}_q, \vec{\omega}) \tag{4.50}$$

and can be decomposed into the factors

$$p(w|\vec{y}_w, \vec{\omega}) = \prod_{k=1}^{K} \omega_{wk}^{y_{wk}} = \prod_{k=1}^{K} p(w|z_k = 1)^{y_{wk}} \text{ and} \tag{4.51}$$

$$p(\vec{y}_w|\vec{\theta}_q) = \prod_{k=1}^{K} \theta_{kq}^{y_{wk}} = \prod_{k=1}^{K} p(z_k = 1|d_q)^{y_{wk}} \tag{4.52}$$

This is still the standard method, where the use of hidden variables lead to equations, which allow tractable inference of unknown parameters, in this case $\vec{\theta}_q$. For convenient syntax, the vectors $\vec{y}_w$ for all words are concatenated to the vector $\vec{y}$.

Second, the prior $p(\vec{\theta}_q|\vec{\theta})$ (eq. 4.45) also can be seen as a mixture model with $|D|$ Dirichlet components and equal component priors. Again, a hidden binary vector $\vec{x} \in \{0, 1\}^{|D|}$ is introduced. The vector follows a 1-out-of-$|D|$ representation, thus all components are zero except the one that indicates the Dirichlet component that explains a specific setting of the topic mixture $\vec{\theta}_q$ of the query document. Thus, the prior of the queries topic mixture is expanded to

$$p(\vec{\theta}_q, \vec{x}|\vec{\theta}) = p(\vec{\theta}_q|\vec{x}, \vec{\theta})p(\vec{x}|\vec{\theta}) \tag{4.53}$$

with the factors

$$p(\vec{\theta}_q|\vec{x}, \vec{\theta}) = \prod_{d \in D} \mathrm{Dir}\big(\vec{\theta}_q|\alpha(\vec{\theta}_d)\big)^{x_d} \text{ and} \tag{4.54}$$

$$p(\vec{x}) = \prod_{d \in D} (1/|D|)^{x_d} \tag{4.55}$$

Note that in contrast to the likelihood of Q, the quantity in question $\vec{\theta}_q$ does not appear in the list of the conditions of the prior distribution (4.45). Now, the EM can be interpreted as hill climbing on the posterior (4.40).

Putting it all together, the logarithm of the complete posterior of the whole model (eq. 4.40) is

$$\begin{aligned} \ln(p(\vec{\theta}_q|Q, \vec{y}, \vec{x}, \vec{\omega})) &\propto \ln(p(Q, \vec{y}|\vec{\theta}_q, \vec{\omega})p(\vec{\theta}_q, \vec{x}|\vec{\theta}, \vec{\omega})) \\ &= \Big[\sum_{w \in W} n(d_q, w) \sum_{k=1}^{K} y_{wk} \big[\ln \omega_{wk} + \ln \theta_{kq}\big]\Big] + \\ &\quad \Big[\sum_{d \in D} x_d \big[\ln \frac{1}{|D|} + \ln \mathrm{Dir}(\vec{\theta}_q|\vec{\alpha}(\vec{\theta}_d))\big]\Big] + c \end{aligned} \tag{4.56}$$

Following the EM framework, the function to be optimized is the expectation of (4.56) as shown in eq. (4.46). Thus, in the E-step we get the posteriors for both, the indicator of the PLSA mixture model

$$\begin{aligned}\mathbb{E}[y_{wk}|w,\vec{\theta}_q^{(s)},\vec{\omega}] &= p(y_{wk}=1|w,\vec{\theta}_q^{(s)},\vec{\omega}) \\ &= \frac{p(w|y_{wk}=1,\vec{\theta}_q^{(s)},\vec{\omega})p(y_{wk}=1|\vec{\theta}_q^{(s)})}{\sum_{k'=1}^{K} p(w|y_{wk'}=1,\vec{\theta}_q^{(s)},\vec{\omega})p(y_{wk'}=1|\vec{\theta}_q^{(s)})} = g_{wk}\end{aligned} \tag{4.57}$$

and the indicator of the prior modeled by kernel density estimation

$$\begin{aligned}E[x_d|\vec{\theta}_q^{(s)}] &= p(x_d=1|\vec{\theta}_q^{(s)}) \\ &= \frac{p(\vec{\theta}_q^{(s)}|x_d=1)p(x_d)}{\sum_{d'\in D} p(\vec{\theta}_q^{(s)}|x_{d'}=1)p(x_{d'})} = h_d\end{aligned} \tag{4.58}$$

The final equations (4.47) and (4.48) for the E-step are derived by substitution and simplification using the following identities $p(w|y_{wk}=1,\vec{\theta}_q^{(s)},\vec{\omega}) = \prod_{k'=1}^{K}\omega_{wk'}^{y_{wk'}} = \omega_{wk}$, $p(y_{wk}=1|\vec{\theta}_q^{(s)}) = \prod_{k'=1}^{K}\theta_{k'q}^{y_{wk'}} = \theta_{kq}^{(s)}$, $p(\vec{\theta}_q^{(s)}|x_d=1) = \prod_{d'\in D}\mathrm{Dir}\big(\vec{\theta}_q|\alpha(\vec{\theta}_{d'})\big)^{x_{d'}} = \mathrm{Dir}\big(\vec{\theta}_q^{(s)}|\alpha(\vec{\theta}_d)\big)$, and $p(x_d) = \prod_{d'\in D}(1/|D|)^{x_{d'}} = 1/|D|$.

In the M-step, the expectation of (4.56) is maximized with respect to $\vec{\theta}_q$ under the condition $\sum_{k=1}^{K}\theta_{kq}=1$. Equation (4.46) is extended by a Lagrange multiplier and the Dirchlet is expanded.

$$\begin{aligned}\mathcal{L}(\vec{\theta}_q) =& \Big[\sum_{w\in W} n(d_q,w)\sum_{k=1}^{K} g_{wk}\big[\ln\omega_{wk}+\ln\theta_{kq}\big]\Big] \\ &+ \sum_{d\in D} h_d\Big[\ln\frac{1}{|D|}+\ln\frac{1}{B(\vec{\alpha}(\vec{\theta}_d)}+\sum_{k=1}^{K}\big(\alpha(\theta_{kd})-1\big)\ln\theta_{kq}\Big] \\ &+ \lambda\Big[\sum_{k=1}^{K}\theta_{kq}-1\Big]+c\end{aligned} \tag{4.59}$$

The first derivative of eq. 4.59 is set to zero

$$\frac{\partial\mathcal{L}(\vec{\theta}_q)}{\partial\theta_{kq}} = \sum_{w\in W} n(d_q,w)g_{wk}\frac{1}{\theta_{kq}} + \sum_{d\in D} h_d(\alpha(\theta_{kd})-1)\frac{1}{\theta_{kq}} + \lambda = 0 \tag{4.60}$$

Solving for θ_{kq} and substituting into the side condition yields the update equation (4.49) for the kth component of the new topic mixture $\vec{\theta}_q^{(s+1)}$ of the query document.

4.3.3 Applications of Bayesian Folding-In

Two applications of Bayesian folding-in are studied in the sequel. First, in order to determine the latent topic mixtures for the documents of a large collection, the original PLSA is applied to a subset of documents of the collection, while the rest of the documents are folded in using Bayesian folding-in. The rationale behind the idea is to avoid overfitting of PLSA by keeping the number of parameters small. As the number of PLSA's parameters grows linearly with the training set of documents, that set is kept small as possible. Bayesian folding-in for the rest of the documents makes use of both the learned word topic associations as well as the learned latent topic mixtures of the training documents.

The second application of Bayesian folding-in is information retrieval. A document collection is processed by PLSA and then queries are folded into the latent document space by Bayesian folding-in. Different strategies of folding in are discussed.

4.3.3.1 PLSA with Bayesian Folding-In

The representation of documents as vectors of mixture proportions of latent topics is useful in several situations. The straightforward way to get those vectors for the documents of a collection is to apply PLSA to the whole collection. However, this way has several drawbacks, namely (i) overfitting due to the large number of parameters of PLSA which is linear in the size of the document collection, (ii) skewed topic occurrences present in the whole collection may distort the latent documents representations to be learned, and (iii) large computational costs.

Therefore, we propose to apply PLSA to a representative subset of documents only, while the rest of the documents is folded into the latent subspace by Bayesian folding-in. This strategy reduces the number of parameters for PLSA and therefore the load of the tempered EM algorithm to get stuck in a suboptimal local maximum of the likelihood. The output of PLSA consists of the word topic association matrix and the latent topic representations of the document in the subset. In order to get a model which is equivalent to the case when PLSA is applied to the whole collection, the representative subset of document has to cover the vocabulary of the whole collection. Furthermore, since PLSA cannot benefit from words which appear in only a single document each word has to occur in at least two documents of the representative subset. So, the first problem to solve is how to select a representative subset of documents.

The second problem is concerned with Bayesian folding-in. As shown in the examples in the previous section, Bayesian folding-in can be seen as hill climbing on a multimodal function. The goal is to find the global maximum among the local maxima. This depends on the start point of Bayesian folding-in. Thus, the second problem consists in selecting a set of suitable start points, so that Bayesian folding-in converges to the global maximum in at least one case. Both problems are discussed in the next two paragraphs.

Algorithm 1 Greedy Algorithm

1: $D = \{d_1, \ldots, d_N\}$
2: $V =$vocabulary(D)
3: $V' = V$
4: $D' = \emptyset$
5: **while** exists a $w \in V$ which occurs in less than two documents in D' **do**
6: $\quad d =$ pick the document in D which has the largest intersection with V'
7: $\quad D = D \setminus \{d\}$
8: $\quad D' = D' \cup \{d\}$
9: $\quad V' =$ words in V which occur in less than two documents in D'
10: **end while**
11: **return** D'

4.3.3.1.1 Representative Subset of Documents Given a collection of documents D, the selection of a representative subset of documents $D' \subset D$ is crucial for a good overall model of the collection. One important criterium for the subset D' is to cover the whole vocabulary of D. The problem is an instance of the minimal set cover problem, where given a universe U and a collection C of subsets of U, the minimal subset of the collection C is wanted, whose union equals the universe U. The greedy algorithm is the algorithm with polynomial runtime which gives the best approximation [48].

In order to run PLSA effectively, the additional requirement is made, that each word of the vocabulary appears in at least two documents of the subset. The greedy algorithm for that problem is given by algorithm 1. The greedy algorithm which gives the best approximation of the set cover problem, is compared to the algorithm which selects documents randomly in line 6.

Note, that for a good model of the document collection an optimal solution of the set cover problem is not necessarily needed. The representative subset D' just has to be small enough to avoid overfitting during PLSA learning of the word topic associations and initial topic mixtures for the kernel density based prior.

4.3.3.1.2 Starting Points for Bayesian Folding-In Bayesian folding-in of a new document is a deterministic process, which starts with an initial topic mixture for that document and iteratively performs hill climbing on the posterior density (4.40) until it converges towards a local maximum. However, as the posterior may have multiple local maxima it depends on the starting point of the hill climbing to which of the local maxima Bayesian folding-in converges in the end. From a data modeling perspective, the local maximum with the largest posterior, i.e. the global maximum, is preferred since we are doing maximum a posteriori estimation. The posterior consists of two factors, one which depends on the given new document or query at hand, i.e. called likelihood, while the other factor, called prior, is independent of the documents to be folded in. From practical experience the prior itself has typically multiple modes and is mainly responsible for the multimodal structure of

the posterior. The likelihood has usually a single mode only. In order to find a small and general set of starting points, which is independent from the document to be folded in, the prior is analyzed only. Note that the prior is independent of the document to be folded in, therefore, the following analysis has to be done only once as a preprocessing step.

The idea to get the set of starting points is to determine the local maxima of the prior. This can be done by hill climbing as well. Note that this maximization is a special case of the maximization of the posterior (4.40), just that the likelihood becomes a constant. Thus, the hill climbing on the prior can be formulated as EM algorithm as well with the following modifications. The posterior for the hidden variable $\vec{y}$ eq. (4.47) is not necessary anymore and can be neglected. The E-step consists only of eq. (4.48). The update formulae (4.49) of the M-step reduces to

$$\theta_{kq}^{(s+1)} = \sum_{d \in D} h_d \theta_{kl} \tag{4.61}$$

A hill climbing on the prior distribution is started at the topic mixture vector of each document in D' and iterated until convergence. As a result the end points of the hill climbing are determined for all documents in D'. Documents which belong to the same local maximum of the prior, have very close end points. Those end points are reduced to a set of representative end points by k-means. The number of clusters k is heuristically chosen between 4 and 10.

For a new document, those representative end points are used as starting points for Bayesian folding-in, i.e. for each representative end point a Bayesian folding-in hill climbing on the posterior is started. This gives k possible topic mixtures for the new document. Among those the topic mixture is chosen with the highest posterior value.

4.3.3.2 Bayesian Folding-In and Information Retrieval

Bayesian folding-in determines for a query or a new document mixture proportions of latent topics, which can be seen as a K-dimensional vector. The found vector can be used to determine similarities to the documents of a given collection, for which such latent representations have been determined before. Usually the similarities are calculated by cosine similarity. Hofmann proposed model averaging [75], which linearly combines for a particular query document pair the similarities determined on different latent representations (for which usually the number of aspect varies) as well as the similarity determined on the original term representations.

Bayesian folding opens two new degrees of freedom to tune a ranking, namely (i) the choice of the starting point for folding-in and (ii) varying the smoothing parameter h. The answer to the choice of the starting point is a different one from the information retrieval perspective as before from the data modeling perspective. For data modeling it is fine to select the start point, for which Bayesian folding-in converges to the global maximum of the posterior. However, information retrieval

has the more informal goal to find documents relevant to the query. As especially short queries may have ambiguous meanings, the most appropriate representation of the query in the latent space may not necessarily correspond to the global maximum of the posterior. Alternative meanings correspond to the set of local maxima of the posterior. Using relevance feedback, an information retrieval system may learn, which local maximum is most appropriate for the user and the retrieval task at hand. If no additional information (e.g. from relevance feedback) is available, the global maximum of the posterior gives the most plausible query representation.

The second new degree of freedom is the smoothing parameter h. The larger h the more the prior changes towards a flat distribution. The kernel density-based prior helps to focus onto the relevant part of the latent space during folding-in. A very large value for h effectively removes this focus and allows all possible mixture proportions of the latent topics for the query representation. In terms of information retrieval, the parameter h can be seen as a quantity, which specifies how general the query can be interpreted to match documents in the returned ranking. Large h means more general, since the latent query representation is allowed to take mixture proportions that are quite distant from the rest of the documents in the collection. The discussion shows the potential of Bayesian folding-in for improvements in information retrieval.

4.3.4 Experiments

Two experiments are performed to study the performance of Bayesian folding-in regarding the following questions: (i) does Bayesian folding-in improve the model of a document collection based on the output of PLSA trained on a subset of the documents, and (ii) does Bayesian folding-in improve the retrieval of relevant documents for a given query.

The lemur package[1] is used for preprocessing the text data as well as training PLSA on a collection of documents. Additionally the *trec_eval*[2] tool (version 8.1) is used to assist the evaluation of the information retrieval experiments.

4.3.4.1 Data and Preprocessing

The proposed applications of Bayesian folding-in are evaluated on public benchmark text corpora. The evaluation of Bayesian folding-in in combination with PLSA can be done on text documents only. For the application to information retrieval, queries with known relevant documents are additionally needed as ground truth to estimate precision and recall. Five document collections are used in this study, namely CISI (1460 document, 112 queries), CRAN (1400 documents, 225 queries) and MED (1030 documents, 30 queries) [3] as well as CARS (1000 documents) and BASEball (1000 documents), which are part of the *20newsgroups* collection [4]. Queries only are avail-

[1] lemurproject.org
[2] trec.nist.gov/trec_eval
[3] ir.dcs.gla.ac.uk/resources/test_collections/
[4] people.csail.mit.edu/jrennie/20Newsgroups/

able for CISI, CRAN and MED.

Preprocessing of documents and queries includes elimination of stop words using the list from the SMART project[5] as well as the elimination of infrequent words. Infrequent words are those that occur in less than δ documents and hence are assumed to convey only minor information. In the first experiment, δ is chosen to be 10, and in the second experiment δ is chosen to be 5, which guarantees that no query becomes empty. Documents containing less than 5 words are neglected. All terms are reduced to word stems using Porters stemmer. Additionally, the headers of the documents of CARS and BASEBALL are removed. Those headers include email addresses and subject lines, which include often very noisy textual information, e.g. the subject lines include all type of prefixes from email clients in random order.

4.3.4.2 Likelihood Experiment

The aim of this experiment is to determine the capability of the approaches to model a document collection. In detail, Bayesian folding-in (BFI) is compared to two PLSAs folding-in (PFI), and (iii) PLSA performed on the entire collection (PLSA). Additionally, the two approaches to select a representative subset of documents are compared, namely random selection and the greedy algorithm.

The overall experimental set-up used to assess the modeling capability of the three approaches is shown in figure 4.12.

Given a collection of documents, the set of document-word pairs does not include stopwords as well as infrequent words. The hold out data, on which the predictive likelihood is determined, is randomly sampled and consists of 10% of those document word pairs. The other 90% of the document word pairs are used as training data. Although the document word pairs are randomly sampled it is guaranteed, that (i) each document occurs in the remaining training data and consists of at least five words, and (ii) each word occurs at least in two different documents in the training data.

Step two consists of selecting a representative subset of documents from the training data. PLSA determines on the representative subset an incomplete model, which consists of the word topic associations of the whole vocabulary and the topic mixtures for the representative subset of documents. The information of the incomplete model is used to fold-in the rest of the documents in training data to determine complete models, which now include the topic mixtures for all documents. The folding-in is done using Bayesian folding-in as well as using PLSAs folding-in. A third complete model is determined by training PLSA on the whole training data. The number of latent topics is always set to $K = 32$.

The complete models are used to determine the predictive likelihood on the hold out data:

$$\mathcal{L} = \sum_{(d,w)\in\text{hold out data}} n(d,w)p(d,w) \tag{4.62}$$

[5] http://ir.dcs.gla.ac.uk/resources/ir_sys

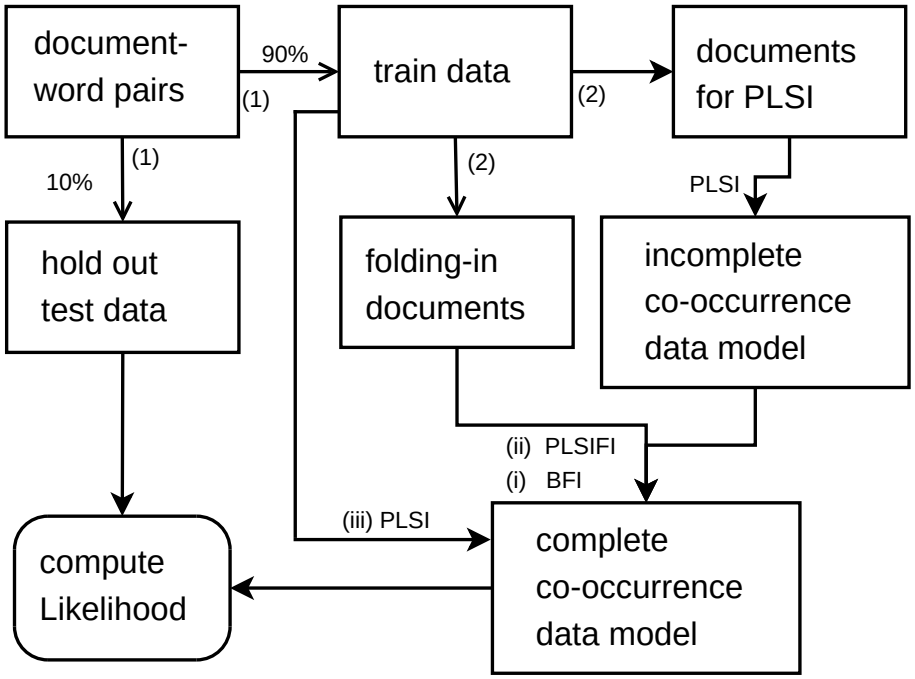


Figure 4.12: Experimental set-up to compare predictive likelihoods of the three studied model variants.

	PLSA	PLSA-FI	BFI
BASE	−107868	−100776	−99741
CARS	−83939	−83975	−83340
CISI	−104319	−97584	−97306
CRAN	−155344	−144479	−143336
MED	−80868	−82903	−82116

	PLSA	PLSA-FI	BFI
BASE	−108316	−102543	−101333
CARS	−84010	−85474	−85124
CISI	−104121	−98634	−98460
CRAN	−155230	−146866	−146088
MED	−81569	−84343	−83488

Table 4.2: Results of the first experiment. Shown are the predictive log-likelihoods of the test data given the three variants of the compared trained co-occurrence data models. The upper table shows results for random selection, the lower table for greedy selection.

	CISI	CRAN	MED	CARS	BASE
rand. sampl.	0.611	0.597	0.577	0.635	0.63
set cov. alg.	0.113	0.105	0.147	0.131	0.133

Table 4.3: Shown are the percentages of all training documents that are selected by the two algorithms used to select representative subsets.

This procedure is repeated 20 times and the averaged likelihoods are reported.

4.3.4.2.1 Results The results of the experiments on the benchmark collections are reported in table 4.2.

The higher the predictive log-likelihoods the better the model has learned the data. Using random selection for the representative subset of documents, Bayesian folding-in gives for all collections but MED the model with the highest predictive likelihood. Additionally, the PLSA-FI model has a better or comparable predictive likelihood then the standard PLSA model.

A second run is performed using the greedy algorithm to select a representative subset of the training documents. Again the results show that for BASEball, CISI, and CRAN Bayesian folding-in is advantageous in terms of predictive likelihood. Furthermore, PLSA folding-in performs better than the standard PLSA model trained on the entire training data. Again, in case of the MED collection but also in case of the CARS collection the standard PLSA model performs best.

One reason explaining this observation may be the fact that the standard PLSA model has to learn parameters proportional to the number of different words and

documents. After preprocessing, CARS and MED consist of the smallest set of different words and documents. MED consists of 1033 documents and 1489 words, CARS consists of 1000 documents and 1236 but e.g. CISI consists of 1460 documents and 1188 words. Additionally, the differences of the number of words and documents is multiplied by the number of chosen topics. Hence, the PLSA models for MED and CARS are the most simple ones and may be not prone to overfitting as strong as this leads to outperforming by PLSA-FI or BFI models. As before one observes in all cases, that Bayesian folding-in always gives better predictive likelihoods than PLSA folding-in does.

The approaches for selection of representative subsets are compared by the mean percentage of the size of the selected subset relative to the set of all training documents. Results are reported in table 4.3. In all cases the representative subset is significant smaller, when using the greedy algorithm. A smaller representative subset makes it harder to learn the topic mixtures of the data collection and thus one may expect the predictive likelihoods to decrease. Indeed the results show, that the PLSA-FI and BFI models using random sampling reach better predictive likelihoods as those using the greedy algorithm.

These results indicate: (i) in agreement with [113] that the PLSA model is prone to overfitting, (ii) that PLSA-FI and BFI may help to reduce the tendency of overfitting, and (iii) that Bayesian folding-in is more robust than PLSA folding-in.

4.3.4.3 Recall Precision Experiment

The second experiment aims at assessing the capability of Bayesian folding-in in comparison to PLSA folding-in to be of use for information retrieval. In detail, the task is to retrieve a set of documents as the answers to a query with high precision and recall. For a given document collection and a query, recall is defined as:$|a \cap b|/|a|$ with the defined sets of relevant documents a and the retrieved documents b respectively. Whereas precision is defined as: $|a \cap b|/|b|$. Due to the need of a list of relevant documents for the queries, this experiment is only done for the data sets CISI, CRAN, and MED.

First, PLSA is performed on the entire document collections for 32 topics resulting in a co-occurrence data model for each of them. As discussed in section 4.3.1 this model estimates for each document a vector in the latent topic space consisting of the learned document-topic mixtures. Afterwards, the queries are folded into the previously obtained co-occurrence data model using (i) PLSA folding-in, and (ii) Bayesian folding-in. The similarity between each query and all documents of the collection is computed and postprocessed to give interpolated recall-precision graphs. Similarities are defined as a weighted sum of similarities in the latent semantic space which are influenced by either PLSA-FI or B-FI, and in the original vector space spanned by the words. This strategy was proposed by Hofmann [75].

4.3.4.3.1 Results The results of the second experiment are shown in figure 4.13. In general, the more the graph approaches the upper right corner the better the retrieval

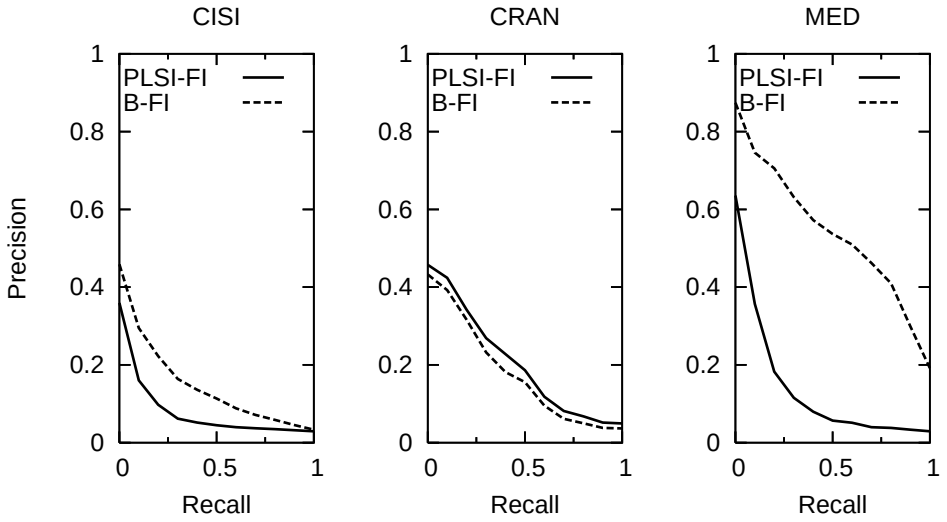


Figure 4.13: Interpolated recall-precision graphs of (i) PLSA folding-in, and (ii) Bayesian folding-in.

performance. In case of CISI and MED, one observes that for all recall values, the obtained precisions using Bayesian folding-in are above those using PLSA folding-in. In case of CRAN, the performances of both methods are comparable. These results indicate that (i) the B-FI benefit using a prior distribution over document-topic mixtures, and (ii) the B-FI is advantageous in information retrieval tasks. The results are in agreement with the results of the predictive likelihood experiment showing that B-FI models always reach higher predictive likelihoods then PLSA-FI models.

The smoothing parameter $h = 0.02$ has been kept constant during all experiments. This value gives a reasonable small set of different starting points for Bayesian folding-in. In general, that parameter should be estimated using cross-validation, which is a general approach to find setting for hyper-parameters of Bayesian methods.

4.4 Summary

We propose a new method for hill climbing a kernel density estimate. The hill climbing procedure is devised as a special case of the expectation maximization algorithm. This shows a new connection between the theories of kernel density estimation and expectation maximization. We demonstrated the usefulness of the new hill climbing for density based clustering. The new connection allows the use of several approximation methods used for the EM algorithm in general in the context of density based clustering.

Furthermore, we demonstrate the application of the new method for combining kernel density priors with standard probabilistic models. In our case, a new Bayesian

method for folding new documents into the latent space determined by PLSA is proposed. Additionally to PLSAs folding-in, a prior based on kernel density estimation with Dirichlet kernels is used. Two application scenarios for Bayesian folding-in are proposed and its superior performance in both scenarios has been demonstrated on real document collections.

5 Data Analysis by Fractal Dimension

Real datasets exhibit patterns and regularities. A main consequence is that in case of high-dimensional vector data the points typically lie on low-dimensional manifolds, rather than being evenly spread out. A similar fact is observed for non-vector data, e.g. text strings, for which only the pairwise distances or similarities between the data objects are known. Real data of that kind never use all degrees of freedom, that are permitted by the formal data representation, e.g. in English words letter 'q' is rarely followed, if at all, by letter 'w'. Both phenomena, low-intrinsic dimensionality and few degrees of freedom, are rooted in hidden regularities in data. Detecting that known data subsets significantly differ in the used degrees of freedom or even find the subsets themselves that have low intrinsic dimensionality is useful in tasks such as indexing and classification. The material presented in this chapter is a revised version of our publications on dimension induced clustering [56] and the application of fractal data analysis to study the degrees of freedom of mitochondrial transit peptides [128].

It has recently been proved [84] that the well-known "curse of dimensionality" translates essentially to a "curse of intrinsic dimensionality," in terms of finding efficient approximations to nearest-neighbor queries. Thus, knowing that a certain subset of data has small intrinsic dimensionality helps to build a more efficient index for this part of the data. Furthermore, separating points based on some notion of "local dimensionality" is helpful in identifying subsets of points that are qualitatively different. For example assuming a geographical setting, locations along a river belong to a 1-D manifold, whereas locations on a lake would belong to a 2-D manifold (for example Figure 5.1). Similarly, road intersections along a highway are on a 1-D manifold, while intersections within a city belong to a 2-D manifold. Thus, discovering low-dimensional manifolds is also useful in its own right.

However, we are faced with three main challenges. The first question that naturally arises is what "dimension" exactly means. Data patterns may be fairly complicated. Assuming that text data follow rules based on regular expressions or that high-dimensional points follow linear trends and always lie on hyper-planes is fairly restrictive. For example, in a more abstract setting the lake and river may lie on the same 2-D plane, but they still differ in dimension. Second, to complicate matters even further, the observations may not even belong to a vector space. Yet, we should be still able to define the dimension of the data. Finally, in practical applications, the dimension of the embedding space is large, in the order of thousands. Any method of practical interest should be able to deal with spaces of arbitrarily high dimension and still successfully estimate the intrinsic dimensionality and if necessary find low-dimensional subsets embedded in the original space. It is thus desirable to

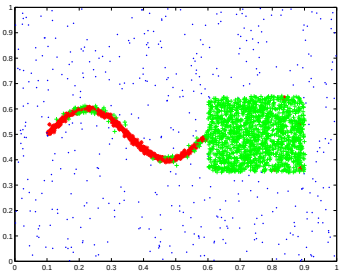


Figure 5.1: A dataset that contains two subsets of different intrinsic dimensionality

characterize data following complex patterns embedded in spaces of any type, vector and non-vector spaces, and devise algorithms for estimating intrinsic dimensionality as well as identifying subsets of data with low dimensionality. As we shall see, it is possible to intuitively define a notion of dimension, which does not depend on the notion of a linear subspace. Furthermore, the algorithms for estimating intrinsic dimension and identifying low-dimensional manifolds are not sensitive to the dimension of the original space and, thus, do not suffer from the "curse of dimensionality".

In order to argue about and detect the existence of a low-dimensional manifold in our data, there must exist a sufficiently large number of points that are densely packed on this manifold. Therefore, it seems reasonable, that the concept of point density captures the low intrinsic dimensionality and using density based methods would be able to detect such subspaces. However, we argue that density alone is not sufficient. For the sake of example consider city locations interspersed among highway intersections. The cities, lying on a 2-D surface, form a distinct data subset. The road intersections form another subset, as a complex network of denser 1-D lines which occupies the *same* space as the cities. Density-based clustering approaches have a limited ability to detect clusters-within-clusters. In this simple example, they would typically produce either a large number of separate city clusters (one for each group of cities enclosed by roads), or one single cluster containing both intersections and cities, depending on the density thresholds.

Consider also the example in Figure 5.1. In this case we have three qualitatively different types of points. The set of points that lie on the curved 1-D line, the set of points that lie on the 2-D cloud, and the noise points that are scattered in the 2-D plane. If the line and the square have the same density, it is impossible to detect that they differ in dimensionality. If the subset are not known in advance (all points have same color), using a density based method is not possible to detect all three of these subsets. Any density threshold will just separate the noise points from the rest. Note also that dimensionality by itself would not be able to separate the square from the noise points, since the noise points are also 2-dimensional. The synergy of density and intrinsic dimensionality gives a clear separation of the three distinct datasets, as it is shown in the figure.

In the reminder of the chapter we first review a suitable definition of intrinsic

dimensionality, which can be applied to vector and non-vector data (Section 5.1). Second, we introduce a simple test procedure to verify that known subsets of the data significantly differ in intrinsic dimensionality (Section 5.2). Furthermore, we present an application of this procedure to biological problem, namely to verify the reduced number of degrees of freedom in mitochondrial transit peptides in plants in contrast to those in animals and fungi. The third subsection, Section 5.3, deals with the problem, that the subsets are not known beforehand but shall be detected in an unsupervised manner. Last, Section 5.4 summarizes the approaches presented and discusses extensions of both approaches, which are subject to ongoing research.

5.1 Correlation Dimension

We draw upon the concept of correlation dimension which is one of several definitions of fractal dimension. The concept of fractal dimension [46,93] has different theoretical definitions, among which Hausdorff dimension is a prominent one. The general idea is to cover the data space by a number of non-empty balls. Hausdorff dimension is defined as the ratio between the logarithm of the minimal number of balls of same size needed to cover the whole data in the space and the logarithm of the balls size. The practical estimation of the Hausdorff dimension requires a solution to the difficult problem of covering the data by a minimal number of balls.

For practical estimation tasks on real data, other definitions of fractal dimension, namely correlation dimension and box-counting dimension, have been used. Because the definition of box-counting dimension is applicable only to vector data, we focus in our discussion on correlation dimension.

First, we introduce the original definition of correlation dimension in terms of distances. Second, we review practical ways to estimate the dimensionality and last, we give illustrative examples that demonstrate the estimation procedures.

5.1.1 Definition

Let $X = \{x_1, x_2, \ldots\}$ be a set of objects and δ_{ij} be the distance between x_i and x_j. The correlation integral $C(r)$ for a given radius r is

$$C(r) = \limsup_{N\to\infty} \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \Theta(r - \delta_{ij}) \tag{5.1}$$

The specific use of the Heaviside function

$$\Theta(b) = \begin{cases} 0 & , \text{ if } b \leq 0 \\ 1 & , \text{ else} \end{cases} \tag{5.2}$$

indicates whether two objects have a distance smaller than r. So, the inner sum over j counts how many objects are in a ball of radius r centered at x_i. The term after the limit denotes the fraction of pairs of objects with an index smaller N that

are closer than r. The limit is derived by assuming an infinite set of objects with respective distances. The correlation dimension d is the rate at which the logarithm of the correlation integral decreases, when r is shrinking.

$$d = \lim_{r \to 0} \frac{\log C(r)}{\log r} \tag{5.3}$$

5.1.2 Practical Estimation

In practice, only finite data sets are available. There are several heuristic methods to estimate correlation dimension from finite data, namely Grassberger-Procaccia [61] and sandbox [86]. We review them briefly and discuss them in the context of non-vector spaces that have some additional restrictions than the usual setting of vector spaces. Here, a non-vector space consists of a set of objects and the pairwise distances between them are known. In the vector space setting, the objects are embedded into a multi-dimensional vector space and the distance is some vector distance measure. Note that in a vector space the distances to arbitrary points can be calculated, also to those points that do not belong to the given data. This is not possible in the setting of a non-vector space.

When dealing with finite data, the original definition of the correlation integral eq. (5.1) cannot be used due to the limit $N \to \infty$. Note that the equation can be seen as an average (outer sum with an factor of $1/N$) of N fractions each giving the fraction of objects contained in a ball (inner sum with an factor of $1/N$). A standard heuristic explained in [127], which is used to get more robust estimates of the fraction of data contained in a ball, is to leave out the center point which induces the ball. The removal of the center points by the mentioned standard heuristic for correlation dimension avoids that all balls would be trivially non-empty by definition, because without the heuristic always the center point would be part of the ball. The empty balls must be filtered before the computation of the correlation integral, in order to avoid distortions of the estimate of the correlation dimension.

We denote by $\hat{p}_i$ the estimate of the fraction of data contained in the ith ball of radius r:

$$\hat{p}_i(r) = \frac{1}{N-1} \sum_{j=1, i \neq j}^{N} \Theta(r - \delta_{ij}) \tag{5.4}$$

In order to avoid empty balls in the following formulas, we denote by $\hat{N}(r)$ the number of non-empty balls, which is basically the number of $\hat{p}_i$s that are larger zero. Note that $\hat{N}(r) \leq N$. Additionally, let be $\hat{I}(r) \subseteq \{1, \ldots, N\}$ the index set of those non-zero $\hat{p}_i(r)$s.

The Grassberger-Procaccia algorithm [61] estimates the correlation dimension by computing an estimate of the correlation integral for several representative values $r_1, r_2, \ldots$ for radius r. In order to estimate the correlation integral for a specific r, the algorithm computes the arithmetic mean of the $\hat{N}(r)$ non-zero fraction estimates

$\hat{p}_i(r)$.

$$\hat{C}(r) = \frac{1}{\hat{N}(r)} \sum_{i_j \in \hat{I}(r)}^{\hat{N}(r)} \hat{p}_{i_j}(r) \tag{5.5}$$

Then, $\log \hat{C}(r)$ is plotted versus $\log r$ in the so called log-log plot and a line is fitted to the points of the linear part of that curve. Correlation dimension is estimated as the slope of the line fitted to the linear part of the curve in the log-log plot.

An alternative to the Grassberger-Procaccia algorithm is the sandbox method. The difference is that the arithmetic mean of the $\hat{p}_i$s is replaced by the harmonic mean of those $\hat{N}(r)$ quantities.

$$\tilde{C}(r) = \hat{N}(r) \Big[\sum_{i_j \in \hat{I}(r)}^{\hat{N}(r)} \hat{p}_{i_j}(r)^{-1} \Big]^{-1} \tag{5.6}$$

The rest is the same as for the Grassberger-Procaccia algorithm.

An open question is how to find the linear part of the correlation integral in the log-log plot? The question is difficult to answer automatically, because there are datasets that do not induce a linear part in the log-log plot. Such cases include (a) there is no linear part at all, and (b) there are multiple linear parts of different slopes. Correlation dimension is not defined in these cases. Therefore, all known estimation procedures require some kind to manual tuning first, to decide whether a single linear part of the correlation integral in the log-log plot does exist and second, to determine the particular interval in the domain of the possible radii that is used for fitting the line. In the next sections, we manually inspected a representative number of log-log plots for each type of data set to decide whether there is a single linear part and to determine an interval in the radius domain that is used for line fitting.

5.1.3 Examples

We illustrate the idea behind correlation dimension by two simple examples (see figure 5.2(a) and (b)). Assume the given data set is a finite sample of two-dimensional points, which are uniformly distributed. The number of points in a ball of radius r around a particular point x_i is approximately proportional to the area covered by the ball, which is πr^2. Thus, the correlation integral grows nearly quadratically for medium values of r. Figure 5.2 (c) and (d) show the log-log plot for Grassberger-Procaccia and sandbox method respectively. In the log-log plots, the linear part of the lower curve has a slope close to two. This agrees with the intuition that the dimension of the data should be two. The slope is estimated by fitting a line to the marked points. The tail for small values for r should be ignored as the correlation integral depends here on balls, which include very few points. The part for very large radii is not informative, as here the balls include all points and therefore the correlation integral is not growing anymore.

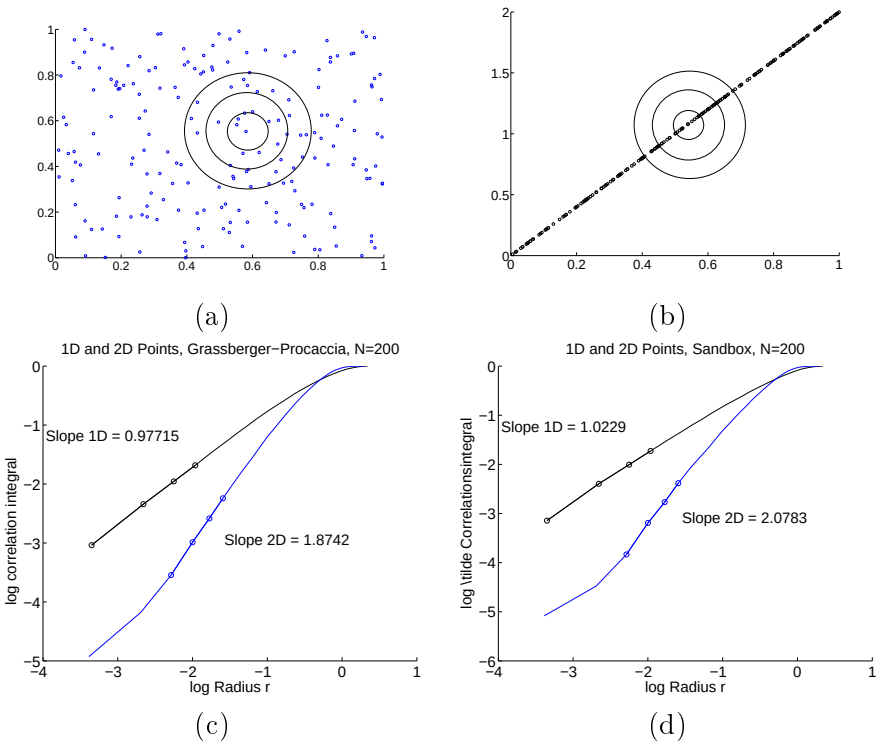


Figure 5.2: Example data with $N = 200$ points uniformly sampled from a plane (a) and from a line (b). Correlation integral versus radius for both data sets and the slopes of the fitted lines, which estimate correlation dimensions using Grassberger-Procaccia (c) and Sandbox (d). The lines are fitted to the marked points.

In the second example, points are sampled from an line, which is arbitrarily embedded in the two-dimensional space. The number of points in a ball of radius r around a particular point x_i is approximately proportional to the length of the part of the line, which is covered by the ball. Thus, the correlation integral grows only linearly. In the log-log plot, the linear part of the upper curve has a slope close to one which also confirms the intuition.

In a further experiment, we demonstrate the dependency of the estimates on the sample size $N \in \{50, 100, 200, 400, 800\}$ of the data. Figure 5.3 shows the results for simulations of uniformly distributed, $d \in \{1, 2, 5, 10\}$ dimensional data. The errorbars show the median with 5% and 95% quantiles of 50 repeated simulations. The parameter N influences both the standard deviation of the estimates as well as the absolute value of the estimates. Small sample size N gives larger standard deviation while large N leads to smaller standard deviation. The figure shows that it is not possible to estimate true dimensionality for high dimensional data, which is mainly due to small sample sizes. Theoretically, the sample size needed grows exponen-

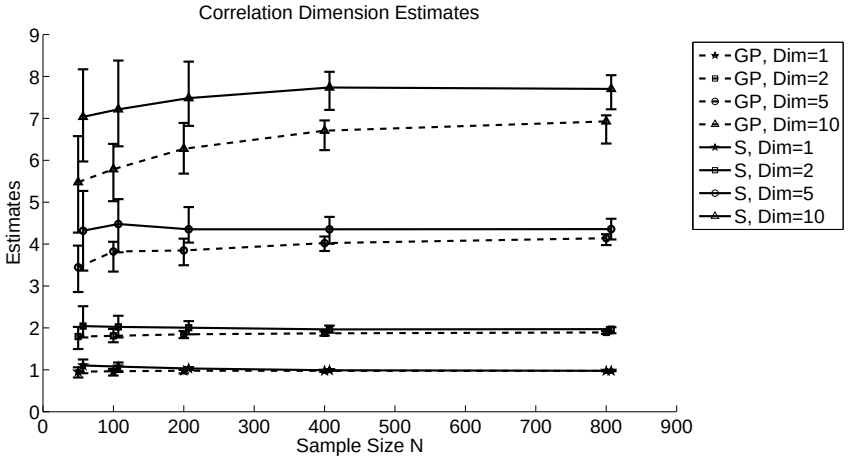


Figure 5.3: Example of estimates of correlation dimension from uniform d-dimensional data using Grassberger-Procaccia (GP) and sandbox (S) depending on sample size. The errorbars show the median with 5% and 95% quantil of 50 repeated simulations.

tially with the number of dimensions. Such very large samples are not available in real settings. However, the estimates of different dimensionalities still become significantly different for sufficiently large samples. The figure also shows that the sandbox method is less sensitive to sample size then the Grassberger-Procaccia algorithm.

5.2 Testing known Classes for different Dimensionality

The typical solution to a classification problem is to find a well working classifier, which can be some model with decision rules. Classifiers are well working, if they can associate new objects with the correct class. A commonly used type of classifiers, generative probabilistic models, are not a single big model, but have a separate model for the data of each class. Models always have parameters that determine the model complexity, e.g. the number of components in a mixture model or the allowed depth in a decision tree. For the task of deciding the model complexities of the class models, it would be beneficial to know that some classes are less complex than others meaning that the data subsets of those classes have smaller intrinsic dimension than the rest of the data.

A classifier is not always build to automate a classification task. It is also used to learn hidden knowledge about the classes from an analysis of the classifier itself, e.g. inspecting a decision tree can reveal interesting properties of the data. Thus, comparing the intrinsic dimensionality of known classes as a hidden property of the data can be useful in its own right.

Comparing the estimates of the intrinsic dimensionalities of different subsets of the data is not just comparing two numbers. The estimates can be biased by the

different sizes of the subsets, the choice of the parameters for line fitting in the log-log plots, and the composition of the particular dataset at hand. Thus, a test procedure is needed to quantify the influences of the randomness inherent in the estimation procedure.

First, we propose a simple test procedure to check, whether the correlation dimensions of two subset are significantly different. Second, we describe an application of the procedure to the analysis of the degrees of freedom of mitochondrial transit peptides in plants, animals and fungi.

5.2.1 Testing by Bootstrap

Let $X = \{x_1, \ldots, x_N\}$ be a dataset of n objects; the first m objects $\{x_1, \ldots, x_m\}$ belong to class A and the remaining $n = N - m$ objects $\{x_{m+1}, \ldots, x_N\}$ belong to class B. Let be $n \leq m$. The question is whether the correlation dimension of the subset belonging to class A is significantly lower than the correlation dimension of the subset belonging to class B.

Why is estimating the correlation dimensions d_A and d_B for the respective classes and comparing the two numbers not satisfactory? The first answer is, we have no idea about the variability of the two estimates, so the result of the comparison may be purely driven by randomness. The second answer points out that each estimate is distorted in a different way, if subset size n is very different from m. This leaves us with two tasks, (i) a reasonable comparison method has to estimate beside the correlation dimensions also the variability of these estimates and (ii) it has to correct the different distortions of the estimates caused by different subset sizes.

We propose to use bootstrap sampling [42] to estimate the variability of the estimate of the correlation dimension. The general idea of bootstrap is to construct a bootstrap sample from a given original dataset of size n by randomly sampling n objects with replacement. Since sampling with replacement may chose some data objects more than once, a bootstrap sample may include duplicates. Assuming the original data does not include any duplicates, choosing the objects uniformly with replacement puts on average about 63% unique objects into a bootstrap sample, while the rest are duplicates. The bootstrap sampling method allows to build random samples as large as the original dataset.

To construct comparable bootstrap samples for each of the two data subsets, the larger one needs to be downsampled to the size of the smaller one. In our setting, we assumed that the subset belonging to class A is smaller than the subset belonging to class B, $n \leq m$. In order to construct a bootstrap sample of size m from the subset with the larger size m, the first step is to draw n objects from the subset belonging to B without replacement. The actual bootstrap sample for the class B is generated in a second step by sampling n objects with replacement from the objects drawn in the first step. Both steps are repeated to generate the next bootstrap sample for class B. Using such a two step procedure instead of directly sampling n objects with replacement from the larger subset keeps the percentage of unique objects in the bootstrap samples at about 63% of the sample size n across both subsets.

A number of L bootstrap samples are computed for both subsets. Correlation dimension is estimated for each of the bootstrap samples. That is the correlation integral is computed for several radii and the logarithm of the radius is plotted versus the logarithm of the correlation integral. A line is fitted to the linear part of that curve and the slope serves as an estimate of correlation dimension. The result consists of two sets of dimensionality estimates, each set having size L.

Last, it is verified by t-tests, whether the correlation dimension of the first subset is smaller than the correlation dimension of the second subset and vice versa. In general, a statistical test like the t-test consists of a null-hypothesis, which states the opposite of the observation. Loosely spoken, it plays the role of the "devil's advocate". In our case, the null-hypothesis is that the mean of the correlation dimension estimated of the bootstrap samples of the subset belonging to class A is equal to the mean of the correlation dimension of the bootstrap samples derived from the subset belonging to class B. The t-test defines a formula to derive an p-values. The null-hypothesis is rejected if the p-value is below a predefined significance level. In case, the null-hypothesis is rejected, we conclude the the correlation dimension of the first subset is significantly smaller than the correlation dimension of the second subset.

5.2.2 Application to mitochondrial Transit Peptides

We applied the task of testing and comparing the correlation dimension of different classes to precursor sequences of mitochondrial proteins [128]. Most mitochondrial proteins are synthesized in the cytosol of eukaryotic cells as precursor proteins carrying N-terminal extensions called transit peptides or presequences which mediate their specific transport into mitochondria. However, plant cells possess a second potential target organelle for such transit peptides, the chloroplast. It can therefore be assumed that mitochondrial transit peptides in plants are exposed to an increased demand of specificity, which in turn leads to reduced degrees of freedom in these transit peptides compared to those of non-plant organisms. We investigates this hypothesis using fractal dimension, namely correlation dimension. Statistical analysis of sequence data shows that the correlation dimension of mitochondrial transit peptides in plants is indeed significantly lower than that from non-plant organisms.

5.2.2.1 Biological Background

Mitochondria are of endosymbiotic origin, i.e. they are derived from the engulfment of a bacterium into a hitherto unknown host cell, an event which finally resulted in the development of eukaryotic cells. In the course of evolution, most of the mitochondrial genes were transferred to the nucleus. As a consequence, most mitochondrial proteins are synthesized in the cytosol of the cell as precursor polypeptides carrying cleavable amino-terminal extensions, named presequences or transit peptides, that mediate transport of the protein 'back' into the organelle.

Comparison of such mitochondrial transit peptides has demonstrated that they

(i) can be quite variable in size, (ii) contain many positively charged, hydrophobic and hydroxylated amino-acid residues, and (iii) have a high tendency to form an amphipathic α-helix [112, 134]. However, since most of these comparisons are based on mitochondrial transit peptides from fungi and mammals (see for example [134]), these conclusions might well be biased and must not necessarily hold true for all species. This is particularly obvious for plants, because plant cells harbor an additional class of organelles of endosymbiotic origin, notably chloroplasts (or generally speaking, plastids). These organelles originate from a second endosymbiotic event in which a cyanobacterium was engulfed by a eukaryotic host cell possessing already mitochondria. It can be assumed that the evolutionary establishment of chloroplasts had an effect also on the selection pressure operating on mitochondrial transit peptides, because these transport signals were suddenly exposed to the situation that a second potential target organelle was present within the same cell. The situation was further complicated by the fact that also most chloroplast genes were phylogenetically transferred to the nucleus. Again, cleavable transit peptides for the transport of the corresponding proteins 'back' into the organelles were developed, which show remarkable similarity to mitochondrial transport signals in terms of N-terminal position and amino acid composition. Considering this scenario, one could predict that mitochondrial transit peptides of plant cells must have adapted to this new situation by developing a higher degree of specialization, in order to prevent permanent transport into the wrong organelle. Supporting evidence for this assumption comes from the observation that several nuclear encoded proteins show dual targeting into both mitochondria and chloroplasts, because they carry transit peptides with ambiguous organelle specificity (for a recent review see [23]). The number of proteins identified with such targeting properties has significantly increased in the past years suggesting that this is a much more common phenomenon than originally anticipated. Still, it can be assumed that mistargeting is only to some degree tolerable for the cell and will probably disturb its integrity and the division of labor between the organelles if it exceeds a certain level.

These considerations led us to the following working hypothesis: while mitochondrial transit peptides in general are characterized by high degrees of freedom in terms of amino acid sequence and composition, plant mitochondrial transit peptides should be significantly less variable in this respect. In order to examine this hypothesis experimentally, mitochondrial transit peptides from one model species each of plants (*Arabidopsis thaliana*), mammalia (*Mus musculus*) and fungi (*Saccharomyces cerevisiae*) were compared by a bioinformatic approach. We used for this purpose correlation dimension, which is a measure of complexity of sequence information within a group of sequences.

5.2.2.2 Methods and Materials

The estimation of the degrees of freedom of a set of protein sequences is a non-trivial task. We interpret the degrees of freedom as the number of independent dimensions, which are necessary to span the data space. In our case, the data space is embedded

into the space of all sequences of length m over the alphabet of the 20 amino acids $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$. The m dimensions correspond to the sequence positions $1, \ldots, m$, which can take either of the different amino acids as values.

In our case, the sequences to be analyzed are transit peptides, which are represented by the 50 N-terminal amino acids of the mitochondrial precursor proteins. In order to constitute a functional transit peptide, not all 50 positions can be filled arbitrarily with amino acids. Instead, some restrictions do apply which are, however, not yet explicitly known. Thus, not each of the positions is counting as an independent dimension and the true dimensionality, which also accounts for the unknown restrictions, is probably less than 50. We avoid the explicit estimation of those restrictions from sequence data, which demands large datasets. Instead, the true dimensionality is directly estimated from sequence similarities.

A common method to compare biological sequences is to align two sequences and compute a similarity score from such alignments. Therefore, we have first computed a similarity score based on alignments for each pair of transit peptides sequences. Then, the correlation dimension was calculated from these similarity scores.

Alignments help to detect biologically relevant similarities between protein sequences. The basic idea behind an alignment between two sequences A and B is to find a transformation from A to B with maximal similarity score in terms of simple edit operations like insert, delete, match, and mismatch. In order to compute an alignment with maximal similarity score, parameters are needed to specify the scores of basic transformation operations. As we are comparing whole transit peptide sequences, we perform global alignments instead of local ones. For the computation of pairwise alignments, we have applied the standard algorithm ClustalW [132].

In our work, we chose the $Blosum_{62}$ matrix as score matrix for matches and mismatches. Furthermore, gaps are penalized with negative scores for gap opening and gap extension. We used a standard parameter combination for gap opening and gap extension, namely -10 and -0.1 respectively, which represents the default setting in ClustalW.

The direct output of such an alignment is the maximal similarity score of the whole transformation. However, those direct outputs are not comparable for different pairs of sequences. As the computation of the correlation integral averages over the number of objects within balls of the same radii but different centers, such a comparison of similarity scores is implicitly assumed. Pairwise alignment in ClustalW already does such a normalization [132], namely by dividing the number of identical sequence positions in the alignment by the number of matched residues. This defines a similarity functions which ranges from 0 to 1.

Correlation dimension is defined in terms of distances instead of similarities. When distances are small, the corresponding similarities are large. Therefore, the definition of correlation integral needs to be adapted to handle similarities. We adapt $\hat{p}_i(r)$ that was previously defined in equation (5.4) by flipping the difference in the argument

of the Heaviside function:

$$\hat{p}_i(r) = \frac{1}{N-1} \sum_{j=1, i \neq j}^{N} \Theta(s_{ij} - r) \tag{5.7}$$

All other equations remain unchanged.

The datasets used were retrieved from the SwissProt/UNIProt database, release 14.1[1]. All entries from mouse (*Mus musculus*), yeast (*Saccharomyces cerevisiae*), and Arabidopsis (*Arabidopsis thaliana*) which have a location attribute containing "mitochondrion" and a topic attribute containing "transit peptide" were collected. Note that the data also include proteins that are only predicted to be mitochondrial proteins. However, SwissProt/UNIProt is quite conservative with those annotations. We obtained 319 entries for yeast, 427 entries for mouse and 224 for Arabidopsis. Since the lengths of the transit peptides were often not known, the 50 N-terminal amino acids of each protein sequence were taken as putative transit peptides.

For each dataset of transit peptides we computed all normalised pairwise alignment scores by ClustalW (version 1.7) using the slow and more accurate alignment method. Note, that the slow alignment methods implemented in ClustalW are essentially the basic methods known as Needleman-Wunsch alignments. As we did not use multiple alignments but pair-wise alignments only, ClustalW is sufficient. New alignment tools like T-COFFEE [108] or MUSCLE [41] offer faster approximations of pairwise alignments or more accurate multiple alignments. Both features are not needed in this project.

Thus, in total a quadratic matrix with normalized pairwise similarities was derived for each set of transit peptides.

5.2.2.3 Results

In the first experiment, the correlation dimension of each of the three datasets of transit peptides is calculated by the Grassberger-Procaccia algorithm. The sandbox method gave comparable results and is therefore neglected. In order to avoid any bias from the different sizes of the datasets, correlation dimension is not calculated on the full datasets but on bootstrap samples of identical size.

For each estimation of the correlation dimension of a dataset, a random bootstrap sample is computed from the original data as described above. The correlation integral is computed for several radii and the logarithm of the similarity radius is plotted versus the logarithm of the correlation integral. A line is fitted to the linear part of that curve and the slope serves as an estimate of correlation dimension.

Figure 5.4 shows for each of the three datasets the results derived from only five random bootstrap samples. The absolute values of the slopes are shown in the insets of the figures. Since visual comparison across the figures is difficult, representative log-log plots of each dataset are combined in figure 5.5. The values of the slopes

[1] `http://www.uniprot.org`, 9/12/2008

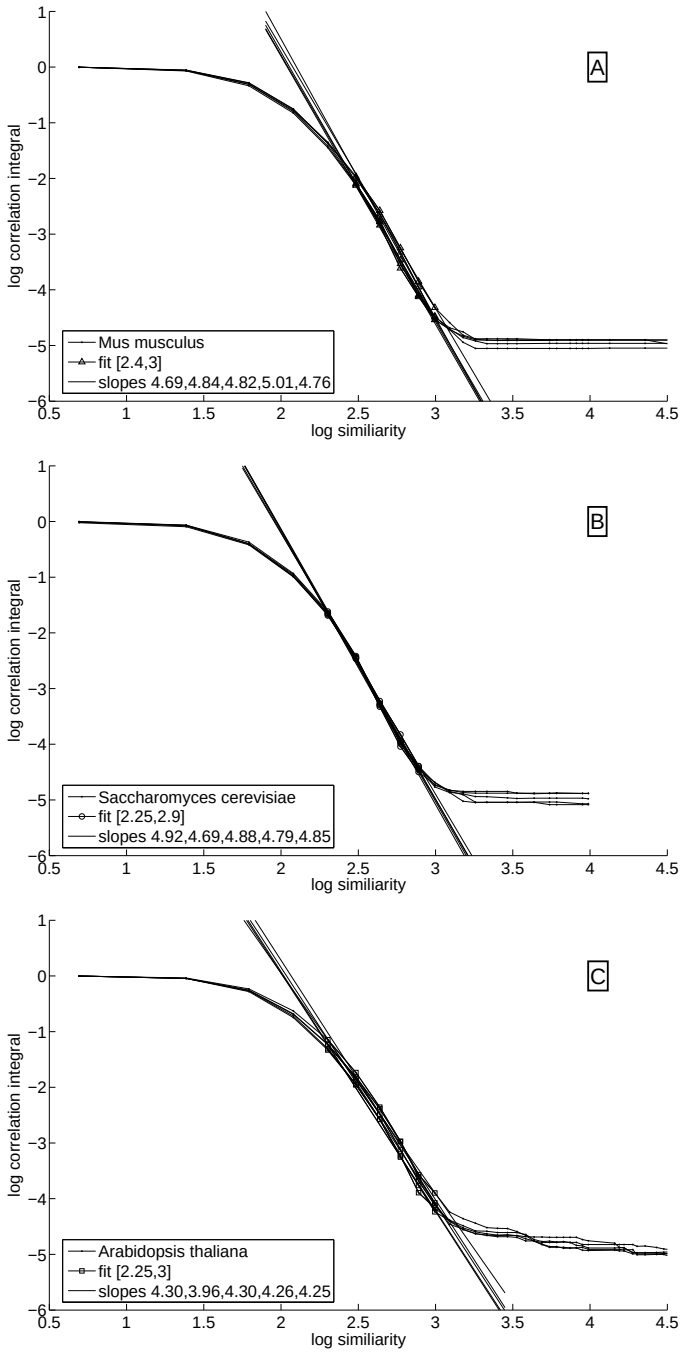


Figure 5.4: Log-log plots and calculated values of correlation dimension (slope) for all datasets (A: *Mus musculus*, B: *Saccharomyces cerevisiae*, C: *Arabidopsis thaliana*) using ClustalW with 10, 0.1 as gap opening and gap extension parameters, respectively. In each case, five examples are shown. The calculated slopes of the fitted lines in those examples are shown in the insets.

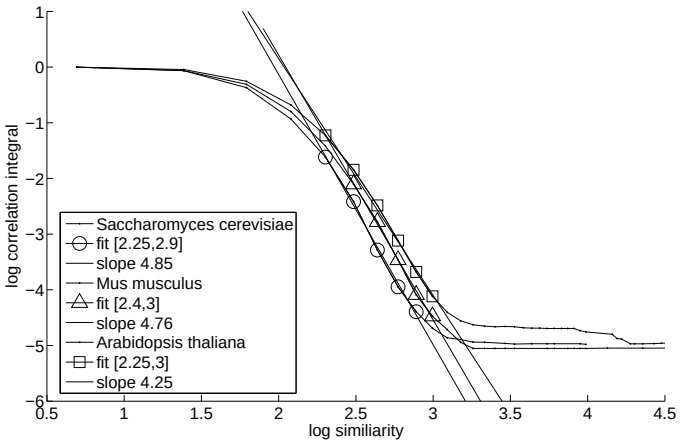


Figure 5.5: Comparison of log-log plots and calculated values of correlation dimension (slope) for all datasets. For further details see the legend to figure 5.4.

calculated for each sample suggest that the transit peptides of *Arabidopsis thaliana* have a lower correlation dimension than those of *Saccharomyces cerevisiae* and *Mus musculus*.

In order to substantiate the results, the calculation was repeated with 1000 random bootstrap samples each of the transit peptides of *Saccharomyces cerevisiae*, *Mus musculus* and *Arabidopsis thaliana*, respectively. The means of the calculated correlation dimensions including error bars showing the standard deviations derived from the results of the 1000 random bootstrap samples is depicted in figure 5.6. Again, it becomes obvious that the mitochondrial transit peptides of *Arabidopsis thaliana* have a lower correlation dimension than those of *Saccharomyces cerevisiae* and *Mus musculus*.

In order to examine to what extent the results of the calculated correlation dimensions are sensitive to the size of the bootstrap samples used, we varied the sample size $N \in \{75, 100, 125, 150, 200, 224\}$. The maximal sample size is determined by the smallest dataset, which in our case is that of *Arabidopsis thaliana*. We generated for each dataset and sample size $L = 1000$ bootstrap samples and estimated the correlation dimension for each bootstrap sample. Subsequently, we derived the mean and standard deviation from the 1000 bootstrap samples generated for each particular sample size. Figure 5.7A shows that with growing sample size the calculated correlation dimension of the transit peptides of *Arabidopsis thaliana* on the one hand and those of *Saccharomyces cerevisiae* and *Mus musculus* on the other hand drift apart.

The visual impression that the transit peptides of *Arabidopsis thaliana* have a lower correlation dimension than those of *Saccharomyces cerevisiae* and *Mus musculus* is verified by t-tests. In our case, the null-hypotheses are that the means of the correlation dimension of the transit peptides of *Saccharomyces cerevisiae* and *Arabidopsis thaliana* as well as those of *Mus musculus* and *Arabidopsis thaliana* are

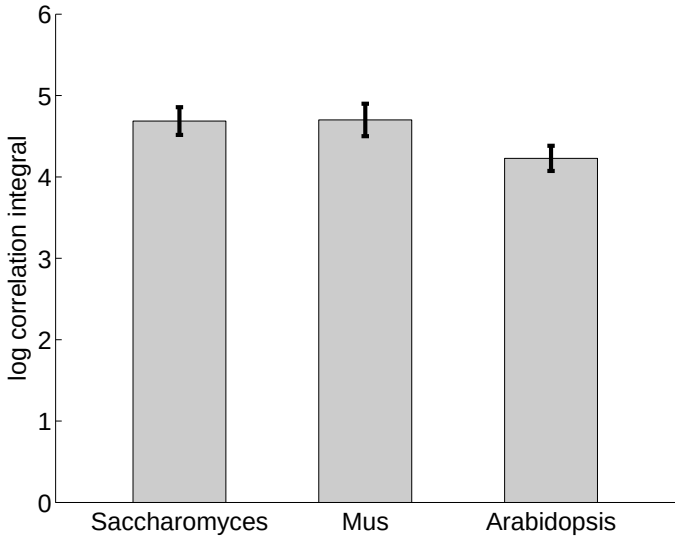


Figure 5.6: Means of the calculated correlation dimensions of *Mus musculus* (Mus), *Saccharomyces cerevisiae* (Saccharomyces) and *Arabidopsis thaliana* (Arabidopsis) taken from 1000 random bootstrap samples each ($N = 224$). The error bars show the respective standard deviation.

equal. Both null-hypotheses can safely be rejected considering that the respective p-values become numerically zero, which is obviously lower than any reasonable standard significance level. This demonstrates the significance of the observation that the correlation dimension of the transit peptides of *Arabidopsis thaliana* is smaller than those of *Saccharomyces cerevisiae* and *Mus musculus*. As a kind of control, the null-hypothesis that the means of the correlation dimension of the transit peptides of *Saccharomyces cerevisiae* and *Mus musculus* are equal, is analogously tested. Remarkably, the resulting p-value is 0.1075 in this instance, which does not even allow to reject the null-hypothesis at a significance level of only 10%. Thus, our analysis based on correlation dimension does not reveal significant differences between *Saccharomyces cerevisiae* and *Mus musculus*.

Effect of data set variation

In order to examine if the results described so far have been biased by certain parameters of the data examined, the analysis was repeated with modified data sets. The first modification concerns the size of the selected transit peptides. In the original analysis, we have taken the N-terminal 50 amino acid residues of each protein as the mitochondrial targeting signal, because the exact size of the transit peptides was only in few cases experimentally determined. Several transit peptides are, however, shorter than 50 residues and it must therefore be assumed that the data sets include also significant amounts of mature protein sequences. This might influence the

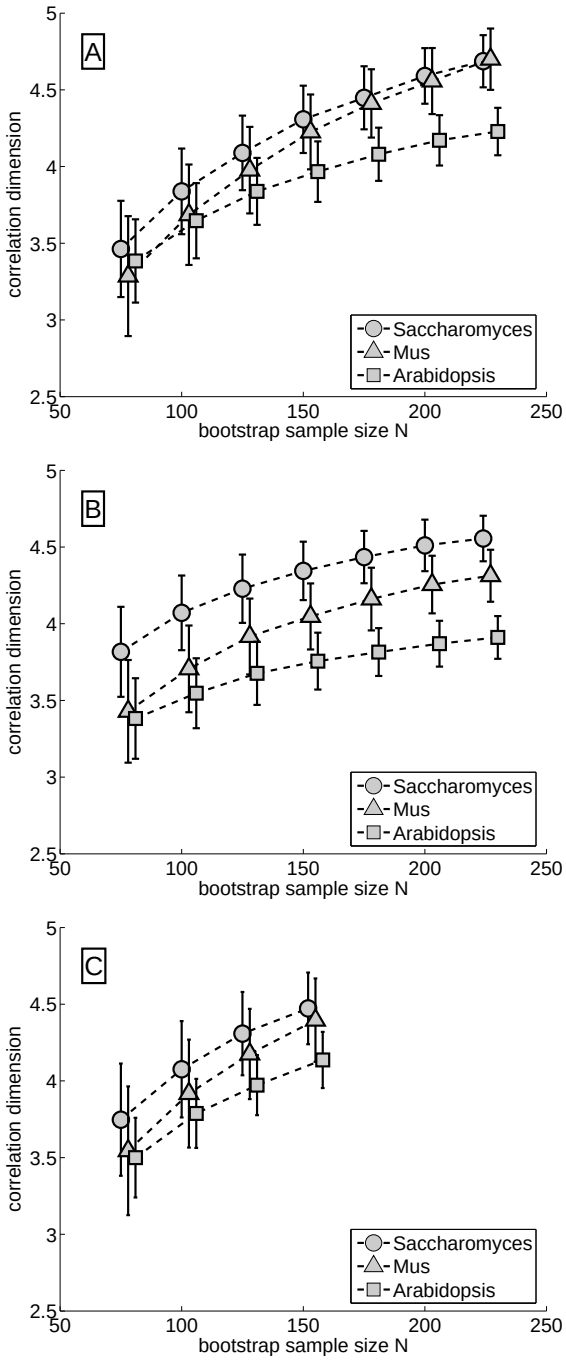


Figure 5.7: Dependency of correlation dimension on the sample size. The data shown are means and standard deviation calculated from 1000 bootstrap samples each. A: Original data set with the 50 N-terminal amino acid residues taken as transit peptides. B: Data entries as in (A), but taking only the 40 N-terminal residues as transit peptides. C: Data set as in (A), but devoid of homologous proteins.

outcome, because mature protein sequences are presumably exposed to completely different selective pressure than transit peptides. In order to reduce the potential effect of such mature sequences, we have performed the analysis also with data sets containing the N-terminal 40 residues of each protein only. Though it will lead to C-terminal truncation of those transit peptides that exceed 40 residues, it will first and foremost reduce the contamination with mature protein sequences. Bootstrap analysis performed with these new data sets yields essentially similar results as described above: the transit peptides of *Arabidopsis thaliana* have a significantly lower correlation dimension than those of *Saccharomyces cerevisiae* and *Mus musculus* (figure 5.7B). Please note that the absolute values of the correlation dimension for transit peptides of length 40 are lower than for those of length 50, because shortening of the assumed transit peptides unevitably decreases the variability in the data, which in turn leads to lower absolute values of the correlation dimension.

A second parameter, which might influence the degree of correlation dimension in a given data set are homologous proteins, which are generally the result of gene duplication and therefore bound to be quite closely related, even in their transit peptide sequences. If the number of homologous proteins within one data set differs significantly from those of the other data sets, it might well have a strong influence on the calculated correlation dimension. In order to take this possibility into account, we have strived to eliminate homologous proteins from all three data sets. For this purpose, we computed alignment scores between the full protein sequences within each data set and built groups using the single linkage algorithm [123]. The groups are built such that no sequence entry from two different groups exceeds the low sequence similarity score of 30[2]. Thus, elements of different groups have quite divergent sequences and are considered to be non-homologous. For the subsequent bootstrap experiments in which N groups are picked randomly, only one representant from each group is randomly chosen. Thus, each bootstrap sample contains at most one entry from a given group which prevents that homologous entries are present when computing the correlation dimension of a bootstrap sample. Using the cut off value of 30 for sequence similarity, 154 groups are defined for the smallest data set (Arabidopsis), which in turn also limits the maximal sample size for this experiment to 154. The results show that even after elimination of duplicates the transit peptides of *Arabidopsis thaliana* have a lower correlation dimension than those of *Saccharomyces cerevisiae* and *Mus musculus* (figure 5.7C). Due to the smaller maximal sample size of 154, the difference is not as pronounced as with the maximal sample size of 224 of the original experiment (figure 5.7A) but it is still statistically significant with numerically zero p-values. Actually, if identical sample sizes are compared between the different experiments, the differences in variability between the transit peptides of Arabidopsis, mouse and yeast are comparable in the two experiments (compare figures 5.7A and C). The absolute values of the correlation dimensions in the data sets devoid of homologous proteins are slightly increased as compared to the corresponding values in the original experiments though, due to the lack of redundancy

[2]ClustalW assigns to each pair of sequences a score between 100 (very similar) and 0 (not similar).

Table 5.1: Normalized usage frequencies of amino acids in the total mitochondrial protein sequences.

AS	*Arabidopsis thaliana*	*Mus musculus*	*Saccharomyces cerevisiae*
A	0.0764	0.0833	0.0635
C	0.0174	0.0176	0.0107
D	0.0528	0.0462	0.0520
E	0.0670	0.0639	0.0619
F	0.0407	0.0371	0.0424
G	0.0711	0.0721	0.0577
H	0.0209	0.0255	0.0213
I	0.0564	0.0476	0.0644
K	0.0665	0.0577	0.0814
L	0.0947	0.1030	0.0985
M	0.0304	0.0241	0.0219
N	0.0398	0.0316	0.0534
P	0.0421	0.0544	0.0453
Q	0.0289	0.0424	0.0380
R	0.0545	0.0628	0.0505
S	0.0788	0.0677	0.0775
T	0.0504	0.0519	0.0572
V	0.0725	0.0707	0.0602
W	0.0096	0.0131	0.0104
Y	0.0291	0.0273	0.0319

in the data sets, which slightly increases the variability and, in turn, the correlation dimension.

Finally, the usage of amino acid residues in total protein sequences (transit peptide plus passenger protein) is analyzed to ensure that the observed effect is not caused by different amino acid preferences in the three organisms. The normalized usage frequencies are computed from pooled sequences, that is, all sequences are concatenated and the normalized frequencies are computed as the number of occurrences of a particular amino acid divided by the total length of the concatenated sequence.

The normalized usage frequencies are shown in Table 5.1. Except for Q and W, which are both quite rare amino acids in proteins and can thus not be responsible for the observed differences in correlation dimension, the normalized usage frequency of amino acid in *Arabidopsis thaliana* is close to those of *Saccharomyces cerevisiae* and *Mus musculus*. This is in line with the assumption that the overall usage of amino acids is similar in all three organisms and confirms that *Arabidopsis thaliana* has no general bias in the amino acid usage. Thus, it must be concluded that the significantly lower correlation dimension of mitochondrial transit peptides of *Arabidopsis thaliana* is a consequence of reduced degree of freedom in the composition of these protein transport signals.

5.2.2.4 Discussion

During the last decade, several algorithms were developed to predict the subcellular localization of proteins by examining their N-terminal targeting sequences. Examples are the neural network-based approaches TargetP [44] and Predotar [125]. Both examine the N-terminal 100 amino acids and learn from training examples to discriminate between mitochondrial transit peptides, chloroplast transit peptides and other transport signals. TargetP predicts a score for each of the first 100 amino acids giving a likelihood whether it represents a transport signal, and classifies on that basis the target organelle of the protein. Predotar uses information on net charge, hydrophobicity and amino acid distribution to predict the target organelle. Other approaches classifying proteins according to their targeting sequence are PSORT and MitoProtII. PSORT [103] is a rule-based expert system and computes the likelihood that a given protein belongs to a specific target. MitoProt II [27] can only distinguish between mitochondrial and non-mitochondrial transport signals.

Neither of these analyses has considered the particularities of plant transit peptides. Instead, the training data used by the described algorithms collect mitochondrial transit peptides from plants, animals and fungi within a single group. Thus, the potential differences between mitochondrial transit peptides of plants and animals or fungi are neglected. This position is supported in [44] by citing a cluster analysis [119], which found no species-correlated differences between mitochondrial transit peptides. However, the data basis of that study used only 14 transit peptides from plants among 144 transit peptides in total, which does not allow any statistical conclusions. In contrast, our results strongly suggest that there are species-dependent differences among mitochondrial transit peptides. Thus, the predictions obtained by TargetP and Predotar have to be reconsidered when analyzing sequences from plants. It is furthermore remarkable that both programs are used to annotate the transit peptides in the Uniprot/Swissprot database.

Information theoretic methods have already been used before to analyze biological phenomena. However, while those analyses often try to find commonalities between different sequences or regions of sequences using mutual information, e.g. to describe molecular coevolution [28], our approach is based on fractal dimension which detects complexity differences between data sets.

To our knowledge, our study is the first one investigating the hypothesis that mitochondrial transit peptides of plants are more specialized and consequently have less degrees of freedom than those of animals or fungi. This hypothesis is tested by estimating correlation dimension of sets of transit peptides from three example organisms. Our results show that the correlation dimension of transit peptides from *Arabidopsis thaliana* is significantly lower than that from *Mus musculus* and *Saccharomyces cerevisiae*, in line with the assumption that plant mitochondrial transit peptides are exposed to increased selective pressure concerning organelle specificity.

5.3 Clustering by fractal Dimension

Real data follow hidden patterns. In many cases, the subset of the data objects is not known that is associated with a specific pattern. Clustering is the task to find subsets in the data, the clusters, that have high intra-clusters similarity and low similarity between objects from different clusters. We extend the clustering concept from finding clustering with low intra-similarity to finding clusters with low correlation dimension.

Correlation dimension is complexity measure which depends on the growth of the correlation integral. The correlation integral is an average of the covered data proportions over many balls each centered at a particular data object. Thus, the correlation dimension captures only average growth behavior over these balls.

Here, we propose the idea of creating a local-growth model for each data object. This growth model depends, in principle, only on pairwise object distances and captures how each object "views" its local neighborhood. Using this model we can characterize each object x_i with two variables (d_i, c_i), where d_i is the *local dimensionality* of the object x_i, and c_i is the *local density*. Intuitively, d_i depends on the growth rate of the number of objects in the neighborhood of x_i, while c_i depends on the density of objects in the neighborhood of x_i.

Both variables are estimated from *local growth curves*. Local growth curves can be computed directly from the data. Our algorithms require only a limited number of nearest-neighbor (NN) queries. These are well-studied and several efficient algorithms exist to answer them. For each point x_i, the local growth curve of x_i is computed, and a line is fitted on a subset of the points of the curve that corresponds to a local neighborhood of x_i. Then, the local dimensionality d_i is defined as the slope of the fitted line, while the local density c_i is defined as the value of the fitted line for a specific radius r^*. We choose r^* so as to maximize the information captured by the set of feature pairs (d_i, c_i), in the sense of minimizing the correlation between d_i and c_i.

Using the local density and local dimensionality, each object x_i is represented by the feature pair (d_i, c_i). Therefore, we map our dataset in a two dimensional space. This has the following advantages. First, we can easily cluster the dataset using an off-the-shelf, two-dimensional clustering algorithm, like EM with a Gaussian mixture model. Second, in an interactive system, the number of clusters and the correct partition can be identified through visual inspection.

Our main contributions are the following:

- Drawing upon ideas from fractals, we propose a general way to characterize the *local* dimensionality of objects. Our definitions are topological and independent of the notion of a linear subspace. Our methods can be applied to datasets of arbitrary dimensionality and non-vector data. Our algorithms are independent of the number of dimensions in the original dataset.

- Our method maps the dataset into a 2-dimensional space. We show how to chose the feature pairs (d_i, c_i), so as to maximize the information they retain

about the dataset, and enhance the visual representation of the dataset.

- We demonstrate how local dimensionality and local density can be used to detect low dimensional m-flats and low-rank sub-matrices. Our algorithms can successfully detect low-dimensional manifolds embedded in high-dimensional spaces, even when they are spatially overlapping.

Additionally, our method does not assume that the objects lie in a vector space and can be also applied to non-vector data, when dimensionality is not directly obtainable from the data representation itself.

First, we briefly discuss related work, broadly divided in two categories: methods that use some notion of density, and methods based on intrinsic dimensionality.

Similar to our method, density-based clustering approaches also rely on local density information in order to partition the dataset.

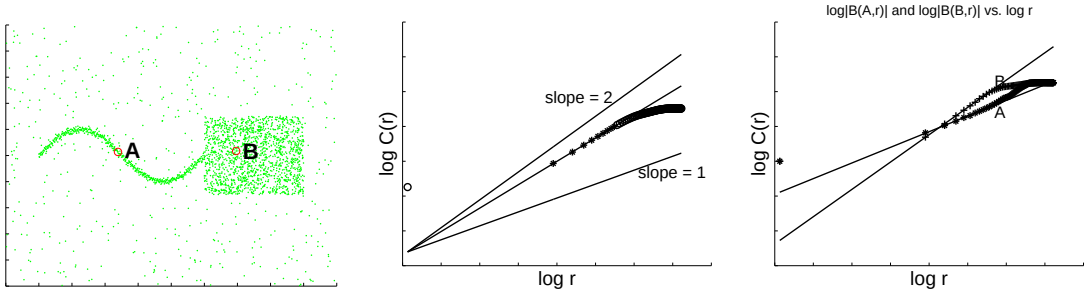
Hierarchical single linkage is a well-known method to find clusters with respect to density. To overcome problems in cases, when clusters are connected by small chains, popular variants like DBSCAN [45] and OPTICS [7] use a modified linkage hierarchy, where points within a cluster have to be reachable via core points (points having a certain minimum number of neighbors). DBSCAN computes a clustering corresponding to a cut in the linkage hierarchy, while OPTICS finds an ordering of the points from which a lower part of the linkage hierarchy can be deduced. Both algorithms fall short in case of clusters within clusters and the true hierarchy contains nodes with degree one. However, as our approach does not rely on spatial separation of the clusters, but focuses on detecting subsets with low intrinsic dimensionality, it can also deal with those cases.

Another density-based clustering method is DenClue [72], which employs kernel density estimation and uses density thresholds to define the clusters to be found. CLIQUE [6] is a density-based method that can also detect subspaces such that high-density clusters exist in them. However, it is grid-based and thus assumes that points lie in vector space. Furthermore, CLIQUE considers only hyper-rectangular clusters and projections parallel to the axes.

In projective clustering, one tries to find dense clusters in a projection of the original data space. Proclus [4] and DOC [115] search the space of axes-parallel projections to find good clusterings of the data. More advanced techniques like Orclus [5] and projective k-means [3] analyze eigenvalues of subsets of the data and can find arbitrary linear projections, in which points are clustered.

Concepts of intrinsic dimensionality from fractals have been successfully used in the database field for numerous problems, such as nearest-neighbor queries [110] and spatial query selectivity estimation [11, 47]. Recent results [84] discuss doubling dimension as measure for intrinsic dimensionality. The proposed algorithm works efficiently, when the intrinsic dimensionality is bounded.

Barbará et al. [9] propose a clustering approach that uses the fractal dimension (box-counting). It computes the fractal dimension of each individual cluster X and of $X \setminus x_i$ and puts x_i into the cluster for which the change is minimal. This requires some initial seed clusters, which may be difficult to guess.



(a) (b) (c)

Figure 5.8: A dataset that contains two subsets of different intrinsic dimensionality

Finally, LOCI [111] is an outlier detection method based on the local distribution of pairwise distances at multiple scales. Although the key concepts are related, LOCI focuses on an entirely different application and does not use the concept of intrinsic dimensionality in any way.

5.3.1 Clustering by Local Correlation Dimension

In this section we give an overview of our new clustering method. As we discussed before, the method draws upon and extends previous density-based algorithms, as well as concepts of intrinsic dimensionality. The two key measures it uses are *local density* and *local dimensionality*. Both are obtained by fitting a line on a subset of points of the *local growth curve*.

We describe local growth curves, and we explain how local density and local dimensionality are computed, and how they are used for clustering the dataset.

5.3.1.1 Local Correlation Dimension

The function $\hat{C}(r)$ as defined in Equation (5.5) computes the average fraction of neighbors of an object within distance r, where the average is taken over *all* non-empty balls induced by the objects in the dataset. However, due to averaging, if the dataset is non-homogeneous, the estimated correlation dimension will not reflect the "true" dimensionality of the data. Figure 5.8 illustrates this point. Figure 5.8(a) shows a dataset with two distinct subsets of points: in the first subset the points lie on a 1-D curve, while the second subset consists of a cloud of 2-D points. As a consequence of taking averages, the correlation dimension of the whole dataset is somewhere between one and two. Figure 5.8(b) shows the line fitted to the $C(r)$ curve and, for comparison, lines with slopes one and two.

Therefore, in the case that a dataset consists of subsets with different intrinsic dimensionality, the correlation dimension of a dataset does not correctly characterize the dimensionality of the dataset. To overcome this problem we extend the definition of correlation dimension for each point in the dataset.

Definition 5 (Local-Growth Curve) *For each object $x_i \in X = \{x_1, x_2, \ldots\}$ and δ_{ij} is the distance between x_i and x_j, we define the* local-growth curve*, to be the function of r, $G_i : \mathbb{R} \to \mathbb{N}$ that computes the fraction of neighbors of x_i in a ball of radius r,*

$$G_i(r) = \lim_{N \to \infty} \frac{1}{N} \sum_{j=1}^{N} \Theta(r - \delta_{ij}).$$

The local growth curve G_i describes the density of the local neighborhood of x_i for all distances r. In addition, the G_i curve contains information about the growth rate of the number of neighbors of x_i. We can now define the *local-correlation dimension* (or local dimension) of point x_i.

Definition 6 (Local-Correlation Dimension) *We define the* local-correlation dimension *d_i of object x_i, as*

$$d_i = \lim_{r, r' \to 0} \frac{\log[G_i(r)/G_i(r')]}{\log[r/r']}. \tag{5.8}$$

As in the case of correlation dimension, when dealing with finite sets, we define the local growth curve to be $G_i(r) = \frac{1}{N} \sum_{j=1}^{N} \Theta(r - \delta_{ij})$, and we compute the local-correlation dimension d_i of an object x_i by the slope of G_i curve in log-log scale. Notice that for finite sets the G_i curve is step-wise – its value chances only when the radius grows to include the next neighbor of an object. As a result, the local-growth curve can be represented without loss of information by specifying its value on a finite set of radii $\mathcal{D}_i \subset \mathbb{R}$, which we call the *domain* of G_i.

5.3.1.2 Local representation

We now describe how to use the local growth curves and the local-correlation dimension in order to represent the dataset.

Let $X = \{x_1, \ldots, x_N\}$ be a finite dataset of N objects for which the distances δ_{ij} are known. For each object x_i we define its domain $\mathcal{D}_i$, and we compute the local growth curve G_i. We then take the G_i curve in log-log scale, and we find the line that fits it best, in a least squares sense. We use L_i to denote this line, and we call it the *linear growth model* for object x_i.

Definition 7 (Linear Growth Model) *We refer to the line*

$$L_i(\log r) = d_i \log r + b_i$$

as the Linear Growth Model *for the object x_i.*

The slope d_i of the line L_i is an estimate of the local-correlation dimension of object x_i. The value b_i is the coefficient computed by the line fitting. Using the linear growth model, we can now represent the object x_i using just two numbers. The first number is the value d_i, the *local dimension* of the object x_i. The second

number is denoted by $c_i = L_i(\log r^*)$, and it corresponds to the density of the dataset in a ball of radius r^*, centered on x_i *as it is estimated* by the linear growth model L_i for object x_i. For example, for $r^* = 1$, $c_i = b_i$. We defer the discussion about the choice of value for r^* to Section 5.3.2, Lemma 2. We call c_i the *local density* of the object x_i. We write $l(i) = (d_i, c_i)$ to denote the representation of x_i by these two parameters.

Definition 8 (Local representation) *The mapping $l(i) = (d_i, c_i)$ is called the* local representation *of x_i, where $c_i = L_i(\log r^*)$.*

To illustrate the intuition behind local representation, consider two specific points A and B that come from the two different subsets in the example of Figure 5.8. Panel 5.8(c) shows $G_i(r)$ for $x_i = A$ and $x_i = B$ together with the fitted lines. In our example, the local dimensionality of point A is 1.20 and that of point B is 1.98. Therefore, we are able to distinguish the subsets with different intrinsic dimensionality using the local dimensionality d_i.

As we will explain in the next section, it is meaningful to ignore the parts of the local growth curve that correspond to very small and very large radii. The reason is that, for small r, the value of $G_i(r)$ is sensitive to local noise effects. Thus, ignoring small radii improves the robustness of the estimated dimension. On the other hand, for large r too many points contribute to the value of $G_i(r)$. Therefore, G_i does not capture local-neighborhood structure around x_i any more; most curves look identical. For these reasons we restrict the domain $\mathcal{D}_i$ of $G_i(r)$ to a smaller subset $\mathcal{F}_i \subseteq \mathcal{D}_i$, which we call the *fitting set* of $G_i(r)$, and it is precisely the range over which we fit the linear-growth model L_i. The details of how the fitting set is determined are discussed in Section 5.3.2.2.

5.3.1.3 Overall clustering

The final step of our method is to detect clusters of points that form low-dimensional manifolds in the ambient space of the dataset. Taking advantage of the simplicity of the local representation $l(x_i) = (d_i, c_i)$, we can perform this step using a standard clustering algorithm. Assuming that the local representation maintains well the information about the local density and the local dimensionality of the -s, the clustering process is relatively easy since it is an operation on two-dimensional data. For our experiments we used the standard EM algorithm with full covariance matrices. Furthermore, the 2-D representation offers an informative visualization of the dataset. In an interactive system it is usually easy to determine the underlying clusters in the dataset.

Finally, we note that the output of our algorithm can be further processed to discover dimensions of interest. First, in case that the algorithm places two manifolds of same dimension and density into one cluster, but the two manifolds do not intersect, then they can be separated by the single-linkage clustering. Second, in case of axis aligned subspaces, we can easily discover the attributes of interest by measuring the variance along each dimension. Finally, in case that we are interested in linear

subspaces, running PCA will reveal the directions of interest. All of these three tasks become significantly easier once the appropriate subset of objects has been identified by our method.

5.3.2 Analysis

In this section we discuss the properties of our definitions, and their implications in the overall approach. As our guide in this discussion we consider the simple cases of points lying on a 2-D grid and on a 1-D line.

5.3.2.1 Discussion and Examples

The definition of correlation dimension in Section 5.1.1 assumes that the size of the set X is infinite. For an infinite real line and an infinite real plane the dimensions are precisely 1 and 2, respectively. In practice, however, we deal with finite sets with finite extent, so we can only compute an estimate of the actual dimension. We will now study the effect of finiteness on the local representation of the objects by investigating two simple cases. The first dataset L consists of N one-dimensional points equally spaced on a line. The second dataset G consists of N two-dimensional points arranged on a grid. Ideally, the intrinsic dimensionality of the line should be one, and the dimensionality of the grid should be two. However, due to the finite size of the datasets, the estimated dimensionalities are different.

Consider the points in the set L and assume that the point x_i is located at position i of the real line. Consider one of the endpoints of the line, e.g., the leftmost point x_1 of the line. The local growth curve of x_1 is $G_1(r) = \frac{r}{N}$ (when computing $\sum_{j=2}^{N} \Theta(r-\delta_{1j})$ we do not count the point itself). For the point x_m in the middle of the line $m = N/2$, the local growth curve is $G_m(r) = \frac{2r}{N}$. In both of these cases, the local dimensionality of points x_1 and x_m is one, as expected. However, consider the point x_p that lies in position $p = N/4$. For radii $r = 1 \ldots \frac{N}{4}$, $G_p(r) = \frac{2r}{N}$, while for radii $r = \frac{N}{4} \ldots \frac{3N}{4}$, $G_p(r) = \frac{r}{N}$. Due to this change in the local growth curve, when fitting a line, the local dimensionality of the point x_p is underestimated. For example, for a line with 500 points, the local dimensionality is estimated to be around 0.87. The d_i values for all points are shown in Figure 5.9.

Determining the correct slope is even harder for the two-dimensional grid. Consider the point x_m in the middle of the grid. For simplicity we will assume that the distance between points is measured using the L_∞ norm. It is not hard to see that the local growth curve for this point is $G_m(r) = \frac{1}{N}((2r+1)^2-1) = \frac{1}{n}(4r^2+4r)$ (again, the point x_m is not counted in the computation). Due to the additive term $4r$ we need to have $r \to \infty$ in order for the local dimension d_m of x_m to tend to 2. In practice, this results in underestimating the dimension of the point. For a 50×50 grid the local dimension of the middle points is estimated to be close to 1.8. Furthermore, simple computations show that for a point x_s on the side of the grid, $G_s(r) = \frac{1}{N}(2r^2 + 3r)$, while for a point x_c on the corner of the grid, $G_c(r) = \frac{1}{n}(r^2 + 2r)$. Again, the local growth curve on the boundary of the grid is different from that inside the grid.

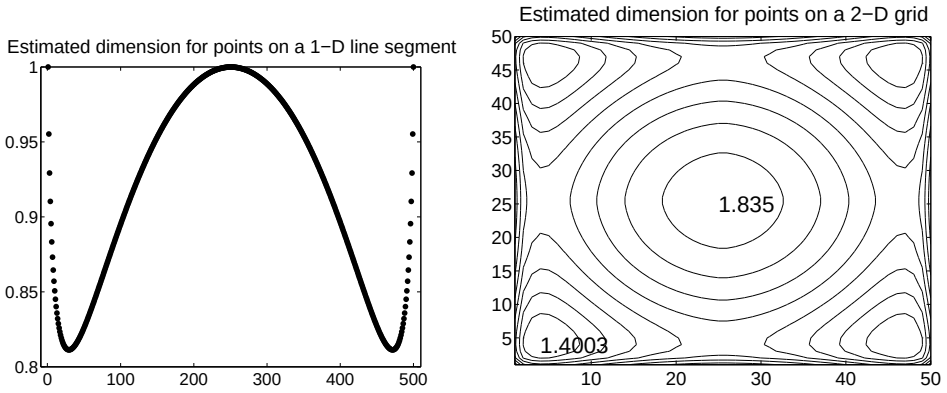


Figure 5.9: Boundary effects on the estimation of the correlation dimension.

Therefore, when the curve hits the boundary there is a change in the local growth curve, which results in further underestimation of the local dimension. An example with a 50×50 grid is shown in Figure 5.9. The contour lines show how the local dimension changes for different points of the grid. The maximum and minimum values are shown on the plot.

We next consider the case of a dataset consisting of a line embedded in a grid. We assume that the line consists of grid points which are replicated μ times. The value μ is the density of the line. For simplicity, assume that the line lies in the middle of the grid and it is parallel to one axis of the grid. Furthermore, in order to avoid dealing with boundary points, assume that the grid extends to infinity in all directions. For a point on the line x_ℓ it is not hard to show that the local growth curve is $G_\ell(r) = \frac{1}{N}((2r+1)^2 + \mu(2r+1) - 1)$. When the value μ is large enough compared to r the growth of $G_\ell(r)$ is dominated by the linear term. Of course as $r \to \infty$, the quadratic part becomes dominant. For a grid point, the growth is the same as before as long as the ball around the point has not reached the line. When the line is reached, the local growth curve becomes the same as for a line point (this is also due to the fact that we consider the L_∞ distance, and the line is parallel to the axis).

In the next subsection we will see how to address the issues raised by the previous examples. The idea is to compute the local dimensionality d_i of each object x_i by fitting a line not to the entire local growth curve G_i, but only on the fitting set $\mathcal{F}_i$. We discuss how to determine $\mathcal{F}_i$ next.

5.3.2.2 Determining the fitting set

An important issue in the definition of local growth curves is the domain of radii over which they are defined. An immediate idea is to define the curves over the interval ranging from the minimal pairwise distance up to the diameter of the dataset. Let

$\mathcal{R}$ denote this interval. This is the maximal interval over which the local growth curves can be defined. However, this approach is extreme, since for most points, the low part of the curve will be zero (balls with small radius contain no points), while the upper part of the curve will be one (balls with large radius contain the whole dataset). This will result in poor estimates for the local dimension of these points.

One approach for dealing with this problem is to restrict the definition of the local growth curves over an interval $[r_{\min}, r_{\max}] \subset \mathcal{R}$, which one might believe that captures the useful information of the local growth curve. However, this approach is also problematic when the density of the dataset differs in different regions of the space. Furthermore, for points that lie in large dimensional spaces, the minimum and maximum distances converge, so it is challenging to find a meaningful interval.

To address these issues, we choose to define a different domain $\mathcal{D}_i$ for each object x_i in the dataset. This domain is defined by growing a ball around x_i such that at each step we extend the ball to include (at least) one more neighbor. In other words, the domain $\mathcal{D}_i$ is precisely the set of radii $\{r_1, r_2, ..., r_n\}$, where r_k is the distance of x_i to its k-th nearest neighbor.

Following the discussion in Section 5.3.2.1, it becomes clear that it is beneficial to restrict the domain $\mathcal{D}_i$ by considering the distances only up to some k_{max}-th nearest neighbor, instead of all possible neighbors. This has the following advantages. First, it captures best the idea of locality upon which our approach is based. As it was demonstrated in the case of the line embedded in the grid, this can help discriminate between points that lie on different manifolds. Second, it helps in avoiding strong boundary effects, since less points hit the boundaries, and thus we can better estimate their "real" dimension. This is shown in Figures 5.10 *(a)*, *(b)*, and *(c)*, where by restricting the interval from above, we obtain a better estimation of the dimension for more points of the line and the grid.

We further restrict the object specific interval R_i from below, by considering only the neighbors that are no closer than the $k_{\min}$-th nearest neighbor. As discussed in Section 5.3.2.1, in the case of the grid this helps obtain a better estimate of the dimension. This becomes obvious when comparing the the figures *(b)* and *(c)* in Figure 5.10. Figure *(c)* is obtained by restricting the interval R_i from below, where we obtain an estimate of the dimension closer to 2.

Furthermore, for small values of k (i.e., for the very first nearest neighbors) the radius r_k might be affected by small local variations of the density of the points. Such density variations might include isolated points or unusually dense areas. Our point is illustrated in Figure 5.10*(d)*. We generated 100 points uniformly at random in a d-dimensional hypercube, for $d = 2$, 5, and 10. By repeating the process of random point generation 1000 times, we estimate the expected distance and the variance of the k-th nearest neighbor from a randomly selected point as a function of k. Figure 5.10*(d)* shows that the variance of the distances of the very first nearest neighbors is large. Therefore, by ignoring the k-th nearest neighbors for $k < k_{\min}$ we obtain a more robust estimation of the local dimension. We are now ready to summarize our observations with the following simple definition.

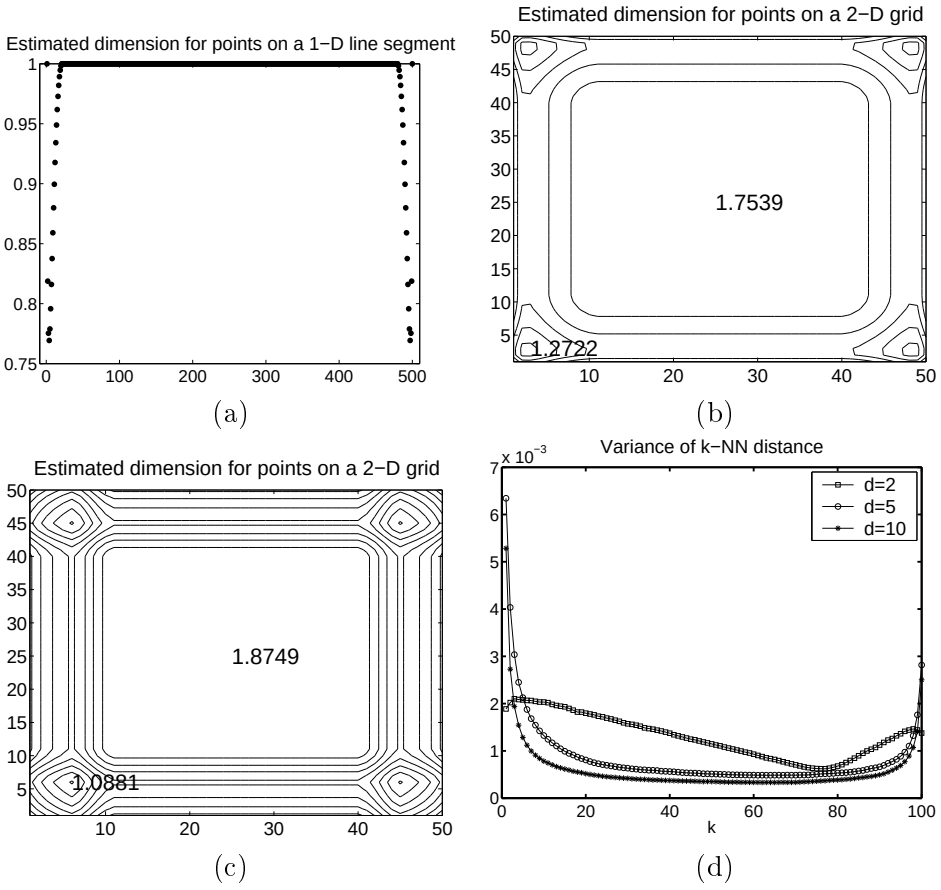


Figure 5.10: Restricting the fitting interval

Definition 9 (Fitting set) *The set of radii* $\mathcal{F}_i = \{r_k \in \mathcal{D}_i \mid r_{k_{\min}}^{(x_i)} \leq r_k \leq r_{k_{\max}}^{(x_i)}\}$, *where* $r_{k_{\min}}^{(x_i)}$ *and* $r_{k_{\max}}^{(x_i)}$ *are the distances of the* $k_{\min}$*-th and* $k_{\max}$*-th nearest neighbors of* x_i *is called the* fitting set.

In our experiments, we have found that the algorithm is not particularly sensitive in the choice of $k_{\min}$ and $k_{\max}$. For example, the values $k_{\min} = 0.01 \cdot N$ and $k_{\max} = 0.1 \cdot N$ give good quality of results in a wide variety of datasets.

5.3.2.3 Estimating local density

In this paragraph we derive an estimation for the radius r^*, which is used for computing the local density $c_i = L_i(\log r^*)$ for each x_i. For simplicity of notation we rewrite Equation (7) as $Y = d_i X + b_i$. We also write $X^* = \log r^*$ and $Y^* = d_i X^* + b_i = c_i$. Notice that by fitting a line to the curve G_i we obtain the parameters d_i and b_i. The goal is to compute the "best" choice of parameter $\log r^*$ that achieves the local representation $l(i) = (d_i, c_i)$ for each x_i.

The main observation is that by setting $\log r^* = +\infty$ in Equation (7), the resulting values c_i of local densities are perfectly positively correlated with the values d_i of local dimensions. The reason is that for $\log r^* = +\infty$ the ordering of c_i's is completely determined by the ordering of d_i's. Similarly, for $\log r^* \to -\infty$, the c_i's are perfectly negatively correlated with d_i's. Since our goal is to use the pair (d_i, c_i) that captures as much information for each x_i as possible, we would like to choose r^* such that the parameters d_i and c_i are uncorrelated. Based on this idea we can estimate the optimal radius r^* for the local representation $l(i)$. Note that it makes a considerable difference for the visualization as well as for automated clustering algorithms, whether the two-dimensional data (d_i, c_i) are correlated or not.

Lemma 2 *The value of* r^* *for which* d_i *and* c_i *are uncorrelated is given by*

$$\log r^* = -\frac{\sum (d_i - \bar{d})(b_i - \bar{b})}{\sum (d_i - \bar{d})^2}$$

Proof: The correlation between the variables d_i and c_i can be estimated by the coefficient

$$r_{dc} = \frac{\sum_i (d_i - \bar{d})(c_i - \bar{c})}{\sqrt{\sum_i (d_i - \bar{d})^2 \sum_i (c_i - \bar{c})^2}},$$

where $\bar{d} = \mathbb{E}[d_i]$ and $\bar{c} = \mathbb{E}[c_i]$ are the expectations of d_i's and c_i's, respectively. To make the correlation zero we need to choose r^* such that the numerator of r_{dc} is equal to zero. Let $\bar{b} = \mathbb{E}[b_i]$ be the expectation of b_i's. Since $c_i = d_i X^* + d_i$, by linearity of expectation we get $\bar{c} = \mathbb{E}[c_i] = \mathbb{E}[b_i + X^* d_i] = \bar{b} + X^* \bar{d}$. The numerator

of the correlation coefficient can now be written as

$$\begin{aligned}
&\sum (d_i - \bar{d})(c_i - \bar{c}) \\
&\quad = \sum (d_i - \bar{d})(b_i + X^* d_i - \bar{b} - X^* \bar{d}) \\
&\quad = \sum (d_i - \bar{d})(X^*(d_i - \bar{d}) + (b_i - \bar{b})) \\
&\quad = X^* \sum (d_i - \bar{d})^2 + \sum (d_i - \bar{d})(b_i - \bar{b}).
\end{aligned}$$

Setting $r_{dc} = 0$ gives the optimal value of $\log r^*$. □

5.3.3 Dimension-Induced-Clustering

We now present the Dimension Induced Clustering (DIC) algorithm. The objective of the algorithm is to partition points so that points in the same cluster lie on dense manifolds of the same dimension.

5.3.3.1 The DIC algorithm

The outline of the DIC algorithm is shown in Algorithm 2. The input to the algorithm is a set X of N elements, that we want to cluster in b clusters. In first step, the algorithm computes for each element x_i, the distance of x_i to its k-th nearest neighbor for all $k = k_{\min}, \ldots, k_{\max}$. The distances of the nearest neighbors of x_i specify completely the local growth curve G_i. By fitting the linear growth model L_i on G_i and by estimating the local density, as in Section 5.3.2.3, we compute the local representation $l(i) = (d_i, c_i)$ for each x_i. Thus, we map the set X into a two dimensional set X_{LR} that contains the local representation of all objects. The task now becomes to cluster the two-dimensional points in X_{LR}. Clustering in two dimensions is conceptually much simpler than clustering in high-dimensional spaces. The correct clustering can often be determined even by simple visual inspection. In the automated case applying an EM (Expectation Maximization) algorithm [63] for fitting a mixture of b Gaussian distributions on the data works well in most cases. If the set X consists of b sufficiently dense subsets that lie on manifolds of different dimension, which are sufficiently separated, the algorithm will be able to separate these subsets.

Algorithm 2 The DIC algorithm. Input: Dataset X of N objects, number of clusters b, Output: Clustering of X into b clusters.

1: **for** $i \in \{1, \ldots, N\}$ **do**
2: Compute k-th NN of x_i, for $k = k_{\min} \ldots k_{\max}$
3: Compute the local representation (d_i, c_i) of x_i.
4: **end for**
5: $X_{LR} = \{(d_1, c_1), \ldots, (d_N, c_N)\}$
6: Cluster the set X_{LR} into b clusters.

5.3.3.2 Efficiency of the DIC algorithm

The complexity of the DIC algorithm is dominated by the complexity of computing for every object x_i the distance to the k_{min} to $k_{\max}$ neighbors of the object x_i. The simple solution to this problem is to compute the distances between all objects in the set X, and for each object x_i sort the objects with respect to their distance from object x_i, and retrieve the necessary information. The time for computing all pairwise distances is $O(N^2)$.

A different approach is to construct an index for the elements in X that supports fast execution of k-nearest neighbor queries. In case that X consists of vector data, spatial index structures can be used for the efficient calculation such as [8, 12, 79]. In case of metric data the OMNI framework [50], or data structures like the M-tree [26] can be used. Since the computation of the local representation is inherently approximate, the use of approximative methods for k-nearest neighbor queries such as locality-sensitive hashing [76], is also possible.

Investigating the construction of the appropriate nearest neighbor index is beyond the scope of this study. We assume that such an index exists, and we use it as a black box for obtaining the distances of the k-th nearest-neighbor queries for $k = k_{\min}, \ldots, k_{\max}$. The efficiency of the DIC algorithm is determined by the efficiency of this index.

5.3.4 Experiments

In this section we study experimentally the properties and the performance of the DIC algorithm.

5.3.4.1 Applications and Datasets

We apply our algorithms on the following types of datasets.

Embedded m-flats: Consider a set X of n points in $\mathbb{R}^d$ that can be decomposed in two subsets U, and F, of size s and f respectively, where $N = s + f$. The points in U are distributed uniformly at random in $(0, 1)^d$. The points in F take values normally distributed around 0.5, with variance 0.01 in the first $d - m$ coordinates. In the last m coordinates, they take values uniformly distributed in $(0, 1)$. As $s, f \to \infty$ the intrinsic dimensionality of the sets U and F approaches d and m respectively. We call the set F an *m-flat*. The value m is the dimension of the m-flat. The set U can be thought of as an m-flat of dimension d, so we say that U has *full dimension*.

The objective of the algorithms is to partition the set X into sets U and F. We apply the DIC algorithm on X, requesting 2 clusters. We will demonstrate that the DIC algorithm, is able to return the sets F and U as the clusters even when m and d are relatively close. The flat F is the set of points with the smaller average intrinsic dimensionality.

Manifolds within manifolds: The setting is similar to the previous one, only this time the set X contains more than one m-flats of different dimensions. Namely, the

set X can be decomposed into sets U, $F_1, \ldots, F_p$, where U has full dimension d, and $F_1, F_2, \ldots, F_p$ are m-flats with dimensions $m_1 < m_2 < \ldots < m_p$ respectively. The m-flats are constructed as described above. Note that since for every flat F_i we always "fix" the first $d - m_i$ coordinates, the m-flats with lower dimension are embedded within the m-flats of higher dimensions. This results in creating a chain hierarchy of manifolds where every manifold is embedded in all the preceding ones in the chain. Again, we apply the DIC algorithm, requesting $p + 1$ clusters. When the dimensionalities of the m-flats are sufficiently separated, the algorithm returns as clusters that p flats and the set U. The average estimated dimension values for each set are ordered according to the actual dimension of the flats.

Low Rank Sub-Matrices: The input is an $n \times m$ matrix that takes values in $[0, 1]$. Within the matrix there is a collection of k rows and ℓ columns, such that the combinatorial $k \times \ell$ sub-matrix has low rank. The objective is to identify the rows and columns of this sub-matrix.

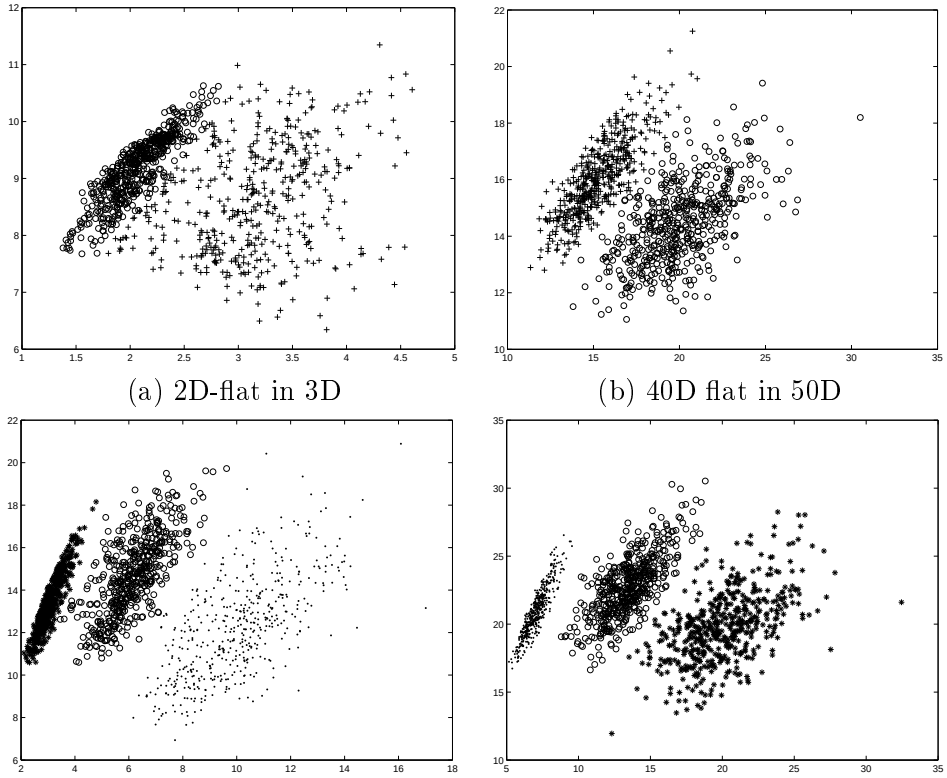
We generate such datasets as follows. First we generate a $k \times \ell$ matrix S of rank exactly r, where $r \ll \min\{n, m\}$. We then plant it in the matrix M. The remaining elements of M are generated uniformly at random, scaled so that the mean is zero and the standard deviation is one. Therefore, if we remove either the k rows, or the ℓ columns of matrix S from M, we obtain a matrix of rank $\min\{n - k, m\}$ and $\min\{n, m - \ell\}$ respectively. To this matrix we add a "noise" matrix X with entries distributed normally around 0, with variance 0.05.

In order to extract S from matrix M we apply the DIC algorithm in two steps. First we perform a clustering of the rows, and we identify the rows of the matrix S. The dimension of these rows is $m - \ell + r$, as opposed to m for the rest of the rows, so the DIC algorithm can easily identify them. We then cluster the columns of M. The dimension of the columns in S is $n - k + r$ as opposed to n for the rest of the columns, so again DIC manages to partition the rows. Given the rows and columns we can extract matrix S.

5.3.4.2 Experiments with the DIC algorithm

In this section we present experiments with the DIC algorithm on various datasets. In all runs of the algorithm, we set $k_{\min} = 10$, and $k_{\max} = 100$, two values that we observed that they work well in practice.

We start by experimenting with datasets that contain a single m-flat F, embedded in a space of higher dimension d, together with a set U of noise points distributed uniformly at random. The datasets are constructed as described in Section 5.3.4.1. Since the objective is to separate the sets F and U, we evaluate our algorithm by looking into the *total classification error* of the algorithm. The total classification error E_{tot} is computed as follows: We first compute the confusion matrix C whose C_{ij} entry contains the number of overlapping points between the i-th cluster of the ground truth and the j-th cluster of the clustering found by the algorithm. Then $E_{tot} = 1 - (\sum_i \max_j C_{ij})/n$.



(a) 2D-flat in 3D

(b) 40D flat in 50D

(c) 3D and 6D flats in 10D

(d) 10D and 20D flats in 30D

Figure 5.11: Discovering m-flats with DIC, (x: dimensionality, y: density)

Our experiments indicate that the DIC algorithm performs exceptionally well in this setting, even in the case that the dimension of the host space and the m-flat are very close, or if the m-flat is embedded in a high-dimensional space. Figure 5.11(a) plots the local representation of the data points when $d = 3$ and $m = 2$, and their clustering. Figure 5.11(b) shows the case where $d = 50$ and $m = 40$. In both cases, the size of the dataset is 1,000 points, of which 500 belong to the m-flat. We observe that the algorithm manages to identify the m-flats successfully. The total classification error is 8.1% in the first case, and 1.2% in the second case.

In order to better understand the performance of DIC, we performed a more detailed experiment, generating datasets with the dimension of the host space being $d = 2 \ldots 10$, and the dimension of the m-flat ranging from 1 to $d - 1$. In all cases, the dataset consists of 1,500 points, 500 of which belong to the m-flat. Table 5.2 reports the average classification error for 20 runs of the algorithm (the numbers are percentages). We observe that the classification error is never more than 39%, and this occurs in the case that the dimension of the host space and that of the m-flat differ by just one.

We now turn our attention to cases where there are more than one m-flats in the

Table 5.2: Classification error of discovering m-flat clusters

	1	2	3	4	5	6	7	8	9
2	9.2								
3	13.0	20.14							
4	14.9	1.53	29.28						
5	16.1	0.26	6.74	26.42					
6	15.4	0.08	0.68	6.99	31.1				
7	7.1	0.02	0.17	1.25	13.7	33.4			
8	10.5	0.00	0.02	0.41	2.1	14.6	36.3		
9	1.4	0	0.01	0.08	0.6	2.9	18.7	37.9	
10	7.4	0.01	0.01	0.04	0.2	0.9	4.2	20.7	38.3

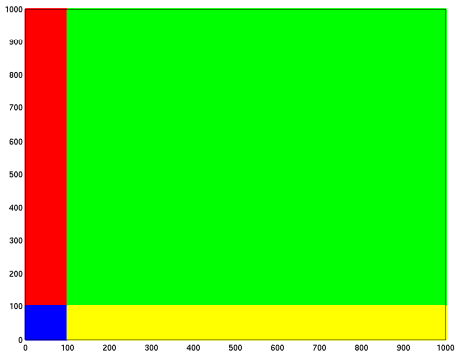


Figure 5.12: Discovering low rank matrices

dataset. Figures 5.11(c) and (d) show the plots of the local representations of two datasets that contain flats of different dimension. In the first case the dimension of the host space is $d = 10$ and the two manifolds have dimension $m_1 = 3$, and $m_2 = 6$. In the second case we have $(m_1, m_2, d) = (10, 20, 30)$, In both cases all three sets of points F_1, F_2, U contain 500 points each. We observe that the DIC algorithm manages to discriminate the three sets. The average classification error is 1.53% for the first case, and 0.51% for the second case, where the average is taken over 20 runs. Datasets with more than three flats are examined in the full version of the paper.

We also experiment with low rank matrices, trying to detect a (combinatorial) 100×100 submatrix of rank 2, within a 1000×1000 matrix. The algorithm proves to be quite successful, obtaining classification error just 0.16%, where the average is taken over 10 runs. In this case the large dimension of the matrix works in favor of our algorithm. The algorithm often achieves a perfect partition of the matrix (4 out of the 10 runs). A case where we obtain a perfect partition of the matrix is shown in Figure 5.12. The blue part of the figure shows the correctly found low rank submatrix, the red and yellow areas correspond to the attached random columns and rows and the green part totally belongs to the random matrix.

5.3.4.3 Comparison with OPTICS

In this subsection we compare our algorithm with OPTICS on the task of finding m-flats within noise. OPTICS takes as input the parameter ϵ_{max} which is the maximum linkage distance, and it produces an ordered visualization of the points in the dataset from which the lower part of a cluster hierarchy can be derived. In order to get rid of the dependency from ϵ_{max} we set it in all cases to the maximum distance in the particular data set, so that OPTICS computes the whole hierarchy.

The primary output of the OPTICS algorithm is a plot. The x axis of the plot shows the indices of the data points in the ordering produced by OPTICS. The y axis in the visualization is reachability distance, which is small if the density at that point is high. The computed hierarchy by OPTICS is similar to a single linkage hierarchy. The plot produced by OPTICS defines clusters as valleys. A valley is defined by a horizontal cutting line, which is chosen by the user. In case of sub-clusters within larger clusters, the plot by OPTICS consists of a large valley, which includes at the bottom smaller valleys divided by small hills.

The OPTICS algorithm assumes that each cluster in the true hierarchy consists of at least two sub-clusters. However, in case of m-flats embedded in other m-flats of higher dimension, this is not true. But one can still look for knees at the right side of a valley in the visualization plot of OPTICS, as it is shown in Figure 5.13(a). This allows to specify a cutoff value for the hierarchical algorithm. The cutting line should be set to the beginning of the knee. Note that in case that we have multiple m-flats, one embedded within the other it is not possible for OPTICS to identify all m-flats using a single cutoff value.

In our comparison with OPTICS we consider datasets where a single m-flat is embedded in a higher dimensional space. The datasets contain 1,000 points, while both the m-flat, as well as the noise set (containing points, uniformly distributed in the full-dimensional space) consists of 500 points each. The following data sets are generated: 2D-flat in 3D, 3D-flat in 5D, 5D-flat in 8D, 6D-flat in 10D, 40D-flat in 50D, and 90D-flat in 100D.

Figure 5.13(a) shows the plot generated by OPTICS for the 2D-flat in 3D. The knee at the right side of the lowest valley is clearly visible and so we choose the cutting value to be 0.09. However, for data with dimension larger than 10, although the algorithm produces the correct ordering with most of the points of the m-flat being in the beginning of the ordering, the knee is not longer visible. An example is shown in figure 5.13(b). Therefore, we could not compute a clustering with OPTICS for the last two high-dimensional data set. Any value seems equally good, resulting in arbitrarily good, or bad results. The same problem arises when trying to use OPTICS for identifying low-rank sub-matrices.

In Table 5.3 we compare the classification error of the clusterings found by OPTICS and the DIC algorithm. The classification errors are similar, with DIC being a little more accurate. The main conclusion from this experiment is that the density-based clustering method OPTICS fails for high-dimensional data. In such cases density alone is not sensitive enough to reveal the structure of the data. Furthermore, the

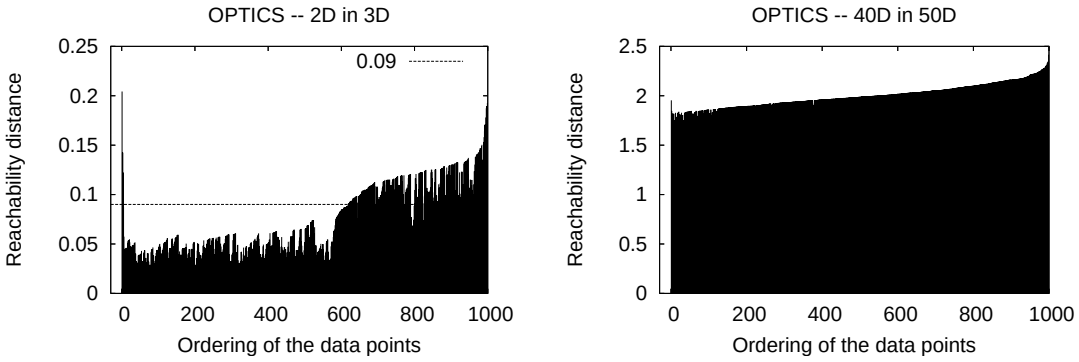


Figure 5.13: OPTICS plots for (a) 2D-flat within 3D noise (b) 40D-flat within 50D noise

Table 5.3: Classification Error of OPTICS and DIC

Data	E_{tot}(OPTICS)	E_{tot}(DIC)
2D in 3D	0.115	0.077
3D in 5D	0.045	0.029
5D in 8D	0.087	0.024
6D in 10D	0.045	0.010
40D in 50D	n.a.	0.010
90D in 100D	n.a.	0.072

OPTICS algorithm requires careful fine-tuning of the parameters in order to produce a meaningful clustering, as opposed to the DIC algorithm which has only few, easy to set parameters. Finally, we note that the DIC algorithm is order independent, as opposed to OPTICS which is sensitive to the order in which the points are visited.

The case of multiple m-flats is considerably more difficult for OPTICS. Multiple m-flats will appear as a single valley since they differ only by density and not by location, which means that one has to find multiple knees. However, the visibility of the knees degrades as the dimensionality of the m-flats increases.

5.4 Summary

We proposed solutions for two problems in data analysis: (i) testing given subsets of data, whether they differ significantly in correlation dimension and (ii) finding clusters in the dataset such that the objects in the same cluster have similar correlation dimension and data density. Correlation dimension is widely applicable, because only distances or similarities between the data objects are needed. This allows the application of data analysis by fractal dimension to both data described by feature

vectors as well as non-vector data.

An application corresponding to the first problem setting is the analysis of mitochondrial transit peptides. The biological research question was whether mitochondrial transit peptides in plants are due to a second target organelle, the chloroplasts, more specialized and consequently have less degrees of freedom than the mitochondrial transit peptides in animal or fungi. We computed correlation dimension of both datasets from the pairwise sequence similarities and observed that mitochondrial transit peptides in plants have significantly lower correlation dimension than mitochondrial transit peptides in animals or fungi.

Cluster analysis by fractal dimension computes a local correlation integral for each data object. The local correlation integral is a function that counts the number of other objects from the data set in a ball of a given radius. The growth rate of the local correlation integral in the log-log space (logarithm of radius versus logarithm of local correlation integral) is the local correlation dimension of a data object. Furthermore, local density is measured as well. Objects with similar local correlation dimension and local density are assigned to the same cluster. The method can successfully distinguish clouds of points randomly spread on hyperplanes in with different dimensionality. The hyperplanes can be arbitrarily arranged in a high-dimensional data space.

Bibliography

[1] A. Tsipouras, J. Ondeyka, C. Dufresne et al. Using similarity searches over databases of estimated c-13 nmr spectra for structure identification of natural products. *Analytica Chimica Acta*, 316:161–171, 1995.

[2] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422:198–207, 2003.

[3] Pankaj K. Agarwal and Nabil H. Mustafa. k-means projective clustering. In *PODS*, 2004.

[4] Charu C. Aggarwal, Cecilia Magdalena Procopiuc, Joel L. Wolf, Philip S. Yu, and Jong Soo Park. Fast algorithms for projected clustering. In *Proc. SIGMOD*. ACM, 1999.

[5] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proc. SIGMOD*. ACM, 2000.

[6] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. SIGMOD*, 1998.

[7] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings SIGMOD'99*, pages 49–60. ACM Press, 1999.

[8] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *JACM*, 45(6), 1998.

[9] Daniel Barbará and Ping Chen. Using the fractal dimension to cluster datasets. In *Proc. KDD*, 2000.

[10] A. S. Barros and D. N. Rutledge. Segmented principal component transform-principal component analysis. *Chemometrics & Intelligent Laboratory Systems*, 78:125–137, 2005.

[11] Alberto Belussi and Christos Faloutsos. Self-spacial join selectivity estimation using fractal concepts. *ACM TOIS*, 16(2), 1998.

[12] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An index structure for high-dimensional data. In *Proc. VLDB*, 1996.

[13] J.C. Bezdek. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer Academic Pub, 1999.

[14] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, New York, NY, USA, 2003. ACM Press.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[16] D. Blei and J. Lafferty. Correlated topic models. *Advances in Neural Information Processing Systems*, 18, 2006.

[17] David M. Blei and John D. Lafferty. Dynamic topic models. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 113–120, New York, NY, USA, 2006. ACM Press.

[18] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[19] D.M. Blei and M.I. Jordan. Variational inference for dirichlet process mixtures. *Journal of Bayesian Analysis*, 1(1):121–144, 2005.

[20] H. H. Bock. *Automatic Classification*. Vandenhoeck and Ruprecht, 1974.

[21] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Selected papers from the sixth international conference on World Wide Web*, pages 1157–1166, Essex, UK, 1997. Elsevier Science Publishers Ltd.

[22] L. Cai and T. Hofmann. Text categorization by boosting automatically extracted concepts. In *SIGIR '03*, 2003.

[23] C. Carrie, E Giraud, and J. Whelan. Protein transport in organelles: Dual targeting of proteins to mitochondria and chloroplasts. *FEBS J.*, 276:1187–1195, 2009.

[24] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, New York, NY, USA, 2002. ACM.

[25] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, 2002.

[26] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. VLDB*, 1997.

[27] M G Claros and P Vincens. Computational method to predict mitochondrially imported proteins and their targeting sequences. *Eur J Biochem*, 241(3):779–786, Nov 1996.

[28] Francisco M. Codoñer and Mario A. Fares. Why should we care about molecular coevolution? *Evolutionary Bioinformatics*, 4:29–38, 2008.

[29] Edith Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Syst. Sci.*, 55(3):441–453, 1997.

[30] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav K. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Symposium on Interactive 3D Graphics*, pages 189–196, 218, 1995.

[31] Jack G. Conrad, Xi S. Guo, and Cindy P. Schriber. Online duplicate document detection: signature reliability in a dynamic retrieval environment. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 443–452, New York, NY, USA, 2003. ACM Press.

[32] Katie Cottingham. Turning data graveyards into gold mines. *Journal of Proteome Research*, 7(1):22, 2008.

[33] R. Craig, J. P. Cortens, and R. C. Beavis. Open source system for analyzing, validating, and storing protein identification data. *J Proteome Res*, 3:1234–1242, 2004.

[34] M. W. Davis and W. C. Ogden. Free resources and advanced alignment for cross-language text retrieval. *TREC*, pages 385–395, 1997.

[35] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[36] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[37] Fan Deng and Davood Rafiei. Approximately detecting duplicates for streaming data using stable bloom filters. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 25–36, New York, NY, USA, 2006. ACM Press.

[38] F. Desiere, E. W. Deutsch, A. I. Nesvizhskii, and P. Mallick. Integration with the human genome of peptide sequences obtained by high-throughput mass spectrometry. *Genome Biol*, 6:R9, 2005.

[39] B. Domon and R. Aebersold. Mass spectrometry and protein analysis. *Science*, 312:212–217, 2006.

[40] S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23:229–236, 1991.

[41] Robert C Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucl. Acids Res.*, 32(5):1792–1797, 2004.

[42] B Efron and R J Tibshirani. *An introduction to the bootstrap.* Chapman & Hall, 1998.

[43] Björn Egert, Steffen Neumann, and Alexander Hinneburg. Fast approximate duplicate detection for 2d-nmr spectra. In *Workshop on Data Integration in the Life Sciences, DILS*, pages 139–155. Springer, LNCS, 2007.

[44] O Emanuelsson, H Nielsen, S Brunak, and G. von Heijne. Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *J Mol Biol*, 300(4):1005–1016, Jul 2000.

[45] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusteris in large spatial databases with noise. In *Proc. KDD*, 1996.

[46] K J Falconer. *Fractal Geometry.* Wiley, 1990.

[47] Christos Faloutsos and Ibrahim Kamel. Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. In *Proc. PODS*, 1994.

[48] U. Feige. A threshold of ln n for approximating set cover (preliminary version). *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 314–318, 1996.

[49] Ronen Feldman and Ido Dagan. Kdt - knowledge discovery in texts. In *Proceedings of the First International Conference on Knowledge Discovery (KDD)*, pages 112–117, 1995.

[50] Roberto F. Santos Filho, Agma J. M. Traina, Jr. Caetano Traina, and Christos Faloutsos. Similarity search without tears: The omni family of all-purpose access methods. In *Proc. ICDE*, 2001.

[51] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[52] K. Fukunaga and L.D. Hostler. The estimation of the gradient of a density function, with application in pattern recognition. *IEEE Trans. Info. Thy.*, 21:32–40, 1975.

[53] G. W. Furnas, S. Deerwester, S.T. Dumais, and T. K. Landauer. Information retrieval using a singular value decomposition model of latent semantic structure. *SIGIR '88: Proceedings of the 11th annual Int SIGIR conference on Research and Development in Information Retrieval*, pages 465–480, 1988.

[54] Peter V. Gehler, Alex D. Holub, and Max Welling. The rate adapting poisson model for information retrieval and object recognition. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 337–344, New York, NY, USA, 2006. ACM Press.

[55] Aristides Gionis, Dimitrios Gunopulos, and Nick Koudas. Efficient and tunable similar set retrieval. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 247–258, New York, NY, USA, 2001. ACM Press.

[56] Aristides Gionis, Alexander Hinneburg, Spiros Papadimitriou, and Panayiotis Tsaparas. Dimension induced clustering. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 51–60, New York, NY, USA, 2005. ACM.

[57] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB'99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, CA USA, 1999. Morgan Kaufmann Publishers Inc.

[58] Mark Girolami and Ata Kaban. On an equivalence between plsi and lda. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 433–434, New York, NY, USA, 2003. ACM Press.

[59] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[60] Daniel Gomes, André L. Santos, and Mário J. Silva. Managing duplicates in a web archive. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 818–825, New York, NY, USA, 2006. ACM Press.

[61] P Grassberger and I Procaccia. Estimation of the Kolmogorov entropy from a chaotic signal. *Phys. Rev. A*, 28(4):2591–2593, Oct 1983.

[62] M. Hamacher, R. Apweiler, G. Arnold, and A. Becker. Hupo brain proteome project: summary of the pilot phase and introduction of a comprehensive data reprocessing strategy. *Proteomics*, 6:4890–4898, 2006.

[63] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[64] Monika Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291, New York, NY, USA, 2006. ACM Press.

[65] M. Herbin, N. Bonnet, and P. Vautrot. Estimation of the number of clusters and influence zones. *Pattern Recognition Letters*, 22:1557–1568, 2001.

[66] H. Hermjakob and R. Apweiler. The proteomics identifications database (pride) and the proteomexchange consortium: making proteomics data accessible. *Expert Rev Proteomics*, 3:1–3, 2006.

[67] Mauricio A. Hernandez and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, 1998.

[68] A. Hinneburg and D.A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings KDD'98*, pages 58–65. AAAI Press, 1998.

[69] Alexander Hinneburg, Björn Egert, and Andrea Porzel. Duplicate detection of 2d-nmr spectra. *Journal of Integrative Bioinformatics*, 4(1):53, 2007.

[70] Alexander Hinneburg and Hans-Henning Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In *Proc. of International Symposium on Intelligent Data Analysis 2007 (IDA'07)*. LNAI Springer, 2007.

[71] Alexander Hinneburg, Hans-Henning Gabriel, and Andre Gohr. Bayesian folding-in with dirichlet kernels for plsi. In *IEEE International Conference on Data Mining, ICDM'07*, 2007.

[72] Alexander Hinneburg and Daniel A. Keim. A general approach to clustering in large databases with noise. *Knowledge and Information Systems (KAIS)*, 5(4):387–415, 2003.

[73] Alexander Hinneburg, Andrea Porzel, and Karina Wolfram. An evaluation of text retrieval methods for similarity search of multi-dimensional nmr-spectra. In *Bioinformatics Research and Development, BIRD 07*, volume 4414/2007 of *Lecture Notes in Computer Science*, pages 424–438. Springer, LNBI, 2007.

[74] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196, 2001.

[75] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM Press.

[76] P. Indyk and R. Motwani. Approximate nearest neighbor - towards removing the curse of dimensionality. In *Proceedings of the 30th Symposium on Theory of Computing*, pages 604–613, 1998.

[77] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. Web usage mining based on probabilistic latent semantic analysis. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 197–205, New York, NY, USA, 2004. ACM Press.

[78] P. Jones, R. G. Cote, L. Martens, and A. F. Quinn. Pride: a public repository of protein and peptide identifications for the proteomics community. *Nucleic Acids Res*, 34:D659–663, 2006.

[79] Norio Katayama and Shin'ichi Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *SIGMOD*, 1997.

[80] Yan Ke, Rahul Sukthankar, and Larry Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, New York, NY, USA, 2004. ACM Press.

[81] James Keeler. *Understanding NMR Spectroscopy*. University of Cambridge, 2002.

[82] Sebastian Klie, Lennart Martens, Juan Antonio Vizcaino, Richard Cote, Phil Jones, Rolf Apweiler, Alexander Hinneburg, and Henning Hermjakob. An application of latent topic document analysis to large-scale proteomics databases. In *Proc. of German Bioinformatics Conference 2007 (GCB'07)*. LNBI Springer, 2007.

[83] Sebastian Klie, Lennart Martens, Juan Antonio Vizcano, Richard Cote, Phil Jones, Rolf Apweiler, Alexander Hinneburg, and Henning Hermjakob. Analyzing large-scale proteomics projects with latent semantic indexing. *Journal of Proteome Research*, 7:182–191, 2008.

[84] Robert Krauthgamer and James R. Lee. The black-box complexity of nearest neighbor search. In *Proc. ICALP*, 2004.

[85] P. Krishnan, N. J. Kruger, and R. G. Ratcliffe. Metabolite fingerprinting and profiling in plants using nmr. *Journal of Experimental Botany*, 56:255–265, 2005.

[86] F. Kun, H. Sorgeb, K. Sailera, G. Bardosa, and W. Greiner. Sandbox method for factorial moments and anomalous fractal dimensions. *Physics Letters B*, 355:349–355, 1995.

[87] T. K. Landauer, P. W. Foltz, and D. Laham. Introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.

[88] Michael Levandowsky and David Winter. Distance between sets. *Nature*, 234:34 – 35, 1971.

[89] X. Li, Y. Gong, Y. Wang, S. Wu, Y. Cai, P. He, Z. Lu, W. Ying, Y. Zhang, L. Jiao, H. He, Z. Zhang, F. He, X. Zhao, and X. Qian. Comparison of alternative analytical techniques for the characterisation of the human serum proteome in hupo plasma proteome project. *Proteomics*, 5:3423–3441, 2005.

[90] Alan H. Lipkus. A proof of the triangle inequality for the tanimoto distance. *Journal of Mathematical Chemistry*, 26(1-3):263–265, 1999.

[91] M. Farkas, J. Bendl, D. H. Welti et al. Similarity search for a h-1 nmr spectroscopic data base. *Analytica Chimica Acta*, 206:173–187, 1988.

[92] Mallick, Schirle, Chen, and Flory. Computational prediction of proteotypic peptides for quantitative proteomics. *Nat Biotechnol*, 25:125–131, 2007.

[93] B B Mandelbrot. *Fractals, Form, Chance, and Dimension.* Freeman, San Francisco, 1977.

[94] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[95] Lynn Margulis. On the origin of mitosing cells. *J Theor Bio.*, 14(3):255–274, 1967.

[96] Lynn Margulis. *Origin of Eukaryotic Cells.* Yale University Press, 1970.

[97] L. Martens, H. Hermjakob, P. Jones, and M. Adamski. Pride: the proteomics identifications database. *Proteomics*, 5:3537–3545, 2005.

[98] L. Martens, M. Muller, C. Stephan, and M. Hamacher. A comparison of the hupo brain proteome project pilot with other proteomics studies. *Proteomics*, 6:5076–5086, 2006.

[99] Geoffrey J. McLachlan and Thiriyambakam Krishnan. *EM Algorithm and Extensions.* Wiley, 1997.

[100] Q. Mei and C. Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *KDD '05.*

[101] Qiaozhu Mei and ChengXiang Zhai. A mixture model for contextual text mining. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 649–655, New York, NY, USA, 2006. ACM Press.

[102] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Duplicate detection in click streams. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 12–21, New York, NY, USA, 2005. ACM Press.

[103] K Nakai and P Horton. Psort: a program for detecting sorting signals in proteins and predicting their subcellular localization. *Trends Biochem Sci*, 24(1):34–36, Jan 1999.

[104] O. Nasraoui and R. Krishnapuram. The unsupervised niche clustering algorithm: extension tomultivariate clusters and application to color image segmentation. *IFSA World Congress and 20th NAFIPS International Conference,*, 3, 2001.

[105] Radford M. Neal and Geoffrey E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. MIT Press, 1999.

[106] A. I. Nesvizhskii and R. Aebersold. Interpretation of shotgun proteomic data: The protein inference problem. *Mol Cell Proteomics*, 4:1419–1440, 2005.

[107] G. Niklas Noren, Roland Orre, and Andrew Bate. A hit-miss model for duplicate detection in the who drug safety database. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 459–468, New York, NY, USA, 2005. ACM Press.

[108] Cédric Notredame, Desmond G Higgins, and Jaap Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205 – 217, 2000.

[109] G. S. Omenn, D. J. States, M. Adamski, and T. W. Blackwell. Overview of the hupo plasma proteome project: results from the pilot phase with 35 collaborating laboratories and multiple analytical groups, generating a core dataset of 3020 proteins and a publicly-available database. *Proteomics*, 5:3226–3245, 2005.

[110] Bernd-Uwe Pagel, Flip Korn, and Christos Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *Proc. ICDE*, 2000.

[111] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *Proc. ICDE*, 2003.

[112] N Pfanner and A Geissler. Versatility of the mitochondrial protein import machinery. *Nat Rev Mol Cell Biol*, 2(5):339–349, May 2001.

[113] Alexandrin Popescul, Lyle H. Ungar, David M. Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 437–444, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[114] J.T. Prince, M. W. Carlson, R. Wang, P. Lu, and E. M. Marcotte. The need for a public proteomics repository. *Nat Biotechnol*, 22:471–472, 2004.

[115] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proc. SIGMOD*. ACM Press, 2002.

[116] K. A. Reidegeld, M. Muller, C. Stephan, and M. Bluggel. The power of cooperative investigation: summary and comparison of the hupo brain proteome project pilot study results. *Proteomics*, 6:4997–5014, 2006.

[117] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[118] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1997.

[119] G Schneider, S Sjöling, E Wallin, P Wrede, E Glaser, and G von Heijne. Feature-extraction from endopeptidase cleavage sites in mitochondrial targeting peptides. *Proteins: Structure, Function, and Genetics*, 30(1):49–60, 1998.

[120] P. Schnell. A method to find point-groups. *Biometrika*, 6:47–48, 1964.

[121] D. W. Scott. Average shifted histograms: Effective nonparametric density estimators in several dimensions. *Ann. Statist.*, 13:1024–1040, 1985.

[122] D.W. Scott. *Multivariate Density Estimation*. Wiley, 1992.

[123] R. Sibson. Slink: an optimally efficient algorithm for the single-linkage cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[124] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[125] I Small, N Peeters, F Legeai, and C. Lurin. Predotar: A tool for rapidly screening proteomes for n-terminal targeting sequences. *Proteomics*, 4(6):1581–1590, Jun 2004.

[126] A. Smellie. Accelerated k-means clustering in metric spaces. *Journal of Chemical Information and Computer Sciences*, 44:1929–1935, 2004.

[127] J C Sprott. *Chaos and Time-Series Analysis*. Oxford University Press, 2003.

[128] Christine Staiger, Alexander Hinneburg, and Ralf Bernd Klösgen. Diversity in degrees of freedom of mitochondrial transit peptides. *Oxford Journal of Molecular Biology and Evolution*, 26:1773–1780, 2009.

[129] C. Steinbeck, S. Krause, and S. Kuhn. Nmrshiftdb-constructing a free chemical information system with open-source components. *J. chem. inf. & comp. sci.*, 43:1733 –1739, 2003.

[130] D. Steinley. K-means clustering: A half-century synthesis. *British Journal of Mathematical & Statistical Psychology*, 59:1–34, 2006.

[131] Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi, and Thomas Griffiths. Probabilistic author-topic models for information discovery. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315, New York, NY, USA, 2004. ACM Press.

[132] J D Thompson, D G Higgins, and T J Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22(22):4673–4680, Nov 1994.

[133] Naonori Ueda and Kazumi Saito. Single-shot detection of multiple categories of text using parametric mixture models. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 626–631, New York, NY, USA, 2002. ACM Press.

[134] G von Heijne, J Stepphuhn, and R G Herrmann. Domain structure of mitochondrial and chloroplast targeting peptides. *Eur. J. Biochem*, 180:535–545, 1989.

[135] Melanie Weis and Felix Naumann. Detecting duplicate objects in xml documents. In *IQIS '04: Proceedings of the 2004 international workshop on Information quality in information systems*, pages 10–19, New York, NY, USA, 2004. ACM Press.

[136] M. Welling, M. Rosen-Zvi, and G. Hinton. Exponential family harmoniums with an application to information retrieval. *Advances in Neural Information Processing Systems*, 17:1481–1488, 2005.

[137] Karina Wolfram, Andrea Porzel, and Alexander Hinneburg. Similarity search for multi-dimensional nmr-spectra of natural products. In *10th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD*, pages 650–658, 2006.

[138] RR Yager and DP Filev. Approximate clustering via the mountain method. *IEEE Transactions on Systems, Man and Cybernetics*, 24(8):1279–1284, 1994.

[139] Hui Yang and Jamie Callan. Near-duplicate detection by instance-level constrained clustering. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 421–428, New York, NY, USA, 2006. ACM Press.

[140] ChengXiang Zhai, Atulya Velivelli, and Bei Yu. A cross-collection mixture model for comparative text mining. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 743–748, New York, NY, USA, 2004. ACM Press.

[141] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Fast density estimation using cf-kernel for very large databases. In *Proceedings KDD'99*, pages 312–316. ACM, 1999.

6 Appendix

Probabilistic models use standard probability distributions as building blocks. Text modeling mainly deals with discrete distributions, which we describe here according to [15]. A probability distribution assigns a non-negative number between zero and one to every value x that a discrete random variable X may take. This is written as $p(X = x)$. To avoid cumbersome notation we write $p(X)$ to denote the distribution and $p(x)$ when the distribution is evaluated at x. Every discrete probability distribution obeys the following constraints $p(X) \geq 0$ and $\sum_X p(X) = 1$. The sum goes over all possible values X may take. The two fundamental rules of probability theory are the

sum rule $p(X) = \sum_Y p(X, Y)$

product rule $p(X, Y) = p(X|Y)p(Y)$

Here $p(X, Y)$ denote the joint distribution of X and Y. The quantity $p(X|Y)$ is the conditional distribution of X given Y and $p(X)$ is the marginal distribution of $p(X, Y)$. In case, two random variables are independent we have

$$p(X, Y) = p(X)p(Y) \tag{6.1}$$

The expectation of some function $f(X)$ under a probability distribution $p(X)$ is a weighted average of the function values

$$\mathbb{E}[f] = \sum_X p(X)f(X) \tag{6.2}$$

The expectation of a function $f(X, Y)$ with respect to a distribution of X

$$\mathbb{E}_X[f(X, Y)] = \sum_X p(X)f(X, Y) \tag{6.3}$$

is still a function with respect to Y. The expectation of a sum of arbitrary random variables decomposed into

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] \tag{6.4}$$

From the product rule together with the symmetry $p(X,Y) = p(Y,X)$ the Bayesian rule of probabilities is obtained:

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \tag{6.5}$$

$$= \frac{p(X|Y)p(Y)}{\sum_Y p(X|Y)p(Y)} \tag{6.6}$$