Computational Methods for Small Molecule Identification

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik

der Friedrich-Schiller-Universität Jena

von Dipl.-Bioinf. Kai Dührkop

geboren am 21. Okt. 1988 **in** Rudolstadt, Deutschland

Gutachter:

1. **Prof. Dr. Sebastian Böcker, Friedrich-Schiller-Universität Jena**
2. **Prof. Dr. Oliver Kohlbacher, Eberhard-Karls-Universität Tübingen**
3. **Prof. Dr. Juho Rousu, Aalto University Espoo/Helsinki in Finnland**

Tag der öffentlichen Verteidigung: 20. September, 2018

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706

Abstract

Identification of small molecules remains a central question in analytical chemistry, in particular for natural product research, metabolomics, environmental research, and biomarker discovery. Mass spectrometry is the predominant technique for high-throughput analysis of small molecules. But it reveals only information about the mass of molecules and, by using tandem mass spectrometry, about the mass of molecular fragments. Automated interpretation of mass spectra is often limited to searching in spectral libraries, such that we can only dereplicate molecules for which we have already recorded reference mass spectra.

In this thesis we present methods for answering two central questions: What is the molecular formula of the measured ion and what is its molecular structure? SIRIUS is a combinatorial optimization method for annotating a spectrum and identifying the ion's molecular formula by computing hypothetical fragmentation trees. We present a new scoring for computing fragmentation trees, transforming the combinatorial optimization into a maximum a posteriori estimator. This allows us to learn parameters and hyperparameters of the scoring directly from data. We demonstrate that the statistical model, which was fitted on a small dataset, generalises well across many different datasets and mass spectrometry instruments.

In addition to tandem mass spectra, isotope pattern can be used for identifying the molecular formula of the precursor ion. We present a novel scoring for comparing isotope patterns based on maximum likelihood. We describe how to integrate the isotope pattern analysis into the fragmentation tree optimisation problem to analyse data were fragment peaks and isotope peaks occur within the same spectrum. We demonstrate that the new scorings significantly improves on the task of molecular formula assignment. We evaluate SIRIUS on several datasets and show that it outperforms all other methods for molecular formula annotation by a large margin.

We also present CSI:FingerID, a method for predicting a molecular fingerprint from a tandem mass spectrum using kernel support vector machines. The predicted fingerprint can be searched in a structure database to identify the molecular structure. CSI:FingerID is based on FingerID, that uses probability product kernels on mass spectra for this task. We describe several novel kernels for comparing fragmentation trees instead of spectra. These kernels are combined using multiple kernel learning. We present a new scoring based on posterior probabilities and extend the method to use additional molecular fingerprints. We demonstrate on several datasets that CSI:FingerID identifies more molecules than its predecessor FingerID and outperforms all other methods for this task. We analyse how each of the methodological improvements of CSI:FingerID contributes to its identification performance and make suggestions for future improvements of the method.

Both methods, SIRIUS and CSI:FingerID, are available as commandline tool and as user interface. The molecular fingerprint prediction is implemented as web service and receives over one million requests per month.

Zusammenfassung

Die Identifizierung kleiner Moleküle ist eine zentrale Fragestellung der analytischen Chemie, insbesondere in der Naturwirkstoffforschung, der Metabolomik, der Ökologie und Umweltforschung sowie in der Entwicklung neuer Diagnoseverfahren mittels Biomarker. Massenspektrometrie ist die vorherrschende Technik für Hochdurchsatzanalysen kleiner Moleküle. Aber sie liefert nur Informationen über die Masse der gemessenen Moleküle und, mittels Tandem-Massenspektrometrie, über die Massen der gemessenen Fragmente. Die automatisierte Auswertung von Massenspektren beschränkt sich oft auf die Suche in Spektrendatenbanken, so dass nur Moleküle derepliziert werden können, die bereits in einer solchen Datenbank gemessen wurden.

In dieser Dissertation präsentieren wir zwei Methoden zur Beantwortung zweier zentraler Fragen: Was ist die Molekülformel eines gemessenen Ions? Und was ist seine Molekülstruktur? SIRIUS ist eine Methode der kombinatorischen Optimierung für die Annotation von Massenspektren und der Identifikation der Molekülformel. Dazu berechnet sie hypothetische Fragmentierungsbäume. Wir stellen ein neues Scoring Modell für die Berechnung von Fragmentierungsbäumen vor, welches die kombinatorische Optimierung als einen Maximum-a-posteriori-Schätzer auffasst. Dieses Modell ermöglicht es uns, Parameter und Hyperparameter des Scorings direkt aus den Daten abzuschätzen. Wir zeigen, dass dieses statistische Modell, dessen (Hyper)Parameter auf einem kleinen Datensatz geschätzt wurden, allgemeingültig für viele Datensätze und sogar für verschiedene Massenspektrometriegeräte ist.

Neben Tandem-Massenspektren lassen sich auch Isotopenmuster zur Molekülformelidentifizierung des Ions verwenden. Wir stellen ein neuartiges Scoring für den Vergleich von Isotopenmustern vor, welches auf Maximum Likelihood basiert. Wir beschreiben, wie die Isotopenmusteranalyse in das Optimierungsproblem für Fragmentierungsbäume integriert werden kann, so dass sich auch Daten analysieren lassen, in denen Fragmente und Isotopenmuster im selben Massenspektrum gemessen werden. Wir zeigen, dass das neue Scoring die korrekte Zuweisung der Molekülformeln signifikant verbessert. Wir evaluieren SIRIUS auf einer Vielzahl von Datensätzen und zeigen, dass die Methode deutlich besser funktioniert als alle anderen Methoden für die Identifikation von Molekülformeln.

Wir stellen außerdem CSI:FingerID vor, eine Methode, die Kernel Support Vector Maschinen zur Vorhersage von molekularen Fingerabdrücken aus Tandem-Massenspektren nutzt. Vorhergesagte molekulare Fingerabdrücke können in Strukturdatenbanken gesucht werden, um die genaue Molekülstruktur aufzuklären. CSI:FingerID basiert auf FingerID, welches Wahrscheinlichkeitsprodukt-Kernels für diese Aufgabe benutzt. Wir beschreiben etliche neue Kernels, zum Vergleich von Fragmentierungsbäumen anstelle von Massenspektren. Diese Kernels werden mittels Multiple Kernel Learning zu einem Kernel kombiniert. Wir stellen ein neues Scoring vor, welches auf A-posteriori-Wahrscheinlichkeiten basiert. Außerdem erweitern wir die Methode, so dass sie zusätzliche molekulare Fingerabdrücke verwendet. Wir zeigen auf verschiedenen Testdatensätzen, dass CSI:FingerID mehr Molekülstrukturen identifizieren kann als der Vorgänger FingerID, und damit auch alle anderen Methoden für diese Anwendung übertrifft. Wir werten aus, wie die verschiedenen

methodischen Erweiterung zur Identifikationsrate von CSI:FingerID beitragen und machen Vorschläge für künftige Verbesserungen der Methode.

Beide Methoden, SIRIUS und CSI:FingerID, sind als Kommandozeilenprogramm und als Benutzeroberfläche verfügbar. Die Vorhersage molekularer Fingerabdrücke ist als Webservice implementiert, der über eine Millionen Anfragen pro Monat erhält.

Acknowledgements

I would like to thank all those people who supported me in writing this thesis. First and foremost, I thank my supervisor Sebastian Böcker who has been extremely supportive with lots of stimulating discussions, and, as he would call it, *crazy ideas*. He has given me the freedom and confidence to go my own way, but helped me, whenever I lost focus.

I also want to thank my co-workers and former co-workers; in particular Marcus Ludwig and Markus Fleischauer, whose office is always open when I need a coffee. Many great ideas were born during these coffee breaks! Franziska Hufsky, Kerstin Scheubert and Tim White deserve credit for assisting me in writing papers. Markus helped me whenever I had problems with the cluster engine or the build script. I also want to thank Kathrin Schowtka for the bureaucracy and administrative work as well as for reminding me to take my holidays.

Many people helped me proofreading this thesis. I want to thank Martin Hoffmann, Thomas Köhler, and Marcus for their support and Eric Bach for his valuable comments about the machine learning chapter. In particular, I want to thank my friend Christoph Wenisch; he read the entire manuscript and helped me with spelling, grammar, legibility, and keeping a consistent appearance for figures, tables and mathematical formulas.

For someone with a bioinformatics background, it is not always easy to understand biochemistry and all the details of mass spectrometry. I thank Louis-Felix Nothias, Emma Schymanski, Steffen Neumann, and Aleš Svatoš for answering all my questions about mass spectrometry. The same is true for machine learning, a topic that was completely new for me. I thank Juho Rousu and his team for answering my questions about this topic, for a lot of fruitful discussions, and for the warm welcome when we visited them in Helsinki.

"Standing on the shoulders of giants" is a well known saying in the scientific community. My "giants" are Florian Rasche, who developed the concept of fragmentation trees, and Markus Heinonen, who developed a kernel support vector machine for predicting molecular fingerprints from tandem mass spectra. Their research is the foundation of this thesis. I am grateful to all my co-authors, in particular Marvin Meusel, Huibin Shen, and Tim White.

I thank the MassBank and GNPS community, Steve Stein, and Frank Kuhlmann and Agilent Technologies, Inc. for providing data. I also want to thank our "power-users", who make me feel that my scientific work also has practical relevance and who provide us with bug reports whenever we were too careless with testing. These are, amongst other, Louis-Felix Nothias and the team around Pieter Dorrestein, as well as Lee Ferguson.

I gratefully acknowledge the funding received through the Friedrich Schiller University of Jena and the Deutsche Forschungsgemeinschaft (DFG), within the projects " Identifying the Unknowns" and "Identifizierung unbekannter Substanzen: Fragmentierungsbäume und Molekulare Fingerabdrücke".

Finally, I would like to thank my mum and dad, my brother and my little sister for their infinite support throughout everything. I am sorry that I did not visit you so often while writing my thesis.

Contents

1 Introduction

When the human genome project announced the successful sequencing of the human genome in 2003, the expectations were high for this will lead to a revolution in life science and help us understanding and treating various diseases. But what we learned from genomics instead is that our genome is just a static view on a highly dynamic and complex biological system. For example, humans can approximately build 10^{12} different antibodies, many magnitudes more than the total number of genes in our genome, which is estimated to be around 20 000 to 25 000. One reason for this diversity is gene regulation: there are various mechanisms that regulate which genes should be expressed (e.g. transcription factor or DNA methylation) and which modify the transcript, for example RNA splicing. Transcriptomics tries to overcome this issue by directly measuring the RNA transcripts in cells. This gives us an insight into the active state of a cell: which genes are expressed and how the transcripts are modified. But regulation does not stop at RNA level: after protein synthesis, proteins can form complex multimeres. For example, antibodies are constructed from two different peptide chains and undergo additional post-translational modifications like glycosylation. These modifications cannot be seen at the RNA level, so we have to measure the cellular proteins. Mass spectrometry is the tool of choice for proteomic experiments. Proteins are either interacting with other proteins, or catalysing biochemical reactions. Metabolites are sources, intermediates, and products of such biochemical reactions. Thus, they are the last step in the information flow from genotype to phenotype. Analysing the metabolome gives a direct read-out of the cellular state. In contrast to proteins, for most metabolites it is not possible to extract information about the metabolites directly from the genome.

Because of their small size, metabolites can often pass barriers like the skin or even the cell membrane. We receive metabolites from our environment, through cosmetic products, pharmaceutics, food, and the microbiome in and on our body. These metabolites are transformed into plenty of new metabolites: this happens in the liver, but also within our cells and by the microbiome in our gut. It is estimated that humans are exposed to 2–3 million different chemicals during their life [93] and it is estimated that 80–85 % of human diseases are linked to hazardous molecules [213].

Environmental science screens for pesticides and other metabolites produced by humans and released into the waste water. Such screenings are also performed in food safety to detect hazardous and illegal substances like pesticides or veterinary drugs [67]. Targeted approaches aim at identifying specific contaminants, but are ineffective when unknown hazardous compounds or metabolised variants of known drugs are to be monitored. In contrast, untargeted approaches that aim to identify all metabolites in a sample, aroused increasing interest in the last years [88].

Untargeted screening is also used for biomarker discovery: many bacteria rely on small secondary metabolites to exert and regulate their virulence [48]. Furthermore, many metabolites are differentially exposed in cancer cells [201] or in patients with certain diseases [1].

Exposure of metabolites from the environment is not only connected to diseases. A review by Aksenov *et al.* [2] gives some examples: the pink colour of flamingo feathers stems from the metabolite canthaxantin, which they absorb through their diet of shrimps and algae. Similarly, the highly toxic dart frog is not producing the poison itself, but accumulates it by eating ants and other arthropods. Metabolites are used for communication, defence, and adaption to environmental stress. Although metabolites play such a crucial role, only a tiny fraction of metabolites are known and even less can be annotated in mass spectrometry experiments today [44, 213].

In 2014, the world health organisation WHO warned about the begin of the "post-antibiotic era" [242]: common bacterial infections, again, become a threat, because they developed resistance against antibiotics. They identified two main problems that are responsible for this crisis: the widespread misuse of antibiotics, in humans but also in animal breeding, and the lack of any discovery of new antibiotic classes since 1970 [8, 221]. A promising approach is to search for new antibiotic classes and other natural products in bacteria or other organisms living in ecological niches other than soil, like in marine environments [8, 92]. While full structural elucidation of such natural products can only be performed using nuclear magnetic resonance instruments, mass spectrometry can help to screen samples for interesting molecules while filtering out molecules that are structural similar to already known classes of natural products [160]. Besides antibiotics, more than half of all small-molecule drugs are either natural products or natural product analogues [186]. It is estimated that there are between 4 000 to 20 000 metabolites present in each eukaryotic organism, with maybe hundred of millions of different metabolites in total [2, 64]. Thus, nature provides an incredible large amount of possible drugs and pharmaceutics from which only a tiny fraction is already discovered. Natural product researchers screen samples from matter found in nature for bioactive metabolites and potential drugs. Due to its high sensitivity and its high-throughput screening capability, mass spectrometry is the predominant technique for such an untargeted analysis.

Although mass spectrometry is a quite old and established method, which was already used to screen for human metabolic disorders such as phenylketonuria in the 1960s and 1970s, it was for a long time a niche field in the scientific community that could not keep pace with the advances and increasing interests in other fields like DNA sequencing [2]. Gas chromatography was for long the primary technique for analysing metabolites, but it can only be applied on very small and volatile compounds. Liquid chromatography and soft ionisation techniques developed in the 1980s made peptide sequencing possible. The field of proteomics has advanced since then and many software solutions for peptide analysis were developed. In metabolomics, however, most researchers are still doing quantitative analysis and biomarker discovery, but without identifying the molecules they measure. Computational metabolomics is still a young field. But this seems to change, with many interesting new approaches being developed in the last years in the field of computational metabolomics [19, 91]. One of these approaches is the computation of fragmentation trees, developed by Florian Rasche and Sebastian Böcker [164]. Fragmentation trees model the behaviour of a molecule in gas-phase chemistry in a strictly combinatorial way. They annotate peaks in a spectrum by molecular formulas and make assumptions about the fragmentation reactions that lead to these fragment ions.

Contribution of this Work

I wrote my diploma thesis about speeding up the computation of fragmentation tree alignments using algorithm engineering techniques like bottom-up dynamic programming. When I started as PhD student, I continued working on tree alignments and, in parallel, on improving the computation of the fragmentation trees itself. I wrote the software SIRIUS for computing fragmentation trees newly from scratch, improving speed and memory usage.

Although I spent much time in programming SIRIUS, I want to focus this thesis not on the technical aspects but on the methodological advances. Therefore, I will mention now but not describe in detail some engineering tricks we developed to speed up SIRIUS: We extended the round robin algorithm for decomposing masses such that it considers mass errors within its computation and optimised its parameters; this leads to a four-fold reduction in running time and reduces memory consumption by up to 94 % [55]. We developed new heuristics for computing fragmentation trees. Although this heuristics sometimes cannot find the exact solution, using them to find lowerbounds speeds up fragmentation tree computation by ten times [59]. I also implemented reduction techniques developed by Tim White and Sebastian Böcker to further speed up fragmentation tree computation [235].

With the availability of thousands of public reference mass spectra, Sebastian Böcker and me reformulated the underlying optimisation problem for computing fragmentation trees as maximum a posteriori probability estimate and learned the parameters of the scoring from data. With this enhanced version of SIRIUS, I attended the "Critical Assessment of Small Molecule Identification" (CASMI), a blind contest, in 2012 [56, 190] and 2013 [57]. Around this time we joined forces with the team of Juho Rousu who pioneered the development of machine learning methods for mass spectrometry. They developed FingerID, a machine learning method for searching mass spectra in molecular structure databases [79]. Together, we developed kernel methods to integrate fragmentation trees into their learning framework [196]. I then reimplemented the FingerID software in Java, again with significant speedups, and developed additional kernels. We also added new fingerprints and developed new scorings. The resulting method was then called CSI:FingerID. We performed the, to the best of my knowledge, most comprehensive evaluation of molecular structure identification methods from tandem mass spectrometry with six different methods being evaluated on three different datasets [58]. I attended again the CASMI contest in 2016 and 2017, this time with CSI:FingerID. In CASMI 2016, CSI:FingerID was the second best method, just beaten by its input output kernel regression (IOKR) variant, developed by Céline Brouard in Juho Rousu group which builds on top of our kernel framework [191]. For CASMI 2017, I implemented a new functionality in SIRIUS for integrating isotope peaks into the tandem mass spectra. With this enhancement, CSI:FingerID could even beat IOKR and was the best method in all contest categories which I attended.

Together with my co-workers, in particular Markus Fleischauer and Marvin Meusel, we developed a user interface for SIRIUS and a server application that runs CSI:FingerID as an online service. CSI:FingerID has been accessed over four million times with more than 200 000 requests on some days. SIRIUS and CSI:FingerID are now integrated into the Global Natural Products Social Molecular Networking (GNPS) [227], as well as into the mass spectrometry frameworks OpenMS [179] and MZmine [157].

Finally, I worked on predicting compound categories from mass spectrometry data. Although this project began some years ago, using the annotations from the Chemical Entities of Biological Interest (ChEBI) database, training such predictors on large scale was only possible with the release of ClassyFire, a web service for structure classification [51] developed by Yannick Djoumbou in David Wishart's lab. Sebastian Böcker and me developed CANOPUS, a tool for predicting these compound categories from ClassyFire, not from structures but directly from mass spectral data.

I also work together with Sebastian Böcker, Kerstin Scheubert, and the group of Rolf Müller on dereplicating natural products using mass spectrometry.

Both topics, compound classification and natural product dereplication, are still work in progress and, therefore, not covered in this thesis.

I presented SIRIUS at the annual conference of the American Society of Mass Spectrometry (ASMS) 2013 in Minneapolis, at the International Conference on Intelligent Systems for Molecular Biology (ISMB) 2013 in Berlin, and at the conference on Research in Computational Molecular Biology (RECOMB) 2015 in Warsaw [54]. I presented CSI:FingerID at the ASMS 2014 in Boston and at the Metabolomics 2016. In 2017, I presented CANOPUS at the Metabolomics in Brisbane. In three of these conferences (ASMS 2014, RECOMB 2015, Metabolomics 2017) I gave an oral presentation. I also participate in the Dagstuhl seminars about computational metabolomics in 2015 [29] and 2017 [3], the Shonan meeting about computational metabolomics in 2017, and the workshop "Current challenges in Eco-metabolomics" at the Deutsches Zentrum für integrative Biodiversitätsforschung (iDiv) in 2017 [154].

My work was founded by the Friedrich-Schiller-Universität Jena and the Deutsche Forschungsgemeinschaft.

In the following, I want to present the two methods I developed and improved during the last years: SIRIUS and CSI:FingerID. As mentioned, I do not want to write about technical details of the implementation or about software engineering techniques. Chapter 2 will give the reader some background about molecular biology and mass spectrometry. In Chapter 3 I will cover the informatics aspects: the background in graph theory, statistics, and machine learning. In Chapter 4 I will give an overview of the methodological development in computational mass spectrometry over the last years. In particular, I will introduce the two methods SIRIUS and FingerID, which are the foundation of this thesis. Afterwards, I will present the maximum a posteriori probability estimate method for SIRIUS in Chapter 6. I will describe its scoring as well as how we estimate the parameters from data. Next, I will evaluate the new scoring on several different datasets. In Chapter 7 I will describe CSI:FingerID and several improvements we have developed: the new kernels, the new fingerprints, and the maximum likelihood scoring. I will evaluate the quality of the kernels and the multiple kernel learning, and then evaluate CSI:FingerID on different datasets against six different state-of-the-art methods.

For the remainder of this thesis, I will use "we" as the first person pronoun, as it is common in scientific literature.

2 Backgrounds in Biochemistry and Mass Spectrometry

2.1 Organic Chemistry

The atom is the basic building block of matter. It consist of a nucleus of protons and neutrons and is enclosed by a cloud of electrons. The number of protons determine the element of an atom. For example, carbon has six protons in its nucleus; hydrogen has one. Organic matter is composed mostly of the five elements carbon (C), hydrogen (H), nitrogen (N), oxygen (O), phosphor (P), and sulphur (S). However, some biomolecules also contain halogens like chlorine (Cl), bromine (Br), iodine (I), fluorine (F), or even metal ions. The number of neutrons determine the isotope of an atom. The sum of protons and neutrons, also called the *atomic mass number*, is written on the upper left of the element symbol. Most carbon atoms contain six neutrons and, therefore, are written as ^{12}C. Another isotope of carbon with seven neutrons is ^{13}C. The number of protons and neutrons (and, to a lesser extend, the number of electrons) determine the mass of the atom, which is denoted in unified atomic mass unit (u) or (historically) in Dalton (Da). 1 u (or 1 Da) is defined as $\frac{1}{12}$ of the mass of a ^{12}C atom and is approximately $1.660\,539\,04 \times 10^{-27}$ kg. The atomic mass and the mass number differ for all atoms except for ^{12}C. Hydrogen ^{1}H, for example, has a mass number of one, but an atomic mass of 1.007 825 032 Da [226]. This difference is called *mass defect*. It is caused by the binding energy in the nucleus and is the reason why mass spectrometry is able to infer the atom composition of molecules from their mass. Each element (and isotope) has an unique mass (and mass defect). Isotopes of the same element usually have the same chemical properties, except for the mass. In nature, we can observe different isotopes for most of the elements. Some isotopes are radioactive and occur in extremely low frequency. An example is ^{14}C. The distribution of isotopes differs slightly between different locations on earth. Therefore, isotope ratios cannot be listed as precisely as atomic or isotopic masses. For carbon there are two stable isotopes: ^{12}C with a relative abundance of 98.93 % and ^{13}C with 1.07 %. Throughout this thesis, we will use the term *monoisotopic mass* for the mass of a molecule where each atom is the stable (not radioactive) isotope with the lowest nominal mass. This follows the definition by Böcker *et al.* [28] and Meusel *et al.* [133] but differs from that of IUPAC, which uses the most abundant isotope instead of the lightest one. For computational mass spectrometry and, in particular, the simulation and fitting of isotope patterns of small molecules, the definition given here is more convenient. See Table 2.1 for the masses and abundances of the common isotopes we can detect in organic molecules. In our work we will always use masses from Wang *et al.* [226] and isotopic abundances from Guha *et al.* [74].

The number of electrons of an atom determines its charge: if the electron number equals the proton number, it is electrical neutral, otherwise it is an *ion*. If an atom has more electrons than protons, it is negatively charged and called an *anion*. If the number of electrons is smaller than the number of protons, it is positively charged and called an *cation*. Ions can be accelerated in electric fields. This is the basic functioning of mass

Table 2.1: Table of the basic elements of organic matter (carbon, hydrogen, oxygen, nitrogen, phosphor, and sulphur) as well as the halogens chlorine, bromine, fluorine, and iodine. The table lists the elements and their natural abundant isotopes with mass, mass defect, and relative abundance. Values are taken from [74, 226].

Element	Symbol	Isotope	Mass (Da)	Mass defect (Da)	Abundance (%)
carbon	C	^{12}C	12	0	98.93
		^{13}C	13.00335484	0.00335484	1.07
hydrogen	H	^{1}H	1.007825032	0.007825032	99.9885
		^{2}H	2.014101778	0.014101778	0.0115
nitrogen	N	^{14}N	14.003074	0.003074	99.632
		^{15}N	15.0001089	0.0001089	0.368
oxygen	O	^{16}O	15.99491462	−0.00508538	99.757
		^{17}O	16.9991317	−0.00086830	0.038
		^{18}O	17.999161	−0.00083900	0.205
phosphor	P	^{31}P	30.97376163	−0.02623837	100
sulphur	S	^{32}S	31.972071	−0.02792900	94.93
		^{33}S	32.97145876	−0.02854124	0.76
		^{34}S	33.9678669	−0.03213310	4.29
		^{36}S	35.96708076	−0.03291924	0.02
bromine	Br	^{79}Br	78.9183371	−0.0816629	50.69
		^{81}Br	80.9162906	−0.0837094	49.31
chlorine	Cl	^{35}Cl	34.96885268	−0.03114732	75.78
		^{37}Cl	36.96590259	−0.03409741	24.22
fluorine	F	^{19}F	18.99840322	−0.00159678	100
iodine	I	^{127}I	126.904473	−0.095527	100

spectrometry. There are several ways to transform a neutral molecule into an ion. In general we can add (or remove) a proton or add (or remove) an electron. A hydrogen atom consists of one proton and one electron. If we remove the electron, we get a positively charged hydrogen which is also a proton. Therefore, protons are usually written as H^+ while electrons are written as e^-. Electrons are also the source of the strongest chemical bonding between atoms: the *covalent bonds* and *ionic bonds*. Covalent bonds are formed between pairs of electrons in the outer electron shell (called *valence electrons*) of two atoms. If the electronegativity between two atoms is very different, covalent bonds become ionic bonds. For example H_2O is connected with covalent bonds, while NaCl is connected with ionic bonds. The connection of multiple atoms via bonds is called a molecule. A molecule consisting of atoms with different elements is also called a *compound*. However, in this thesis we will use both terms synonymously. Covalent bonds can be distinguished between single bonds (formed by one pair of electrons), double bonds (two pairs), and triple bonds (three pairs). There is also the special case of aromatic bonds: These are ring structures of single and double bonds where the bond type is dynamic: it is not possible to locate the single and double bonds within the ring. Because covalent bonds are always formed by pairs of electrons, the number of valence electrons in molecules is usually even. A molecule with an odd number of valence electrons has an unpaired electron. This is chemical unstable. The molecule will "try" to react with other molecules in its environment to become chemically stable. We call these molecules *radicals*. In general, molecules are chemically stable if all their valence electrons are paired in some covalent bond. This knowledge can be used to estimate the number of double bonds, triple bonds, and rings in a molecule if its atomic composition is known. This equation is called ring double bond equivalent (RDBE) and is defined as

$$\mathrm{RDBE}_{\Sigma} = 1 + \frac{1}{2} \sum_{e \in \Sigma} \big(\#e \cdot (\mathrm{valence}(e) - 2) \big) \tag{2.1}$$

$$\mathrm{RDBE}_{\mathrm{CHNOPS}} = 1 + \tfrac{1}{2} \Big(2\#\mathrm{C} - \#\mathrm{H} + \#\mathrm{N} + \#\mathrm{P} \Big) \tag{2.2}$$

for arbitrary chemical alphabets Σ (eq. (2.1)) and for the specific alphabet CHNOPS (eq. (2.2)). Here #e denotes the number of atoms for element e. However, there exist some exceptions to this rule [107]. Basically, these equations show that for each pair of valence electron we can form a single bond to an atom. If we do not have enough atoms to satisfy all valence pairs, we have to either form double bonds, triple bonds, or rings. Elements with odd number of valence electrons are special, because their valence electrons can only be paired if the number of such odd valence elements is even. This is also known as *nitrogen rule*, as nitrogen is the most common element in organic molecules with odd valence electrons. If a molecule violates this rule, the RDBE becomes a fractional number. This is the case for all radicals. Also hydrogen is special, because it only has a single valence electron. A molecule with five hydrogens and a single carbon is highly unlikely, because carbon typically has a valence of four. The RDBE of such a molecule would be negative. The rule that forbid such atomic compositions is part of the "Senior rules" [193]. For a given set of atoms we can use the RDBE to determine if a neutral molecule with these atoms can exist: we check if the RDBE is positive and even.

We can write a molecule as *molecular formula* by writing down the elements and their abundance in a molecule in alphabetical order (except carbon and hydrogen, which are named first in carbon containing molecules). This is also called Hill notation. An example is $C_6H_{12}O_6$, the molecular formula of glucose. If the molecule is an ion, we write its charge

state on the upper right. A protonated glucose (a glucose with an additional proton) is written as $C_6H_{13}O_6^+$ and a deprotonated glucose (a glucose that lost a proton) is written as $C_6H_{11}O_6^-$. The molecular formula does not tell us anything about the constitution of the molecule, but just about its atomic composition. Fructose, for example, has the same molecular formula as glucose, but a different constitution. We call molecules with same molecular formula but different structure *structural isomers*. *Structural formulas* or *skeleton formulas* are used to draw a molecule with its constitution. Skeleton formulas are specialised for organic molecules: The drawing of carbon atoms is omitted, as almost all biomolecules have a carbon backbone. Also hydrogens and bonds to hydrogens are usually left out, because the number of neighbour hydrogens of an atom can be determined from its free valences. Of course there are always exceptions, where it is necessary to explicitly draw the hydrogens or carbons. Hydrogens that can be determined from the free valences are also called implicit hydrogens.

As covalent bonds are oriented in a three dimensional space, molecules with same molecular formula and same constitution can still differ in their volume. We call the orientation of the bonds the *stereochemistry*. Molecules that differ only in their stereochemistry are called *stereoisomers*. In general, we cannot distinguish stereoisomers using mass spectrometry because the stereochemistry does not change the mass of the molecules. However, other methods that depend on the volume and shape of a molecule are able to distinguish some stereoisomers (for example liquid chromatography, see Section 2.6).

2.2 Representation of Molecules in Cheminformatics

As a computer scientist it is natural to look on molecules as labelled (multi)graphs. Readers who are not familiar with the concept of graphs might first want to read Section 3.1. The atoms of a molecule can be represented as vertices labelled with the element symbol. The covalent bonds can be either represented as weighted edges (with edge weights are 1, 2, or 3) or as multiple edges between the same set of vertices (graphs with this property are called multigraphs). The graph representation omits many details like stereochemistry and quantum chemistry, but it allows us to use efficient graph algorithms and decades of theoretical research in graph theory to deal with cheminformatic problems.

2.2.1 Text-based Data Formats

Text representations are important to store molecules in text based databases or transfer molecules in text format or via text based protocols like HTTP. The MDL connection table format represent a molecule as adjacency matrix with atom coordinates. In SMILES, linear molecules are written as sequence of element symbols and bond symbols. Implicit hydrogens and single bonds can be omitted. For example, the SMILES for ethanol is CCO. Ethylene is written as C=C. Side chains are enclosed in brackets. Rings are denoted by numbers between the ring-closing atoms. See Fig. 2.1 for an example. There are many ways to write down a molecule as SMILES. For example, ethanol can also be written as OCC or C(O)C. Canonical SMILES try to overcome this issue by define a unique way to describe a molecule but failed and produced non unique strings for some structures [233]. Today there is no official standard for canonical SMILES. Instead, the international chemical identifier (InChI) was introduced, using graph isomorphism algorithms to order atoms and bonds in an unique way [80]. It is less intuitive to read than SMILES, but is organised in

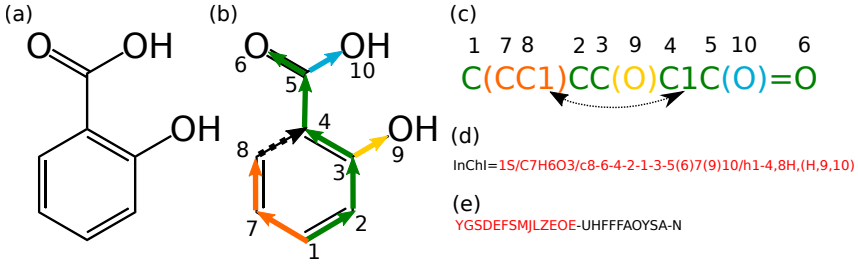


Figure 2.1: Example for a SMILES and InChI representation of the metabolite salicyclic acid. (a) The structural formula of salicyclic acid. (b) For building the SMILES we arbitrary transform the molecule graph into a tree and number the atoms (except hydrogens). The molecule graph contains a cycle that cannot be reflected in the tree structure (dashed arrow). (c) One possible SMILES string that represents the molecule. Each branch in the tree is enclosed in brackets. The atoms with number 4 and 8 are reconnected into a cycle by adding the digit 1 behind the atom symbols. (d) The InChI of salicyclic acid, consisting of the main layer, which itself consists of three sublayers: molecular formula, atom connectivity, and hydrogen atoms. (e) The hashed InChI key of salicyclic acid. The first 14 characters are the hashed main layer (red). The remaining characters are the hash of the empty stereochemistry and empty charge layers.

layers. The first layer is the main layer, which itself consists of a molecular formula layer, an atom connectivity layer, and an hydrogen atom layer. The second layer defines the stereochemistry. There are also layers for charges and isotopes. To check if two molecules are identical, one just have to check if their InChIs are identical. The layer based systems allows to omit certain properties like stereochemistry when performing the identity test. As the InChI contains all information of the molecular graph, it can be arbitrary long. Many database systems prefer fixed length identifiers. The InChI-Key is a hashed Base26 version of the InChI and exactly 27 characters in length. Due to the nature of hashes, two molecules might have different InChIs but the same InChI-Key. However, the probability for such a collision is extremely low [156].

Beside text formats to describe whole molecules there are also special formats for describing molecular substructures (SMARTS) or chemical reactions (SMIRKS). SMARTS is a superset of SMILES and introduce special characters for atom and bond wildcards, conditions on the atom environments, as well as logical operators. For example "C∼[CH2]∼[CD1]" describes three connected carbons with unspecified bonds where the middle carbon atom is connected to exactly two hydrogen and the third carbon atom has exactly one non-hydrogen neighbour atom.

2.2.2 Molecular Fingerprints

Canonical representations allow to test for identity, but they cannot measure the similarity of two molecules. In fact, two molecules can have very different InChIs or SMILES but still a very similar structure. The concept of chemical similarity is important, especially in pharmacology: if two molecules are structural very similar, they might also have similar chemical properties, bind to the same enzymes, and show similar bioactivity. This relationship between the chemical structure and its bioactivity is called structure related activity relationship (SAR). Based on this assumption quantitative structure related activity relationship (QSAR) models aim to predict biochemical activity from physicochemical properties of the molecules. There is no natural way to compare two

molecules. From a chemists perspective, two molecules are similar if they have similar backbones and functional groups. From a computer scientists perspective, two molecules can be considered similar if they share a large common substructure, which leads to the NP–hard maximum induced subgraph isomorphism problem [202], or if there exist an high-scoring alignment [167]. Different applications might require different similarity measures. See [147, 237] for more details about chemical similarity and QSAR.

Molecular fingerprints were developed as a very fast method for comparing molecules: They are binary vectors indicating the presence and absence of *molecular properties*. A molecular property can be a substructure like a functional group but also encode the presence of certain atoms, atom configurations, or bonds. When encoding a molecule as fingerprint, information about the localisation and frequency of its molecular properties are lost. Molecular holograms generalise the concept of fingerprints by counting molecular properties [147]. In this thesis we will sometimes refer to molecular fingerprints as binary vectors or as set of molecular properties, whatever is more appropriate in the context. A common method for comparing molecular fingerprints is the *Tanimoto coefficient*, in computer sciences better known as Jaccard index [236]. It is calculated as

$$\text{Tanimoto}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.3}$$

for two sets A and B of molecular descriptors. Other methods for similarity calculations are the Dice coefficient or the normalised dot product (better known as cosine similarity). The cosine similarity can also be used for molecular holograms. Distance measures like the euclidean distance or the XOR distance (number of different molecular properties) are also in use. See [11] for a comparison of these methods.

Computing molecular fingerprints is a core feature implemented in many cheminformatic toolkits like CDK [205, 238], OpenBabel [150], or RDKit. A large number of different fingerprint types is available. See [15, 37, 52] for reviews about the different fingerprint types and how they perform and complement each other. Most fingerprints are substructure-based: they ship with a list of structural keys (often in SMARTS format) and check for the presence of these substructures in the molecule. The advantage of substructure-based fingerprints is that they can encode for known functional groups; a disadvantage is that they are restricted to a rather small list of manual selected substructures. PubChem [228] and MACCS fingerprints are well known examples of this category. Path-based fingerprints, instead, enumerate over all paths in a molecule. Shortest path fingerprints enumerate over all pairs of atoms and report the shortest path between them. Both fingerprint types report a finite set of paths per molecule, but an infinite set over all possible molecules. They use a hash function to map a path to an index in the fingerprint vector. Circular fingerprints use the Morgan algorithm [137] to enumerate over all atoms and their neighbourhood within a given radius: First, all atoms are labelled (for example by their element symbol). Then in each iteration, each atom is relabelled by concatenating its label with the sorted labels of its neighbouring atoms. Instead of concatenating strings, one can also calculate a hash function in each step. Circular fingerprints can be calculated extremely fast and encode a large number of substructures. ECFP (extended connectivity fingerprint) [174] is an example of a circular fingerprints. There are other circular fingerprints like FCFP or SCFP that just differ in the labelling of the atoms. As circular fingerprints are infinite in length (like path fingerprints), they are usually hashed to fit into a fixed length vector. See Fig. 2.2 for a few examples of

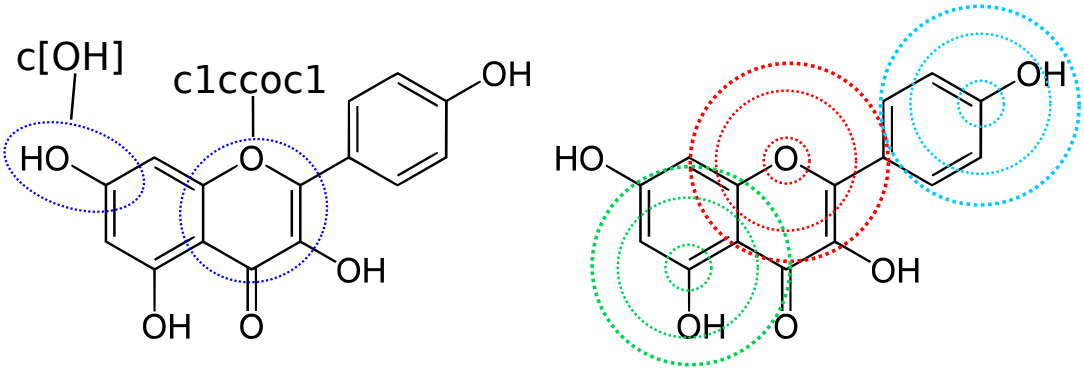


Figure 2.2: This figure shows the metabolite kaempferol. Left: Two substructure based molecular properties are highlighted within blue circles. The "c[OH]" SMARTS describes a phenol functional group. The "c1ccoc1" SMARTS describes a furan substructure. In both cases, the lowercase letter c stands for an carbon within an aromatic ring. Right: Depiction of the circular fingerprints like ECFP. For each atom we store its neighbourhood (depicted as coloured dashed circle) with fixed radius of 0, 1, or 2 as a hash. In this example, we only draw neighbourhoods for three different atoms.

substructure molecular properties and circular molecular properties which are used within our methods.

Fingerprints are used for ligand screening [68] and QSAR models [53]. However, two things should be considered here: First, most molecular fingerprints are optimised for small molecules. As they do not encode for the localisation and frequency of a molecular property, larger molecules can have very similar fingerprints but still look very different. A good example are peptides, which share the same building blocks (amino acids) and, therefore, the same molecular properties. To compare peptides it is crucial to consider the order and frequency in which the molecular properties occur. It is also reported that the Tanimoto coefficient is not suited to molecules with high variance in size [50]. Another problem is that path and circular fingerprints use hashing to keep the length of the fingerprint constant. The consequence is that very different substructures may fall into the same hash bins (an effect which is called *collision*).

2.3 Metabolites and Metabolomics

The term *metabolites* embraces all small molecules (usually with a mass below 1000 Dalton) in living organisms. Metabolites are distinguished from polymeres like peptides, DNA, or glycans which are chains or trees of smaller building blocks. For example, a single amino acid is itself a metabolite, a chain of amino acids (called a peptide) is a polymer. However, there is no clear definition of metabolites. Lipids, for example, are sometimes counted as metabolites and sometimes not. For the methods presented in this thesis the biochemical reasoning is less important: These methods deal with molecules that do not consist solely of simple building blocks, because for such polymeres like peptides we can find more efficient methods. In addition, they are optimised for molecules which are very small.

Biochemists distinguish between primary metabolites and secondary metabolites. *Primary metabolites* are necessary for growth, replication, and survival of cells. Many primary metabolites are well researched and conserved across many species. In contrast, *secondary metabolites* are produced on demand to deal with environmental stress, defend

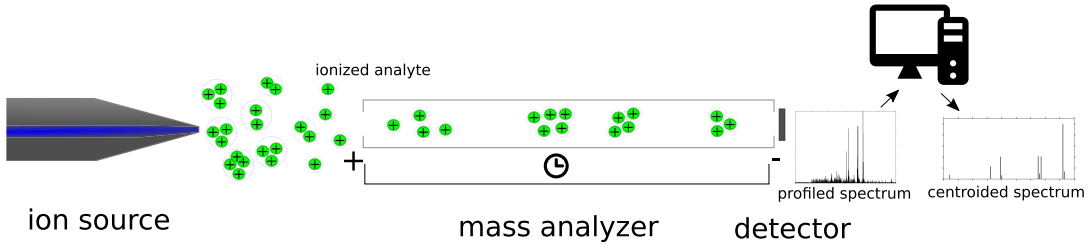


Figure 2.3: Schematic representation of a time-of-flight mass spectrometry instrument with an electrospray ion source.

against herbivores, microbes or viruses, or as signalling molecules. They are often very specific for a species or phylogenetic family and improve the organism's fitness and adaption to an ecological niche [239]. There is a large structural variety in secondary metabolites and most of them are still unknown. The term *natural product* is sometimes used synonymously to metabolites, but more often refers to secondary metabolites.

The research of all metabolites that can be detected in a sample is called *metabolomics*.

2.4 Mass Spectrometry

Mass Spectrometry instruments enable high sensitive and high-throughput analysis of molecules by recording their molecular mass. These instruments can be used to measure large molecules as well as very small molecules. The accuracy of the instruments depends on the size of the measured molecule; thus we use the relative scale *parts per million* (ppm) to specify the accuracy of an instrument. High mass accuracy instruments can measure molecular masses with an accuracy of few ppm or even below one ppm. The *resolution* corresponds to the minimal mass difference that can be distinguished by the instrument. Thus, instruments with low resolution cannot distinguish between ions with very similar masses (called isobaric ions), while high resolution instruments can even discriminate between different isotopologues.

The basic setup of a mass spectrometry instrument consists of an *ion source*, a *mass analyser*, and a *detector*. The ion source is necessary to ionise the analyte. We distinguish between soft ionisation techniques that keep the analyte intact and hard ionisation techniques that fragment the analyte. The reason for ionisation is that ionised molecules can be controlled within electromagnetic fields; this is the basic principle of every mass spectrometry instrument. Neutral molecules without a charge cannot be detected. The mass analyser separates the ions by their mass-to-charge ratio. Finally, the detector records incoming ions with equal mass-to-charge ratio.

We will shortly explain the functionality on an example: the time-of-flight (ToF) mass spectrometry instrument with electrospray ionisation (ESI), see Fig. 2.3. The electrospray ionisation is a soft ionisation technique that presses the sample and a liquid solvent through a thin, highly charged capillary. At the tip of this capillary, the sample and solvent molecules get charged, either by receiving protons (positive ionisation) or by loosing protons (negative ionisations), depending on the polarity of the capillary tip. Small drops of solvent and sample molecules are formed, move away from the tip, and successively break into smaller droplets while the solvent evaporates. Finally, most droplets consist solely of the charged analyte. For small molecules, most of the generated ions are single-charged.

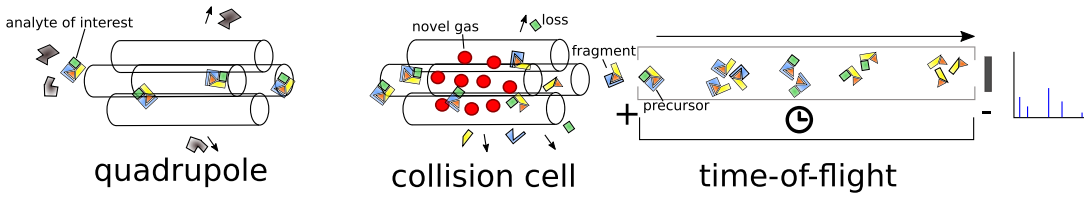


Figure 2.4: Schematic representation of Q-ToF instrument for tandem mass spectrometry.

The charged ions are now accelerated in an electromagnetic field and fly through a long flight tube. At the end of the tube they hit the detector and are recorded. All ions with same charge receive the same kinetic energy, but ions with high mass need more energy to accelerate than ions with low mass. Therefore, the mass of an ion can be deduced by measuring the time an ion needs to reach the detector. The mapping from the physical recorded value (here, the measured time and the used voltage) is obtained by calibration: ions with known masses have to be measured regularly in the instrument to fine-tune this mapping.

The output of a mass spectrometry scan is a spectrum: this is a list of mass-to-charge (m/z) and intensity values recorded by the mass detector. When multiple ions with same mass hit the detector, multiple m/z and intensity values are recorded. Thus, in a first step, a peak picker software transforms this *profiled spectrum* into a discrete list where each pair of m/z and intensity values belongs to a single recorded ion species. This is called a *centroided spectrum*.

Fourier transform instruments like Orbitrap or Fourier Transform Ion Cyclotron Resonance (FTICR) instruments work quite differently. In both instruments, ions are "trapped" either by cycling in a high magnetic field of a super-conducting magnet in FTCIR, or by cycling around an electrode in Orbitrap. The frequency of this cycling can be detected and is inversely proportional to the ions mass-to-charge ratio. Computing the m/z values involves the Fourier transformation, thus the name of these instruments. Fourier transform instruments have the highest resolution and mass accuracy.

Adducts When using electrospray ionisation in positive mode we usually detect *protonated* ions; these are molecules that carry an additional proton. In negative ion mode we most often detect *deprotonated* ions; molecules that lost a single proton. We write $[M+H]^+$ for protonation and $[M-H]^-$ for deprotonation. If the sample contains salt, it might happen that a sodium atom is attached to the molecule instead of a proton; we denote this as $[M+Na]^+$. Analogously, in negative ion mode chlorine atoms can be attached, which we will write as $[M+Cl]^-$. We call these additions *adducts*, as they are not covalently bound to the molecule. There is a large variety of adducts, like ammonium $[M+NH4]^+$ or potassium $[M+K]^+$.

In this thesis we will deliberately ignore adducts and simply assume that all ions are protonated. All our methods can be easily extended to deal with arbitrary adducts, but this only makes the description of the methods tedious. Note that our software implementation, SIRIUS, can naturally deal with arbitrary adducts.

2.5 Tandem Mass Spectrometry

Mass spectrometry is only able to measure the mass-to-charge ratio of molecules. The mass of a molecule depends on the atomic composition: the number of carbons, hydrogens and so on. The connectivity of these atoms, however, has only marginal impact on the mass; the mass difference between two structural isomers is not detectable with mass spectrometry. *Tandem mass spectrometry* (MS^2) was introduced to distinguish between structural isomers or isobaric compounds (compounds with same nominal mass that might not be distinguishable with the instrument's mass accuracy). In MS^2, one mass spectrometry instrument is used to separate ions with a specific mass. These ions are then fragmented. Finally, the masses of the fragments are measured with a second mass spectrometry instrument. The resulting MS^2 mass spectrum contains the masses of the fragments and their intensity (which is related to their frequency). Tandem mass spectrometry is highly reproducible. Thus, as soon as we know the MS^2 spectrum of a molecule, we can use tandem mass spectrometry to detect this molecule in other samples.

Again, we will give a concrete example of such a tandem mass spectrometry setup: a quadrupole time-of-flight instrument (Q-ToF) with collision-induced dissociation (CID), see Fig. 2.4. A quadrupole consists of four circular and parallel rods on which oscillating electric fields are applied. For a given ratio of voltages, only ions with specific mass-to-charge ratio can pass the rod; other ions will have unstable trajectories and collide with the rods or leave the quadrupole. Quadrupoles can be used as mass analyser; in contrast to ToF analysers they have a much lower resolution and mass accuracy. In a Q-ToF setup, the quadrupole is just to isolate ions with a specific mass for the MS^2 analysis. Next, there is another quadrupole, called the collision cell. It can be filled with noble gas (or, because it is cheaper, nitrogen gas). On their trajectory through the collision cell, the ions collide with the noble gas, receive kinetic energy, and fragment. This is called *collision induced dissociation*. The *collision energy* influences how fast the ions move through the cell and how much kinetic energy they receive when colliding with the Nobel gas. A fragmentation can be the cleavage of one or multiple bonds, but also arbitrary complex chemical reactions (called *rearrangements*). If the ion carries only a single charge, it breaks up into a charged and an uncharged fragment. Only the charged fragment can pass the collision cell and moves, due to the electric field, into the third mass spectrometry instrument: the ToF instrument. Here, the charged fragments are recorded in an MS^2 spectrum. For low collision energies, some ions pass the collision cell without fragmenting. These intact ions are called *precursor ions*. The charged fragment ions are also called product ions, but in this thesis we will use the term *fragments* instead. The undetectable uncharged fragments are called *losses*. In *untargeted mass spectrometry*, the instrument records alternately MS and MS^2 scans: In MS mode, the first quadrupole lets all ions pass through and the second quadrupole is operated on low voltage such that the ions move very slowly through the collision cell and do not fragment. Thus, the intact ions are recorded in the ToF instrument and an MS scan is taken. The software then selects an ion with high intensity for fragmentation: the instrument is then switching to MS^2 mode, isolates the selected ions in the first quadrupole, fragments them in the collision cell, and records the MS^2 spectrum.

Note that there are many different systems beside Q-ToF: In ion traps, for example, isolation, fragmentation and recording is done within the same mass spectrometry instrument. For the methods presented here we do not care about the choice of mass spectrometry instrument as long as it provides high mass accuracy. This is the case for time-

of-flight, Orbitrap, and Fourier Transform Ion Cyclotron Resonance (FTICR) instruments. However, the choice of fragmentation technique, the used novel gas, the collision energy, and also the instrument can have a big impact on the fragmentation behaviour and the resulting MS^2 spectrum. Nevertheless, spectra of the same compound recorded on different instrument types are often quite similar [149, 230].

There are several related techniques that do not isolate ions with very similar mass, but instead fragment a wide range of masses at once. These techniques are called MSall, MSe, or SWAFT, depending on the manufacturer.

2.6 Liquid Chromatography

Chromatography is necessary for an untargeted analysis of samples with many ions, for example blood or urine. Here, frequent metabolites (in particular primary metabolites) interfere with the detection of rare metabolites. Furthermore, many ions with same or similar mass make it impossible for tandem mass spectrometry to isolate a single ion species. A chromatography separates analytes by certain chemicophysical properties besides mass; for example by polarity or volume. Gas chromatography and liquid chromatography are the predominant separation techniques in mass spectrometry. New techniques like ion mobility can be used in addition, for better separation.

A chromatography system consists of a column with a stationary phase and an mobile phase. The mobile phase moves together with the analyte through the column. The analytes bind to the stationary phase with different affinity and, thus, separate.

In gas chromatography (GC), the mobile phase is a gas. Due to the high temperature in gas chromatography columns, this technique is unsuitable for thermal unstable molecules like peptides and larger metabolites. In liquid chromatography (LC) the mobile phase is a liquid. Liquid chromatography coupled with mass spectrometry is abbreviated with LC-MS; analogously, we use GC-MS for gas chromatography and mass spectrometry. In this thesis we will focus solely on LC-MS. Whereas all the methods presented here can be extended to gas chromatography, too, this is a highly non-trivial undertaking that introduces a couple of new problems; One has to deal with derivatisations, which are often necessary to make molecules thermal stable, as well as with hard ionisation techniques like electron ionisation which are the predominant ion source for gas chromatography.

3 Combinatorics, Statistics and Machine Learning

The structural annotation and identification process we present in this thesis involves techniques from combinatorics, statistics, and machine learning. This chapter shall give the reader a short introduction in these three fields of computer science. Furthermore, we will explain the notations and formalisms we use throughout this thesis.

Combinatorics is a field of mathematics that deals with counting, enumerating and construction of finite structures. Such finite structures include but are not limited to strings, graphs, trees, or sets of discrete elements. One example is the mass decomposition, where we use combinatorics to enumerate all molecular formulas (which can be represented as multisets of atoms) with a given mass. *Combinatorial optimisation* is an area of combinatorics that consists of finding an optimal structure from a finite set of structures. An example is the maximum colourful subtree problem in Section 4.2.1 where we search for an explanation of a mass spectrum in form of a tree with maximum score. A common algorithm engineering technique in combinatorial optimisation is *dynamic programming*. It solves an optimisation problem by integrating over the solutions of simpler subproblems. Its efficiency arrives from the fact that it does not have to enumerate all structures but, instead, can summarise over parts of the search space.

Combinatorial optimisation often requires some scoring function that determines which structure is optimum. When working with biological data the best scoring is often obtained using a statistical model. In such a model we can encode which structures are more likely to observe; we will call this kind of probability a *prior* and we can estimate it using expert knowledge or by fitting *parameterised probability distributions* to databases of known structures. We can also encode which structures are in agreement to our observed measurements. We call this probability a *likelihood*. Finding the most likely explanation given the measured data usually involves a maximum likelihood analysis (which ignores the prior) or a maximum a posteriori analysis (which uses both, prior and likelihood). When working with complicated structures and multiple variables we have to deal with dependencies between these variables. A simple trick is to claim that all variables are independent. We will use this trick several times in this thesis, as it makes the problem easier to solve, although on biological data this assumption is almost always wrong.

From statistics to machine learning is a smooth transition. For example, Naive Bayes is a machine learning technique that uses the mentioned independence assumption to fit multivariate distributions and obtain a maximum likelihood estimate. Most machine learning techniques like linear regression, support vector machines or deep neural networks are based on the foundation of statistics and statistical learning theory. We will use machine learning to extract information from our measured data in situations where we do not know the intrinsic relationship between the data and the extracted information. This is, for example, the case for structure elucidation from tandem mass spectra. Although there have to be some relationship between the measured peaks and the structure of the measured molecules, gas phase chemistry is complex to such an extend that nobody really

knows how this relationship looks like. Even quantum chemistry calculations are not sufficient to explain the measured data [200]; probably because there are too many hidden parameters in the experimental setup and measurement process. Machine learning allows to fit a function that maps the measured spectra to the molecules by extracting patterns from the measured data that are correlated to the molecular structure.

Next, we will explain each of this field in more detail and introduce the notations and formalisms we will use in the following chapters.

3.1 Graphs and Trees

Graphs are a concept in combinatorics to model objects and their relationships to each other. We will first introduce graphs and the terminology used in this thesis in a formal way. We recommend Diestel [49] for a comprehensive background in graph theory and related concepts.

A *graph* $G = (V, E)$ is a pair of *nodes* and *edges* $E \subseteq V \times V$. An edge connects two nodes from the graph. We note an edge e between the vertices u and v as $e = \{u, v\}$. In *directed graphs*, the edges are ordered pairs of nodes and point from one node to another. We write an directed edge e from node u to v as $e = (u, v)$ or, shorter, as $e = uv$ and call u the start node and v the terminal node of e. We call two edges *adjacent* to each other if one of their nodes appears in both sets (undirected graphs) or the start node of one edge is equal to the terminal node of the other edge (directed graphs). Similarly, two nodes u and v are adjacent if there is an edge $\{u, v\}$ (or uv for directed graphs). The *out-degree* of a node u is the number of adjacent edges where u is the start node. A graph is *connected* if for each two nodes there exists a path containing these nodes. A *path* is a sequence of edges such that no edge appears twice in the path and each consecutive edges are adjacent. A path is a *cycle* if the start node of the first edge equals the terminal node of the last edge. A directed graph without cycles is called a *directed acyclic graph* (DAG). A connected undirected graph without cycles is called a *tree*. An *arborescence* is a DAG with a special node called *root* with the property that for any node there is exactly one path from the root to this node. We can transform any tree into an *arborescence* by choosing a node as root and then make all edges directed such that they point away from the root. In this thesis we will often use the term tree synonymously for arborescence if it is either clear from the context that the tree is directed or if the direction of the edges is not important. For arborescence trees, the *induced subtree* $T[u]$ for any node $u \in V$ is the tree consisting of all nodes and edges which are reachable from u. For two nodes u,v with $v \in T[u]$ we say that u is the *ancestor* of v and v is the *descendant* of u.

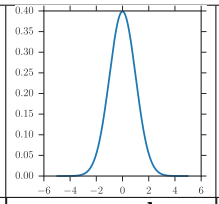
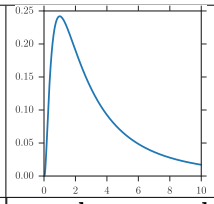
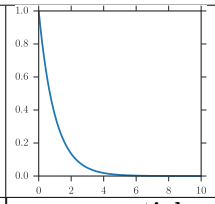
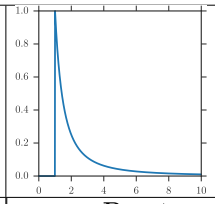
3.2 Bayesian Statistics and Parameterised Distributions

Bayesian statistics centres around Bayes' theorem for conditional probabilities:

$$\mathbb{P}(M|D) = \frac{\mathbb{P}(D|M)\mathbb{P}(M)}{\mathbb{P}(D)} \tag{3.1}$$

Here, $\mathbb{P}(M|D)$ is the *conditional probability* that M is true given D. M is usually a model or a hypothesis. $\mathbb{P}(M)$ is the *prior probability* that reflects our initial knowledge: how likely is our model without doing any observations. D is data derived from measurements and

Table 3.1: The four probability distributions which are repeatedly used in this thesis. From left to right: normal distribution (also called Gaussian distribution), log-normal distribution, exponential distribution, and Pareto distribution. For each of these parametric distributions we describe the parameters, the cumulative distribution function, the probability density function, and the first two moments. We also plot examples for the probability density function using standard parameters: $\mu = 0$, $\sigma^2 = 1$ for normal and log-normal distributions, $\lambda = 1$ for exponential distribution, and $x_{min} = 1$ and $k = 1$ for Pareto distribution.

Probability density function plot				
Name	**normal**	**log-normal**	**exponential**	**Pareto**
Parameters	mean μ, variance $\sigma^2 > 0$	mean μ, variance $\sigma^2 > 0$	rate $\lambda > 0$	scale $x_{min} > 0$, shape $k > 0$
Support	$x \in (-\infty, +\infty)$	$x \in (0, +\infty)$	$x \in [0, +\infty)$	$x \in [x_{min}, +\infty)$
Cumulative distribution	$\frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$	$\frac{1}{2} + \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{\ln(x)-\mu}{\sigma\sqrt{2}}\right)\right)$	$1 - e^{-\lambda x}$	$1 - \left(\frac{x_{min}}{x}\right)^k$
Probability density	$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$	$\lambda e^{-\lambda x}$	$\frac{k x_{min}^k}{x^{k+1}}$
Mean	μ	$e^{\mu + \frac{\sigma^2}{2}}$	$\frac{1}{\lambda}$	$\begin{cases} \infty & \text{for } k \leq 2 \\ \frac{x_{min}^2 k}{(k-1)^2(k-2)} & \text{for } k > 2 \end{cases}$
Variance	σ^2	$\left(e^{\sigma^2} - 1\right) e^{2\mu + \sigma^2}$	$\frac{1}{\lambda}$	$\begin{cases} \infty & \text{for } k \leq 2 \\ \frac{x_{min}^2 k}{(k-1)^2(k-2)} & \text{for } k > 2 \end{cases}$

observations. In this context, $\mathbb{P}(M|D)$ is called the *posterior probability*: the probability that our model is true after observing D. The central idea of Bayesian statistics is that probability is a subjective measure: it changes when we get more information. For example when we throw dices we initially assume that every outcome is equally likely. However, when we repeatedly roll the same number, chances that the dice is loaded are increasing. Thus, with new observations we change the probability of our initial hypothesis "the dice is fair".

Bayesian statistics is commonly used in combinatorial optimisation. In maximum a posteriori probability estimation we want to find a model M that maximises the posterior probability $\mathbb{P}(M|D)$. In combinatorial optimisation the model is usually the structure we want to derive, say a graph, a set, or a tree. D is our observation: the data we use to decide which hypothesis is most appropriate. Usually, we cannot calculate $\mathbb{P}(D)$, because we do not know the distribution of the measurements. But for maximising $\mathbb{P}(M|D)$ it is sufficient to maximise the nominator: $\mathbb{P}(M|D) \propto \mathbb{P}(D|M) \cdot \mathbb{P}(M)$. Here, $\mathbb{P}(D|M)$ is called the *likelihood*: the probability that we make an observation D if our model is true.

For estimating the posterior probability we still need the prior probability $\mathbb{P}(M)$. Often, this probability is not available or cannot be calculated easily. *Maximum likelihood* is a method that only maximises the likelihood, implicitly assuming a flat prior for all models. We can think of maximum likelihood as "letting the data decide and not our prior belief". Maximum likelihood is commonly used for estimating the parameters for parametric probability distributions from measured data.

A *probability distribution* is a function that assigns probabilities to a random variable. A *random variable* is a stochastic variable which possible values are outcomes of a random experiment. For example, when throwing a dice the corresponding random variable could be the number of pips on the top face of the dice. This would be a discrete random variable and the corresponding probability distribution would be discrete, too, representable as a histogram. When measuring outcomes of physical processes, we usually derive continuous random variables. A continuous probability distribution is described by a *cumulative probability function* F for which $F(x)$ is the probability of observing a value lower or equal to x. The derivative of the cumulative probability function is called the *probability density function*. There is no easy statistical interpretation of the density function evaluated at a single point. However, it is common, and we will do this repeatedly in this thesis, to use the density as relative likelihood: Given two values x_1 and x_2 and a probability distribution with density function f, we interpret $\frac{f(x_1)}{f(x_2)}$ as how much values around x_1 are more likely than values around x_2.

See Table 3.1 for the probability distributions which are used in this thesis.

3.3 Machine Learning

In this section we will introduce the basic principles of supervised machine learning as well as two popular machine learning methods that are used in this thesis: linear and kernel support vector machines, and deep neural networks. We recommend Friedman *et al.* [66] as comprehensive introduction in statistical learning theory, covering a much wider range of methods and algorithms.

3.3.1 Supervised Machine Learning and Classification

The general idea of supervised learning is to predict a *label* from a set of input variables, called *features*. We will denote the set of all possible features $\mathcal{X}$ and the set of all possible labels $\mathcal{Y}$. Features and labels can be numerical vectors, scalars, or discrete categories. In general, we can encode any kind of inputs and outputs into numerical vectors, although, such a mapping is more complicated for structural data like graphs or text documents. We call the learning task *regression* for $\mathcal{Y} = \mathbb{R}$ and *classification* for $\mathcal{Y} = \{-1, 1\}$. In this thesis we will focus on classification problems and in the following we will assume that the features are real valued vectors and the labels are either -1 or 1.

Supervised learning aims to find a predictor function $f : \mathcal{X} \to \mathcal{Y}$ from a (possibly infinite) set of functions (called the *hypothesis space*) $\mathcal{F}$. The predictor function f is learned on a set of training examples (also called *observations*) $\mathcal{T} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathcal{X}$ is the feature vector and $y_i \in \mathcal{Y}$ is the label of the i-th training example. We search for a function f that *generalises* well; that means it should predict not only the correct labels of the training examples, but also the correct labels for new observations.

Supervised learning can be formulated as *structural risk minimisation* problem [219]:

$$\underset{f \in \mathcal{F}}{\text{minimise}} \quad C \sum_{i=1}^{n} \mathcal{L}(y_i, f(\mathbf{x}_i)) + \Omega(f) \tag{3.2}$$

Here, the *empirical loss* $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a function that measures how good f predicts the labels of the training points; it returns lower values with increasing similarity between the prediction $f(x_i)$ to the real labelling y_i. The *regulariser* $\Omega : \mathcal{F} \to \mathcal{R}$ penalises functions

with high complexity. Such functions are vulnerable for *overfitting*; that means that the function adapts to patterns which are specific only for the training points, for example noise on the features or labels. A highly complex function tends to learn these noise patterns to minimise the empirical loss and, thus, might perform worse on new observations. The hyperparameter C controls the trade-off between the quality of the prediction and the complexity of the predictor function.

3.3.2 Linear Support Vector Machines

A linear support vector machine (SVM) is a supervised learning method for classification, where the hypothesis space $\mathcal{F}$ is the set of linear functions, the regulariser Ω is the l_2 norm, and the empirical loss $\mathcal{L}$ is the *hinge loss* $\mathcal{L}_{\text{hinge}}$ which is defined as:

$$\mathcal{L}_{\text{hinge}}(y, \hat{y}) = \max(0, 1 - \hat{y} \cdot y) \qquad y \in \{-1, 1\}, \quad \hat{y} \in \mathbb{R} \tag{3.3}$$

The resulting structural risk minimisation problem is

$$\underset{\mathbf{w},b}{\text{minimise}} \quad C \sum_{i=1}^{n} \max\left(0, 1 - (\langle \mathbf{w}, \mathbf{x} \rangle - b) y_i\right) + \tfrac{1}{2} ||\mathbf{w}||^2 , \tag{3.4}$$

where $\mathbf{w}$ is the vector of coefficients of the linear function, b is the bias term and $\langle \cdot, \cdot \rangle$ denotes the inner product. The classification function of a SVM, which outputs the class labels -1 and 1, is defined as $g(x) = sign(\langle \mathbf{w}, \mathbf{x} \rangle - b)$.

Historically, classification by linear functions was described geometrically as finding a hyperplane that separates the data points of the two classes. The first method to compute such hyperplanes was the perceptron [178]. It searches for a linear hyperplane that minimises the distance of misclassified training points from this hyperplane. However, if the training data is not linear separable, the method does not converge. But even if the data is separable by a linear model, there might be an infinite number of solutions which do not generalise equally well.

The *optimal separating hyperplane* searches for the separating hyperplane that maximises the distance to the closest training points [217, 218]. The region between the hyperplane and these closest training examples is called *margin*. Maximising the margin leads to an unique solution and it can be argued that of all possible hyperplanes the optimal separating hyperplane has the best generalisation property [66].

A hyperplane is a set of points $\mathbf{x}$ that satisfy the equation $\langle \mathbf{w}, \mathbf{x} \rangle - b = 0$, where $\mathbf{w}$ is the normal vector to the hyperplane and $\frac{b}{||\mathbf{w}||}$ is the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$. The distance of a point $\mathbf{x}$ from the hyperplane is $\frac{\langle \mathbf{w}, \mathbf{x} \rangle - b}{||\mathbf{w}||}$.

We want to maximise the margin M; that is the distance between the hyperplane and the closest training examples. In other words, for all training examples the distance to the hyperplane has to be equal or larger than M. The corresponding optimisation problem is

$$\begin{aligned} &\underset{\mathbf{w},b,M}{\text{maximise}} \quad M \\ &\text{subject to} \quad \frac{\langle \mathbf{w}, \mathbf{x} \rangle - b}{||\mathbf{w}||} y_i \geq M \quad \text{for all } i \in 1 \dots n \,. \end{aligned} \tag{3.5}$$

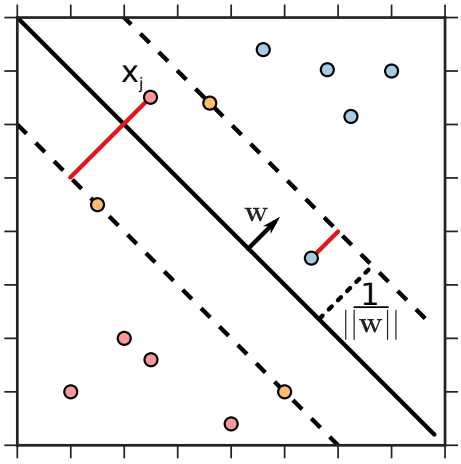


Figure 3.1: Example of a linear support vector machine. The hyperplane (black line) separates the feature space into two parts. The red dots belong to class -1, and the blue dots to class 1. The dashed lines denote the margin. The yellow dots on the margin are the closest training examples for which $y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle - b) = 1$ holds. The normal vector $\mathbf{w}$ is orthogonal to the hyperplane. The distance between the hyperplane and the margin is $\frac{1}{||\mathbf{w}||}$. There are two data points which are misclassified or within the margin. The red lines denote the distances from these points to the right side of the margin. We labelled one of these examples as x_j. This distance from x_j to the right side of the margin is equal to the assignment of the slack variables ξ_j and to the hinge loss $\mathcal{L}_{\text{hinge}}(y_j, \langle \mathbf{w}, \mathbf{x_j} \rangle - b)$.

We can rewrite eq. (3.5) as $(\langle \mathbf{w}, \mathbf{x} \rangle - b)y_i \geq M||\mathbf{w}||$. Because this equation is scaling invariant to $\mathbf{w}$ and b, we can define $||\mathbf{w}|| = \frac{1}{M}$. Thus, the optimisation problem is equivalent to

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimise}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & (\langle \mathbf{w}, \mathbf{x_i} \rangle - b)y_i \geq 1 \quad \text{for all } i \in 1 \dots n\,. \end{aligned} \tag{3.6}$$

However, there is still the problem that the training examples might not be linear separable. The *soft margin* was developed to solve that problem [41]. It introduces so called *slack variables* ξ_i for all training examples. These variables assign an error to observations that are within or on the wrong side of the margin.

The optimisation problem now minimises the sum over all slack variables and the norm of the function:

$$\begin{aligned} \underset{\mathbf{w},b,\xi}{\text{minimise}} \quad & C\sum_{i=1}^{n} \xi_i + \tfrac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & (\langle \mathbf{w}, \mathbf{x_i} \rangle - b)y_i \geq 1 - \xi_i \quad \text{for all } i \in 1 \dots n \\ & \xi i \geq 0 \quad \text{for all } i \in 1 \dots n \end{aligned} \tag{3.7}$$

Note, that this optimisation problem is equivalent to the structural risk minimisation in eq. 3.4. In particular, for any training example x_i the slack variable ξ_i for the optimal solution $\mathbf{w}, b, \xi$ is equal to the hinge loss $\mathcal{L}_{\text{hinge}}(y_i, \langle \mathbf{w}, \mathbf{x_i} \rangle - b)$. See Fig. 3.1 for a depiction of a linear support vector machine.

The hinge loss (and, analogously, the slack variables) is zero for training examples outside the margin. This results into the characteristic property of a SVM that the prediction

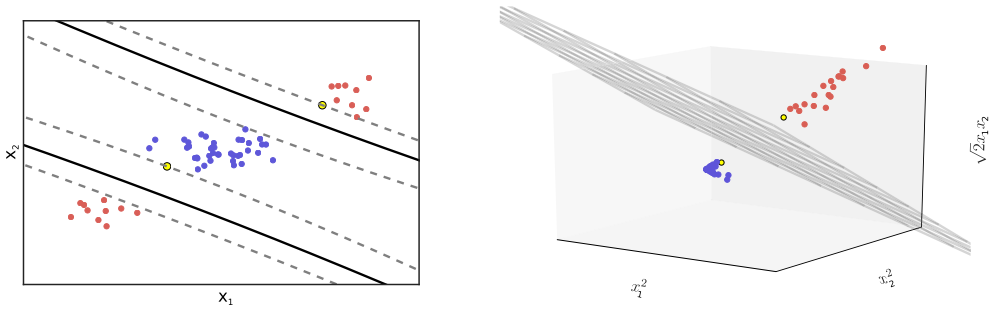


Figure 3.2: Example for a kernel support vector machine. Left: The two dimensional feature space with positive data points in blue and negative data points in red. The yellow points are the support vectors. The black line is the projection of the three dimensional hyperplane in the two dimensional space and the dashed lines are the margin. A two dimensional linear slope would not be able to separate both classes. Right: The three dimensional quadratic transformation using the function $\phi(\mathbf{x}) = (\mathbf{x}_1^2, \mathbf{x}_2^2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2)^T$. In this higher dimensional space, the data points can be separated by a linear plane.

function can be described solely by points within the margin (or misclassified points). This becomes apparent if we look at the Lagrangian dual of the optimisation problem in eq. (3.7) [187]:

$$\begin{aligned} \underset{\alpha_1 \ldots \alpha_n}{\text{maximise}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \alpha_i \langle \mathbf{x_i}, \mathbf{x_j} \rangle y_j \alpha_j \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \quad \text{for all } i \in 1 \ldots n \end{aligned} \tag{3.8}$$

Here, $\alpha_1 \ldots \alpha_n$ are the Lagrange multipliers. α_i is 0 if x_i is outside the margin and $\alpha_i > 0$ if x_i is lying on or within the margin. Training examples x_i with non-zero α_i are called *support vectors*. From the dual we can easily obtain the solution $\mathbf{w}$ from eq. (3.7) as $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}$. Furthermore, the distance of a point $\mathbf{x}$ to the hyperplane can be computed directly as $\sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x_i}, \mathbf{x} \rangle - b$. This means that we never have to compute the hyperplane $\mathbf{w}$ but can instead evaluate the inner product between any observation and all support vectors. We will use this property in the next section to deal with highly nonlinear problems that cannot be separated by a linear hyperplane.

3.3.3 Kernel Support Vector Machines

So far, we only used classification functions that are linear in the feature space. However, for most problems the unknown mapping from input to output is highly nonlinear. A simple trick to introduce nonlinearity is to transform the feature space $\mathcal{X}$ into a (possibly high dimensional) feature space $\mathcal{F}_x$ using some transformation function $\phi : \mathcal{X} \to \mathcal{F}_x$. In this transformed feature space $\mathcal{F}_x$, the problem might get linear separable. See Fig. 3.2 for an example of two-dimensional data points which are transformed into a three dimensional space using a quadratic function.

However, when adding all quadratic combinations of features, the dimensionality of $\mathcal{F}_x$ grows quadratically with dimensionality of $\mathcal{X}$ and we will blow up the number of features and, therefore, running time and memory usage of the method. This becomes apparent for highly structured data like graphs where a meaningful transformation ϕ results into very high or even infinite dimensional spaces. Kernels are introduced to solve this problem. While kernels can be used in many machine learning applications, we will focus solely on kernel support vector machines.

A kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a continuous, symmetric and positive semidefinite function; that means the following conditions hold:

$$K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}) \quad \text{for all } \mathbf{x}, \mathbf{x}' \in \mathcal{X} \tag{3.9}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \quad \text{for all } m \in \mathbb{N}, \text{ and all } \{(x_i, \alpha_i)\}_{1=1}^{m} \subseteq \mathcal{X} \times \mathbb{F} \tag{3.10}$$

For every such kernel function K, there exist a transformation ϕ into a Hilbert space $\mathcal{F}_x$, such that the kernel computes the inner product in this Hilbert space [86]:

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{F}_x} \quad \text{for all } \mathbf{x}, \mathbf{x}' \in \mathcal{X} \tag{3.11}$$

Note that $\mathbf{x} \in \mathcal{X}$ is no longer necessary a numerical vector. It can be any kind of data, say an image, text document, or graph, as long as ϕ maps the data to a Hilbert space $\mathcal{F}_x$ for which an inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}_x}$ is defined.

In eq. (3.8) the input features only occur as inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. We can replace this expression by a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. In many cases it is possible to directly compute K without even evaluating the transformation ϕ. This is called *kernel trick*. For example, the function $K_{\text{poly}(1,2)}(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^2$ is the quadratic kernel. It implicitly computes the inner product of the input features and all quadratic feature combinations for two training points. It does so, without computing any of these feature combinations explicitly.

Given a set of training examples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and a kernel function K, the kernel matrix $\mathbf{K}$ is a $n \times n$ Gram matrix defined as:

$$[\mathbf{K}]_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{F}_x} \tag{3.12}$$

For training a kernel support vector machine we just need the kernel matrix $\mathbf{K}$ and the labels $y_1, \ldots, y_n$, but neither the feature vectors $\mathbf{x} \in \mathcal{X}$ nor the transformed feature vectors $\phi(\mathbf{x}_i) \in \mathcal{F}_x$ have to be given for training. However, to predict the label of a new observation $\mathbf{x}'$, we have to be able to compute the vector $(K(\mathbf{x}', \mathbf{x}_1), \ldots, K(\mathbf{x}', \mathbf{x}_n))$. For prediction, kernel support vector machines usually have to store the training examples in memory.

Kernel machines allow highly nonlinear transformations of the feature space and, therefore, nonlinear predictor functions. But this comes to a price: the memory usage and computation time of kernel support vector machines scale with the number of training examples; space requirement is $O(n^2)$ and training (without the computation of kernels) is performed in $O(n^3)$ with n is the number of training examples [30].

Because every continuous, symmetric, and positive semidefinite function is a valid kernel, we can use any similarity function as kernel as long as it satisfies these conditions. The desired property of a good kernel is that data points that are related to each other (i.e.

share the same class label) are also close to each other in the transformed feature space and, therefore, have a high similarity. In the following, we will introduce some popular kernel functions.

Linear Kernel The linear kernel is simply defined as $K_{\text{lin}}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x} \rangle$. It can be used if the number of features is larger than the number of training points. In this case, solving the dual problem (eq. 3.8) might be faster and requires less memory.

Polynomial Kernel The polynomial kernel expands the input space by adding combinations of the input features. It is defined as $K_{\text{poly(c,d)}}(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + c)^d$. Here c is a hyperparameter trading off the influence of higher-order versus lower-order terms in the polynomial and d is the degree of the polynomial.

Radial Basis Function The Gaussian radial basis function (rbf) kernel is defined as $K_{\text{rbf}}(\mathbf{x}, \mathbf{x}') = e^{-\gamma ||\mathbf{x}-\mathbf{x}'||^2}$. In general, the rbf kernel can be applied to any distance metric $D : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as $K_{\text{rbf}}(\mathbf{x}, \mathbf{x}') = e^{-\gamma d(\mathbf{x}, \mathbf{x}')}$. This allows us to plugin any kind of data into the kernel as long as we can measure their pairwise distance with some metric. The parameter γ controls the locality of the kernel: A small γ means that the prediction of a new observation is only influenced by very similar training examples. The rbf kernel is a popular kernel that shows very good performance in a wide range of settings.

Histogram and Probability Product Kernels The histogram kernel is quite popular in image recognition [12]. But it can be used whenever the input can be represented in a histogram. There are two ways to compare histograms: We can sum up for each bin the minimum density in both histograms. This is called *histogram intersection kernel* [12]. We can also multiply for each bin the densities in both histograms and sum them up. Generalising the histogram kernel to any continuous probability distribution leads to the probability product kernel [96]: For two probability distributions $p(x)$ and $p'(x)$ we define the probability product as $K_{\text{pp}}(p, p') = \int p(x) \cdot p'(x) dx$. The probability product kernel has a nice statistical interpretation: It yields the expectation of one distribution under the other [96]: $K(p, p') = E_p[p'(x)] = E_{p'}[p(x)]$.

Kernel Combinations We have defined the polynomial kernel as a kernel taking vectors as input. But because the kernel is using the inner product of its input, it can be applied on any other kernel. For example, given any kernel K^* we can define a polynomial kernel of degree 2 as $K_{\text{poly(0,2,K}^*)}(\mathbf{x}, \mathbf{x}') = (K^*(\mathbf{x}, \mathbf{x}'))^2$. Similarly, we can apply the rbf kernel on any other kernel K^* as $K_{rbf}(\mathbf{x}, \mathbf{x}') = e^{-\gamma(K^*(\mathbf{x},\mathbf{x}) + K^*(\mathbf{x}',\mathbf{x}') - 2 \cdot K^*(\mathbf{x},\mathbf{x}'))}$. Many other kernels share this property: they can be applied not only on the input features but also on other kernels and, therefore, on (possibly high dimensional) transformations of the input features. A very simple kernel combination is the *sum* of kernels. Adding two kernels is equivalent to concatenating their transformed input feature vectors. We can also multiply two kernels, which is equivalent to a quadratic kernel that combines every feature from one transformed feature space with every feature from the other transformed feature space.

Decomposition Kernels A special family of kernels we will use throughout this thesis are decomposition (or convolution) kernels [77, 129]. They operate on inputs $\mathbf{x} \in \mathcal{X}$ that can

be decomposed into a (multi)set of features from a feature space $\mathcal{V}$. We say that $v \in \mathbf{x}$ are the parts of $\mathbf{x}$. Given a kernel $K_\mathcal{V}$ that compares two parts from the feature space $\mathcal{V}$, we can define a kernel on $\mathcal{X}$ as:

$$K_\mathcal{X}(\mathbf{x}, \mathbf{x}') = \sum_{v \in \mathbf{x}} \sum_{v' \in \mathbf{x}'} K_\mathcal{V}(v, v') \tag{3.13}$$

There is a large variety of kernels available in literature. The primarily goal of kernels is not to increase the dimensionality (and, therefore, make data separable), but to restrict the set of possible predictor functions to a meaningful subset using prior knowledge about the domain of the problem. For example, image data is already very high dimensional and often linear separable. But classification performance on images can be improved when using bag of visual words [134] or spatial pyramid kernels [115]. These kernels add the prior that features in the image (e.g. an eye and a nose) that are related to each other are also located close to each other.

Multiple Kernel Learning

As mentioned before, adding or multiplying kernels with each other yields again a valid kernel. Combining several kernels instead of just using the best kernel often results in better prediction performance. Each kernel can focus on different aspects or information of the input. For example, if we have an image with an image caption, we can combine kernels that operate on images with kernels that operate on texts. If the input are molecules, we can combine kernels that operate on structural properties with kernels that operate on physicochemical properties. But also several kernels on the same data representation can improve performance: each individual kernel might have some bias which can be reduced by the kernel combination. However, when we add more and more weak kernels together we might end up with a noisy kernel and larger prediction errors. Thus, we need some procedure that either selects or weights kernels. Such a procedure is called *multiple kernel learning* (MKL). See Gönen and Alpaydın [70] for an review to MKL with an overview of many MKL methods. Some MKL approaches learn the weights of the kernels together with the support vectors in one optimisation problem [161]. But in this thesis we will focus on the centred alignment MKL method, called *ALIGNF*, which learns the kernel weights in a separate step before learning the support vectors [42]. Shen *et al.* [197] presented an regularised variant of ALIGNF called *ALIGNF+*.

3.3.4 Deep Neural Networks

Kernels introduce nonlinearity into the regularised linear regression and classification methods. An alternative for allowing nonlinear functions are neural networks. Their research dates back to the 1940s when McCulloch and Pitts [126] introduced a calculus based on artificial neurons. These neurons were inspired by biological neurons: when their input signals exceed a certain threshold, they fire a signal. Mathematically, these neurons take the signum on the linear combination of input signals. The non-linear function, here the signum, is called the *activation function*. Later, the Rosenblatt perceptron applies this idea for classification [178]. In the following years, research focused on imitating the brains functionality, and the artificial neurons get more and more complex. This might be the reason why the invention of the backpropagation algorithm by Werbos [234] was initially

ignored by the community and critical recognised after its rediscovery by Rumelhart and Parker [152, 180]: it was clear that from a biological perspective backpropagation cannot be the way our brain is learning. However, 30 years after its discovery, backpropagation is still the main algorithm used for training artificial networks.

The most common neural network architecture is the multilayer perceptron or *feed-forward network*. A feed-forward network consists of at least three layers, where each layer can be metaphorical seen as row of artificial neurons. The input layer encodes the input and is equivalent to the vectorised representation $\mathbf{x}$ of the input discussed in Section 3.3.1. Analogously, the output layer encodes the output of the network and is equivalent to $\hat{\mathbf{y}}$. In between both layers there can be an arbitrary number of *hidden layers*. Each hidden layer is an internal representation of the input. The network learns with each layer a new embedding of the input into some vector space by applying a linear function followed by a non-linear (activation) function. If the network contains more than one hidden layer, it is called a *deep neural network* (DNN). Training such deep networks with backpropagation was a big problem for long time: with each layer the computed gradient either vanishes or explodes, resulting in numerical instabilities. The first breakthrough came with training each layer separately in an unsupervised pretraining [17, 85, 159]. Later, it could be shown that the vanishing gradient problem can be solved or at least reduced by using other activation functions like the hyperbolic tangens or, most popular nowadays, the rectified linear unit (ReLU) [69].

The research of neural networks nowadays is less inspired by biology; instead, artificial neural networks are now based on the foundation of statistical learning theory and training neural networks can be described as an optimisation problem like eq. (3.2) in Section 3.3.1. Neural networks are a general framework that allow for modelling a variety of different architectures for domain specific problems. For example, when using the hinge loss as loss function and the l_2 norm as regulariser, a feed-forward network without hidden layer becomes equivalent to a linear support vector machine. When adding a single hidden layer with a very large number of neurons, the networks starts resembling a kernel support vector machine [16].

Similar to kernel machines, the strength of neural networks lies in the flexibility that allows for designing different network architectures for different problems. It is very easy to build a network that can separate or even memoise the data; but much harder to design a network that generalises well. Deep neural networks are successful in image recognition by using convolutional neural networks and in language and speech recognition by using recurrent neural networks. Thus, special network architectures are crucial to achieve state-of-the-art performance.

For a comprehensive review of the history and applications of deep neural networks, see the review by Schmidhuber [185].

3.3.5 Evaluating Classification Methods

We have shown that supervised learning can be expressed as an optimisation problem that minimises the empirical loss. However, the loss functions we use for training are often neither intuitive nor suited for evaluating a predictor's performance. Here, we will present several measures for evaluating classification models. In particular, we want to evaluate the performance of a model that predicts the membership of an instance to a certain class. We can distinguish four possible cases: The *true positives* (TP) are all correctly predicted instances that belong to the class, while the *false negatives* (FN) are class members which

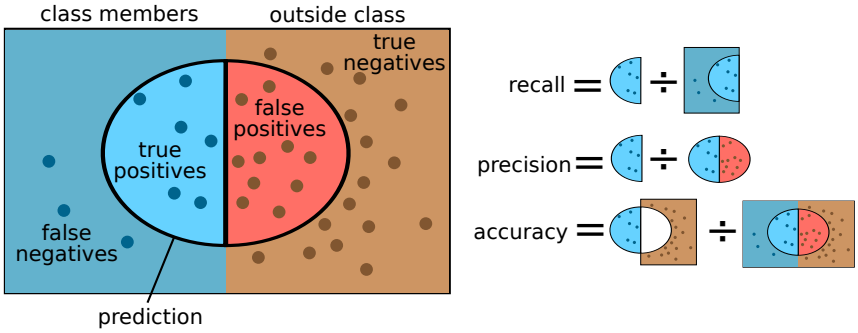


Figure 3.3: Graphical depiction of recall and precision. The left side of the square contains all instances belonging to the class; the right side are all instances outside the class. The circle in the middle contains all instances that are predicted to be in the class.

are predicted wrongly. The *false positives* (FP) are predicted as class member, but do not belong to the class. Finally, the *true negatives* (TN) are correctly predicted as not belonging to the class. These four cases are often visualised in a table with rows are corresponding to the real classes and columns to the predicted classes. Such a table is called *confusion matrix*.

Accuracy and F1 Score The most simple measure is the *accuracy* which is the relative frequency of correctly predicted labels. Accuracy is calculated as $(TP + TN)/(TP + FP + TN + FN)$.

Accuracy is unsuitable when the class membership is very imbalanced; for example, when there are much more instances outside than inside the class. For such biased data, we prefer other measures like recall, precision, and F1. For computing these measures, we always assume that most instances are outside the class. If this is not the case, one can simply look at the inverse problem. The recall is the ratio of correctly labeled class members to the total number of class members. We calculate it as $recall = TP/(TP+FN)$. The precision measures how many of the instances, for which we have predicted that they are member of the class, are correct. Thus, $precision = TP/(TP + FP)$. See Fig. 3.3 for a graphical depiction of these measures. The F1 score is the harmonic mean of recall and precision:

$$F1 = \frac{2 \cdot recall \cdot precision}{recall + precision} \tag{3.14}$$

Receiver Operating Characteristic The receiver operating characteristic (ROC) is a visualisation of classification methods over varying thresholds. A SVM classify training points with a decision value above zero as class members. However, when changing this threshold to a value above or below zero, we obtain other false positive and true positive rates. A ROC curve is drawn by plotting the true positive rate against the false positive rate over varying thresholds. The area under curve is a common measure to compare ROC curves of different predictors.

Cross-validation Cross-validation is used to estimate how "accurately" a predictor function will perform on the unseen data, and can also be used to estimate some external parameters specified by the user. One round of cross-validation involves partitioning

a dataset into two complementary subsets, performing the analysis on one subset (the training set), and validating the analysis on the other subset (the testing set). Multiple rounds of cross-validation are performed using different partitions, and results are averaged over the rounds.

An n-fold cross-validation corresponds to partitioning the dataset into n subsets of (almost) identical size. Then, n rounds of cross-validation are performed where in each round, $n-1$ subsets are concatenated to form the training set, and the remaining subset is used as the testing set.

An alternative to cross-validation is to use independent data for evaluation. If enough data is available, we recommend to do both.

4 Related Work: Computational Mass Spectrometry

In this chapter we will give an overview of the research done in computational mass spectrometry over the last years. We introduce the concept of fragmentation trees, which is vital for the methods presented in this thesis. But we will also explain other methods which are in use for analysing small molecules with mass spectrometry.

4.1 Preprocessing

There are several experimental setups possible in mass spectrometry, for example LC-MS or imaging mass spectrometry [127]. In general, for the identification of small molecules it is not important how the data is measured as long as MS^2 spectra with high mass accuracy are recorded. However, each experimental setup requires some preprocessing that transforms the measured data into a standardised format which then can be used by other analysis tools. These preprocessing steps consist of noise filtering, baseline correction, normalisation, feature detection and integration [102, 103, 207], feature grouping [31, 111] and retention time alignment [113]. Popular software frameworks that implement the preprocessing of LC-MS metabolomics data are OpenMS [179], MZmine [157], and XCMS [18, 208].

The output of the preprocessing is a feature table, where each row corresponds to one ion species that itself consists of several adduct species, in-source fragments, multimeres, and isotope peaks. This table is then used for quantitative analysis, bio marker detection, and statistical analysis.

In the following we want to focus on methods for annotating mass spectra. All following methods simply take preprocessed peak lists from MS^2 and the corresponding isotope peaks from MS as input.

4.2 De Novo Molecular Formula Annotation

A single measured peak in a mass spectrometry experiment can only reveal information about the elemental composition (molecular formula) of an ion. Structural isomers have indistinguishable masses. However, identifying the molecular formulas of measured ions is already a challenging problem: most peaks in an LC-MS run are ambiguous and can be explained by several molecular formulas, even when using instruments with high mass accuracy [106]. This is particularly the case for compounds above 400 Da, see Fig. 4.1. Without applying any constraints, the number of possible molecular formulas with n atoms composed of at most k different elements is $\binom{n+k-1}{k-1}$. Dynamic programming allows the fast enumeration of all molecular formulas within a certain mass range [27, 55]. Molecular formula constraints [82, 107] can be applied to reduce the diversity of possible explanations but cannot solve the underlying problem by themselves. It is understood that by applying

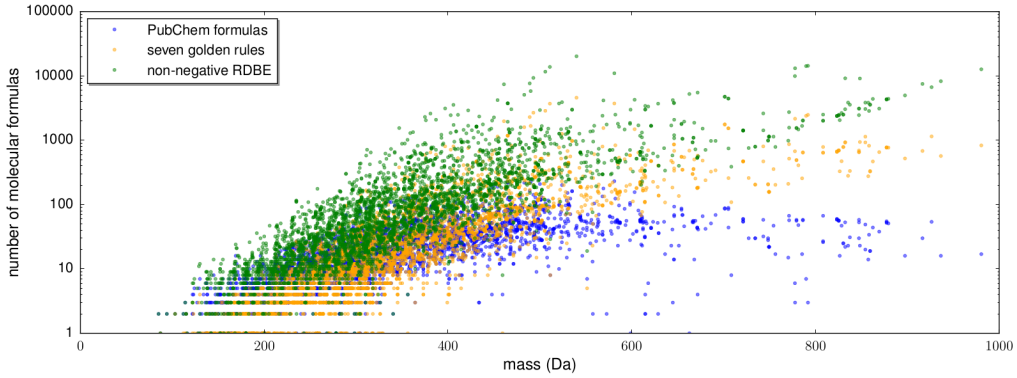


Figure 4.1: Number of molecular formulas that match the mass of some precursor peak in the Agilent and GNPS dataset (see Chapter 5), using the maximum of 10 ppm and 2 mDa as allowed mass deviation. Note the logarithmic scale of the y-axis. The green dots are all molecular formulas with positive RDBE, see eq. (2.1). More restrictive filtering such as the Seven Golden Rules [107] (orange) further reduce the number of molecular formulas to be considered; nevertheless, multiple explanations remain for most precursor ions. We find that 1.6 % of the compounds in our datasets *violate* the Seven Golden Rules. We also report the number of molecular formulas found in PubChem for the above mentioned mass accuracy. The figure is taken from [22].

more restrictive filters, we may filter out the correct molecular formula, limiting novel discoveries; this is particularly the case if we restrict ourselves to molecular formulas from some molecular structure database such as PubChem. Methods for identifying the molecular formula of an unknown compound usually require data beyond tandem mass spectrometry [130, 176, 177]. In particular, several methods successfully use isotope patterns for this purpose [7, 28, 107, 122, 151, 158, 214]. In contrast, network-based methods [138, 146, 231] do not aim at the identification of a single molecular formula or compound. While there are methods that can analyse MS^2 data, they still require qualitative isotope pattern [131, 158]. However, SIRIUS is a method for molecular formula identification that shows excellent performance even when restricted solely to MS^2 data. It provides algorithms for isotope pattern analysis [28, 117] in MS as well as a fragment ion analysis [163, 164] in MS^2. The fragment ion analysis involves the computation of fragmentation Trees which were introduced by Böcker and Rasche [25] in 2008. In the following, we will give a short introduction about fragmentation trees.

4.2.1 Fragmentation Trees

Fragmentation trees (FTs) are an annotation of the MS^2 spectrum. They model the fragmentation process that generates the fragment ions: Each node in the tree assigns a molecular formula to a fragment peak, whereas edges represent fragmentation reactions and are labelled with the molecular formula of the corresponding loss. Peaks for which no node exists in the tree are considered noise. The molecular formula of the root is the putative molecular formula of the precursor ion. See Fig. 4.2 for an example of a FT. Given an experimental spectrum, nodes and edges of a fragmentation tree can be scored according to how good the spectrum is explained by the tree [25]. Clearly, the term "fragmentation tree" has been used much earlier than 2008 in the MS literature [81, 168]; the important difference is that FTs in [25] are computed directly from the data by an automated method, without knowing the molecular structure of the compound, and *without*

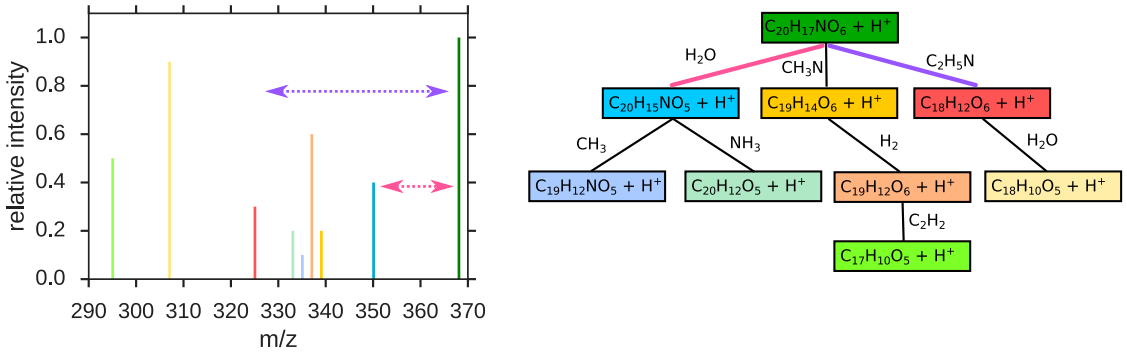


Figure 4.2: Example of a fragmentation tree (right) and the corresponding MS^2 spectrum (left). Each node in the tree corresponds to a peak in the spectrum with same colour. The edges in the tree correspond to mass differences in the spectrum.

the need for a database of tandem mass spectra or molecular structures. We stress that FTs are computed using tandem MS data; one can also do so using multiple MS data [182, 183] but this is not a requirement. This is fundamentally different from "spectral trees" [195] which solely describe the experimental setup of a multiple MS experiment; see the review by Vaniya and Fiehn [216] on the subject.

Fragmentation trees can be used to identify the molecular formula of a compound [25]: For each possible molecular formula explanation of the precursor ion, the fragmentation tree with maximum score rooted with this molecular formula is computed. The FTs and, hence, the molecular formulas are ranked according to the obtained scores. The scoring used in [25] was further evolved in [164] and [165], in particular by including a list of radical losses which are not considered implausible.

Rasche *et al.* [164] also showed that FTs can contain viable structural information about an unknown compound. In particular, computed FTs were manually evaluated by MS experts. For 79 FTs having a total of 808 losses, they found that more than 78 % of the losses were annotated as "correct" by MS experts. Rasche *et al.* [165] showed that FT *alignments* can be used to derive information about a compound's molecular structure, beyond the molecular formula of the compound: In particular, FT alignments can be used to search a spectral library for a compound which is structurally similar (but not identical) to the query compound, in case the query compound itself is missing from the database [47, 220].

In this thesis, we improved the scoring of FT computation and developed methods for structural elucidation based on fragmentation trees.

Computation of Fragmentation Trees The behaviour of metabolites in a collision cell is not fully understood. It is not clear how and in which order the ions fragment and, therefore, how the correct fragmentation tree should look like. Combinatorial optimisation is used to compute fragmentation trees: For each molecular formula explanation of the precursor ion, a *fragmentation graph* is built; that is a directed acyclic graph with nodes for each molecular formula explanation for any peak in the spectrum and with edges for all possible losses between two peaks in the spectrum. Such a graph can be easily built by enumerate over all peaks and decompose their mass using some user defined mass error

tolerance. Each of these formulas ends up as a node in the graph. For each two nodes u, v for which the molecular formula of v is a subset of the molecular formula of u an edge labelled with the molecular formula difference of both nodes (the loss) is inserted.

To define this as an optimisation problem, scores are added onto the edges and nodes of the graph. Böcker and Rasche [25] uses a rather ad hoc scoring scheme that penalises large mass errors, favours certain losses which are known as common (via expert knowledge), and puts more priority on high intensive peaks. Furthermore, edges with small losses are favoured over edges with large losses.

The optimal fragmentation tree is a subset of edges and nodes with maximum score where each node except the root has exactly one incoming edge. To ensure that each peak in the spectrum gets at most one explanation, all nodes that stem from the same peak are coloured with the same colour. Each colour is allowed to be used at most once in the tree. The computational problem underlying FT computation has been coined the Maximum Colourful Subtree problem [25]; unfortunately, this problem is NP–hard [166] and, therefore, cannot be solved in polynomial time unless P = NP. Nevertheless, there exist a number of algorithms (both exact and heuristic) to solve the problem in practice [25, 166, 235]. So far, the most efficient method for solving this problem is Integer Linear Programming (ILP) [166]:

A binary variable x_{uv} for every edge from u to v in the fragmentation graph is defined. The weight coefficient $w(u, v)$ is set to the weight of the edge uv plus the weight of the node v. We formulate the integer linear problem

$$\text{maximise} \quad \sum_{(u,v)\in E} w(u,v)\, x_{uv} \tag{4.1a}$$

$$\text{subject to} \quad \sum_{\substack{uv\in E \\ \text{with } c(v)=i}} x_{uv} \;\leq\; 1 \qquad \text{for all } i \in C, \tag{4.1b}$$

$$\sum_{vw\in E} x_{vw} \;\geq\; x_{uv} \qquad \text{for all } uv \in E, v \neq r \tag{4.1c}$$

$$x_{uv} \in \{\, 0, 1 \,\} \qquad \text{for all } uv \in E \tag{4.1d}$$

with C is the set of colours and $c(v)$ is the colour of node v. A solution of such an integer linear program is an assignment to each variable. If a variable is 1, the corresponding edge is part of the optimum fragmentation tree. The colour constraint (4.1b) forbids assignments in which a colour is used more than once. The tree constraint (4.1c) enforces that the resulting tree is connected. Note that the original definition by Rauf *et al.* [166] contained a third constraint that forbids cycles in the graph (and, thus, enforces a tree structure). However, White *et al.* [235] found that this constraint is redundant: If there is a cycle in the assignment, there have to be a node with two incoming edges. But if this happens, the colour constraint is violated.

Compared to the original Dynamic Programming algorithm from [25], the ILP is several orders of magnitude faster and, thus, can consider more peaks in the computation.

4.2.2 Isotope Pattern Analysis

Isotope pattern provide valuable information for mass spectrometry analysis: First, they can be used to distinguish noise from real metabolites; only metabolites have an isotope pattern. Furthermore, from the distance between isotope peaks we can derive the charge

of the ion: For single charged ion species, the distance between consecutive isotope peaks is roughly $m/z = 1$. For double-charged species it would be $m/z = \frac{1}{2}$.

Beside that, isotope peaks can also reveal additional information about the molecular formula of the metabolite. This can be done by simulating an isotope pattern for a hypothetical candidate molecular formula and then compare the simulated pattern with the measured one [7, 28, 107, 158]. Instead of determining the exact molecular formula, some methods focus on predicting the presence of rare elements (like halogens) from isotope patterns [133].

Simulating Isotope Patterns The intensities and masses of isotopologues can be calculated with polynomial expansion methods [33] or Fourier transform methods [171, 173]. However, as the number of terms (and isotopologues) is exponential in the number of atoms, reducing the search space is crucial. This can be done by pruning the search tree and only consider isotopologues with high abundance [122]. The instrument resolution is not high enough to detect all isotopologues. Instead, isotopologues with very similar masses are detected as a single peak. By only calculating the intensity for cluster of isotopologues with similar mass, the isotope pattern can be computed very efficiently [110]. Similarly, the mass of such cluster can be computed efficiently as weighted mean of the mass of all isotopologues within this cluster [26, 172].

Böcker *et al.* [26] uses the concept of discrete random variables and convolution for simulating isotope patterns with limited resolution. For metabolomics it is usually convenient to assume that all isotopologues with the same nominal mass are measured as a single peak. If several peaks are measured near the position of an theoretical isotope peak, they can be simply merged via weighted mean (with the downside of loosing the information of the single masses). An exception is sulphur: The mass of ^{34}S is strongly shifted to the left of its nominal mass and on high resolution instruments, isotopologues with ^{34}S are sometimes observed as individual peaks. We will deliberately ignore this case and assume that there is one isotope peak per nominal mass.

We say that an isotope pattern of a molecular formula X consists of an array of isotope abundances and an array of mass defects. The mass defect is the difference between the

Algorithm 1 Algorithm for convoluting two isotope patterns. Each isotope pattern consists of an array of mass defect values and intensity values.

function CONVOLUTION($M_1, \dots M_m$, $I_1, \dots I_m$, $M'_1 \dots M'_n$, $I'_1, \dots, I'_n$)
  let $\hat{M}$ and $\hat{I}$ be two arrays with size m+n
  for $i \leftarrow 1$ **to** m **do**
    for $j \leftarrow 1$ **to** n **do**
      $\hat{M}_{i+j-1} \leftarrow \hat{M}_{i+j-1} + (M_i + M'_j) \cdot (I_i \cdot I_j)$
      $\hat{I}_{i+j-1} \leftarrow \hat{I}_{i+j-1} + (I_i \cdot I_j)$
    end for
  end for
  for $k \leftarrow 1$ **to** $(m + n)$ **do**
    $\hat{M}_k \leftarrow \hat{M}_k / \hat{I}_k$
  end for
  return $\hat{M}$ and $\hat{I}$
end function

isotope mass and its nominal mass. For molecular formulas of a single atom both arrays are known: Oxygen, for example, has three isotopes and, therefore, three abundances and three mass defect values. See Table 2.1 for the abundances and mass defects. The convolution function (algorithm 1) takes two isotope pattern of arbitrary molecular formulas X and Y and performs an convolution of the underlying discrete random variables. The outcome is the isotope pattern for the molecular Formula $X + Y$.

For simulating the isotope pattern of any molecular formula we have to recursively split the molecular formula into two subsets, calculate their isotope pattern, and combine them with the convolution function in algorithm 1. Finally, we have to transform the (averaged) mass defect numbers into mass values. The i-th mass defect value is transformed by adding it with the nominal mass of the monoisotopic peak plus i.

4.3 Searching in Spectral Databases

The most widely used method for structure elucidation by tandem mass spectrometry is spectral library search [135]. This is in particular the case for GC-MS and electron ionisation data, because electron ionisation is known to generate highly reproducible spectra [203]. In contrast, LC-MS introduces a lot of variety: Ion trap fragmentation produces slightly different spectra than collision induced dissociation and the collision energy is an additional parameter that affects the fragmentation pattern.

Library search for LC-MS is performed by comparing the recorded spectrum against all library spectra with similar precursor ion mass. The most common scoring function for library search is the *cosine similarity*: here, each spectrum is normalised and binned into a vector. The dot product of both vectors is computed. The resulting score is between zero (both vectors are orthogonal; the spectra share no single peak) and one (both vectors/spectra are identical). Two modifications of the cosine similarity are repeatedly suggested: High mass peaks are often more specific than low mass peaks. This can be incorporated by multiplying the peak intensities with their mass, or the square of their mass [204]. Furthermore, intensities in spectra are exponential distributed: there are many low intensive peaks and very few high intensive peaks. To avoid that the similarity score is dominated by these few high intensive peaks, it is suggested to take the square root or logarithm of peak intensities [204]. Other scoring schemes like euclidean distance or probabilistic models were reported to be less effective [204].

Naturally, with library search we can only identify structures which were already recorded in some spectral library. Although the number of spectra in public libraries is growing very fast, there is a large overlap within these libraries in terms of the number of metabolites [223]. There is only a limited number of metabolite standards which can be obtained easily and it seems that these metabolites are measured over and over again. Shahaf *et al.* [194] have built a database by selecting interesting plant compounds that are structural very different from structures in other public spectral libraries. While this is a very promising approach for extending the space of reference spectra, it is very elaborate and expensive. So far, the number of metabolites in public and commercial spectral libraries is around 30 000 [223]. In contrast, the number of metabolites in the universal natural product library is above 150 000 and the PubChem contains almost 100 000 000 compounds.

This limited coverage of spectral databases has several implications: First, it means that only around 1.8 % of the ions in a mass spectrometry experiment can be annotated

by spectral library search [44] while we lack spectral reference data for the remaining ions. Second, spectral database search makes it impossible to discover new metabolites which were never measured before. Third, because we do not know if the true explanation of an ion is present in the database, we have to struggle with false-positive identifications: spectra that (randomly) match to a database spectrum but still are a different molecular structure.

Note that a database search will always return the database hit with maximum score, even if the true compound is not contained in the database. Various cutoffs and heuristics were developed to decide if a database search result is significant. For example, in GNPS a database hit is significant if at least 6 peaks are shared and the cosine similarity is above 70 % [227]. Recently, Scheubert *et al.* [184] presented three methods for significance estimation in spectral database search: In empirical Bayes a p–value is estimated by fitting two parameterised distributions (one for true hits and one for false hits) on the observed database scores [60]. The two other methods use a target-decoy approach: a decoy database is created that contains spectra that look like real spectra but which never contain a true hit. When searching in both databases, decoy and target database, one can estimate a false-discovery rate for a given score threshold by counting hits in the decoy database with score above the threshold. While decoy databases are common in proteomics [98], there was so far no method for creating decoy databases for metabolites. Scheubert *et al.* [184] suggest two approaches for generating decoy spectra; one is using our method for computing fragmentation trees which we will present in Chapter 6.

Analogue search is a technique that does not aim to find the exact molecular structure, but to find a very similar structure in a spectral library. Instead of searching spectral libraries for compounds with same masses, the search is either extended to certain biotransformations, or simply the whole database is searched through. The underlying idea is that spectral similarity correlates with structural similarity [47, 188]. Spectra can be compared with cosine similarity, by comparing the *inverse spectrum* [18, 47] (which is the spectrum we obtain by subtracting the precursor ion mass from every peak), or by spectral alignment algorithms [227]. Rasche *et al.* [165] has shown that aligning fragmentation trees instead of raw spectra improves the correlation to structural similarity.

Molecular networks are a graph representation of spectral data where MS^2 spectra with high similarity are grouped within a clique or a dense cluster in the graph [231]. When single ions can be identified, for example by analogue search, these identifications can be propagated to all highly similar MS^2 spectra in the dataset [45]. Similarly, MS^2 spectra can be clustered in a hierarchical tree, where clades correspond to structural similar compounds or compound classes [165]. Van der Hooft et al. [215] uses unsupervised learning to extract spectral features in a measured dataset. These spectral features can then be manually annotated and assigned to structural features [225].

4.4 Searching in Structure Databases

To overcome the coverage problem of spectral libraries, several methods for searching in structural libraries were proposed in the last years. This alone does not help for discovering novel metabolites which are not contained in any structure database. However, this can be achieved by screening in silico databases. Such databases are constructed either by enumerating over all combinatorial possible molecular structures with certain mass or

molecular formula [73, 153], or by applying biotransformations on known compounds [97, 198].

Many methods were proposed for searching in structure databases. We can distinguish three main paradigms:

Comparing Spectra against Structures Whenever we have some scoring method that compares a spectrum with a structure, we can screen structure libraries with a spectrum as input and rank candidates with matching mass according to the scoring. This approach requires both the query mass spectrum and the candidate structures to be given. Most scoring methods are based on combinatorial fragmentation: the molecular structure graph is disconnected in all possible ways and the resulting fragments are matched against the experimental spectrum. Initially, this approach was targeted at explaining the fragmentation of a known structure [78, 83]. MetFrag was the first method that used it to score and rank candidate structures for a given spectrum [241]. MAGMa speeds up the combinatorial fragmentation by representing fragments as bit sets [170]. MAGMa uses a considerably simple scoring: instead of using bond dissociation energy like Wolf *et al.* [241] or Hill and Mortishire-Smith [83], MAGMa uses predefined scores for disconnecting single, double, and aromatic bonds. Also MIDAS is using such a simple scoring [229]. MAGMa+, in contrast, uses some heuristics to select one of several predefined scorings [222]. MS-FINDER introduced hydrogen rearrangement rules [212].

In contrast to combinatorial fragmentation, rule based approaches use only fragmentation reactions from a limited, manual curated list of known chemical reactions. The commercial MassFrontier belongs to this category.

Predicting Spectra from Structures The second paradigm is to simulate an MS^2 spectrum *in silico*, then compare it to the experimental spectrum. Simulating a fragmentation spectrum requires that candidate structures are given, whereas the actual query mass spectrum is only needed at a later stage. Thus, it is possible to process whole structure libraries and generate in silico spectral libraries. The most common method for this task is CFM-ID (competitive fragmentation modelling) [5, 6]. Here, molecule fragmentation is modelled as Markov chain where in each step a bond or ring might disconnect. In contrast to combinatorial fragmentation, CFM-ID assigns probabilities for all possible fragmentation events and, thus, is able to predict relative intensities for the fragment ions. The underlying statistical model was initially a linear and quadratic regression model [5] and was later replaced by a neural network [6]. The model is trained using expectation maximisation [5].

A different approach are quantum chemistry methods which predict the MS^2 spectrum *ab initio* [14, 36, 71, 84]. The main limitation of these methods is the immense computation time necessary for quantum calculations; simulating a single spectrum takes hundreds of CPU hours [71]. Although, ab initio methods are useful for spectral annotation and understanding fragmentation pathways, evaluations so far have shown no advantage of quantum chemistry methods in comparison to the much faster CFM-ID [200].

Predicting Structural Features from Spectra Instead of predicting the mass spectrum from a known structure, methods following this paradigm try to predict the structure from the mass spectrum. Here, only the mass spectrum is taken as input. In theory, such approaches would allow for novel structure discovery, as they do not rely on structure

databases. However, so far no method is able to resolve a complete structure. Instead, structural properties are predicted from the spectrum which then can be used to screen structure databases.

FingerID and CSI:FingerID, one of the methods presented in this thesis, use kernel support vector machines to predict molecular fingerprints from MS^2 spectra. These fingerprints are then compared against the calculated fingerprints from database structures [79]. Input output kernel regression (IOKR) projects the input spectrum into the molecular kernel space and uses the euclidean distance to database structures as scoring. Mrzic *et al.* [140] use association rule learning to learn the relationship between spectral features and molecular substructures. ChemDistiller [114] combines combinatorial fragmentation and fingerprint prediction, but is using integral mass kernels instead of the high resolution kernels from [79].

It is remarkable that in silico generation of molecular structures, simulating MS^2 spectra as well as predicting structures from spectra was already pioneered by the DENDRAL project in 1970 [34, 63].

4.4.1 Annotation with Metadata

A popular approach for improving molecule identification is the integration of metadata like citation counts, industrial production rates or patents [20, 121, 144, 181, 191]. Statistically, these are priors that represent our knowledge about the probability to observe a metabolite in a sample. In environmental science such priors are very helpful: it is very likely to observe only such chemicals in waste water, which are massively produced in industry. Prior are never universal: the prior probability of a metabolite to be observed in a plant sample and in a waste water sample are completely different. Thus, when using priors we always have to check if they fit to our specific sample. A non-informative or flat prior is always preferable to an specific but inappropriate prior [119].

Using metascores like citation counts when evaluating on reference data will overestimate a methods performance and often results in very wrong conclusions, because evaluations are often done on well known standards [21]. A nice example is given in Little *et al.* [121], where the authors state that identifying large compounds above 600 Dalton is much easier, because there are fewer database entries in this mass region. However, there is no reasoning why nature should be less diverse in high mass region than in low mass region. It is more likely that our chemical databases are vastly incomplete in the high mass region.

It also has to be understand that such priors are completely useless for doing novel discoveries. Science subsists on the discovery of surprising, unexpected findings. Results with a very low prior probability are often of special interest. Results with high prior probability do only repeat what we already know.

4.4.2 Retention Time Prediction

Retention times recorded in LC-MS can be used to guide structure elucidation. This can be done by directly predicting retention times for the chromatographic system from reference data using regression methods [10, 13]. However, these predictions cannot be generalised on other platforms and are often limited to specific compound classes [139, 162]. Other methods first predict octanol–water partitioning coefficient (log P) from molecular structures and then map log P values to retention times using already identified compounds as reference. Although such a mapping is often not very accurate, it helps to reject or

confirm proposed structure candidates [87, 104] or improves the scoring [181]. A novel approach is the prediction of retention order, which is more robust and generalisable than retention time prediction [9]. Beside liquid chromatography, the relatively new ion mobility technique is a promising method that can be used to improve compound identification [132].

4.4.3 Structural Elucidation for Specific Compound Classes

Automated identification for metabolites is so challenging due to the structural diversity of small molecules. Many methods restrict the analysis to specific compound classes. Peptide identification is the foundation of proteomics. For unmodified peptides the problem is practically solved [169]: There are numerous methods for de novo sequencing of peptides, see [128] for a review about this topic. Furthermore, generating in silico databases for peptides using genome data is straightforward; giving a complete list of software methods is out of scope of this work, but see Nesvizhskii [143] for an review about this topic. Estimation of p-values and false discovery rates is much more advanced than in metabolomics [61, 112, 118, 209]. The current challenges in proteomics lie in detecting post-translational modifications [39, 189] as well as identifying peptides in mixtures of proteins of many species (metaproteomics) [141].

Peptidic natural products differ from proteomic peptides in that they are often not directly encoded in the genome and contain non-proteinogenic amino acids [136]. Furthermore, they may form cyclic peptides and, thus, cannot longer be represented as string. Peptide dereplication algorithms have to be adjusted for this special class of peptides [136, 145].

Glycosylation is the most common form of post-translational modifications. Glycans consists of simple building blocks but, different from DNA and peptides, cannot be represented as string but as tree. Still, many approaches from peptide sequencing can be adapted to glycans [224]. See Dallas *et al.* [46] for an review about automated analysis of glycans and glycopeptides.

Lipids cover a broad range of hydrophobic or amphipathic small molecules [62]. Their fragmentation behaviour is well studied and, thus, several methods for in silico generation of MS^2 spectra for lipids were developed [101, 108].

Note that methods which are developed for identifying metabolites in general can also be applied on metabolites of a specific compound class. In particular, machine learning methods like CFM-ID will show improved identification rates when trained exclusively on peptides [5]. Compound classes are a human concept; in nature we will often observe a smooth transition from one class to another one. Strict definitions for compound classes are often impossible. Some non-ribosomal peptides might look so different from other peptides, that tools developed for peptide sequencing might have a worse performance than tools that are developed for identifying arbitrary metabolites.

5 Mass Spectrometry Data and Benchmark Sets

Method development relies on publicly available datasets for evaluation and training. SIRIUS was originally evaluated on less than 200 spectra [25, 164]. Since then, the number of publicly available datasets is rapidly increasing. See [223] for a comprehensive review about mass spectrometry databases in metabolomics. We will give a short overview of the most important spectral libraries.

The largest open repositories are the Global Natural Products Social (GNPS) Public Spectral Libraries [227], MassBank [89], and the human metabolome database (HMDB) [240]. GNPS is a public dataset as well as a platform for analysing data. User can upload their experimental data and process it on GNPS. Tandem mass spectra are searched against the GNPS library, but also compared pairwise to cluster similar spectra. These clusters, together with potential database hits are visualised as molecular networks [243]. MassBank is a data repository for LC-MS and GC-MS. Originating in Japan, there exist also an European MassBank server called NORMAN and a North American MassBank server called MoNA. The latter is not only mirroring MassBank data but provides more detailed metadata. HMDB is a dictionary of human metabolites as well as metabolites that might be detected in human samples (e.g. drugs or food metabolites). For many metabolites it provides NMR and/or tandem mass spectra.

Next to these open repositories there are many commercial or closed repositories. The METLIN database contains tandem mass spectra of over 14 000 compounds (however, many of them are small peptides of length 2-3) [75]. METLIN also offers an analysis platform called XCMS Online [208]. While searching in the database is for free, there is no possibility to download the spectra from the database. The largest commercial library for mass spectrometry is the library of the National Institute of Standards and Technology (NIST). While originally a library for electron ionisation mass spectra, it nowadays contains also spectra for more than 16 000 compounds. The *Agilent* dataset is commercially available under the name "MassHunter Forensics/Toxicology PCDL" (version B.04.01) from Agilent Technologies Inc. (Santa Clara, CA, USA), and contains compounds of forensic and toxicological interest.

For training and evaluating our methods we use data from GNPS, MassBank, Agilent, and NIST. In total we have 50 704 measurements of 18 009 different structures available. Each measurement can be either a single spectrum, or multiple spectra of the same compound recorded on different collision energies. Spectra from Agilent and NIST are recorded on different collision energies. MassBank does also contain spectra measured in ramp mode. GNPS only provides the merged spectrum. To generalise over all these datasets and instruments as well as experimental setups we will ignore the collision energy and always merge spectra with different collision energies. Throughout this thesis, we will use the term *spectrum* synonymously for a merged spectrum from different collision energies. Similarly, we will use the term *structure* to refer to all compounds that have the same connectivity. We do not distinguish between stereoisomers within the spectral

Table 5.1: Datasets of tandem mass spectra used throughout the development of SIRIUS and CSI:FingerID. Given is the number of merged spectra that were used for training and/or evaluating the method. The CSI:FingerID 1.1 dataset is separated in positive mode "CSI:FingerID 1.1 (pos)" and negative mode spectra "CSI:FingerID 1.1 (neg)". "Current dataset" denotes the set of al MS^2 spectra from GNPS, MassBank, Agilent 2.0 and NIST. (*) The MassBank spectra were used in [58] for evaluation and served as independent dataset.

Database	GNPS	MassBank	Agilent	NIST 2017	Total
SIRIUS 3 [22]	2 005	1 333	2 046	–	5 384
CSI:FingerID [58]	4 138	625*	2 120	–	6 883
CSI:FingerID 1.1 (pos.)	7 385	754	–	10 979	19 118
CSI:FingerID 1.1 (neg.)	3 645	1 299	–	5 879	10 823
Large evaluation set (pos.)	13 147	2 030	3 387	15 429	33 993
Current dataset	17 903	3 362	3 968	20 462	45 695

libraries. One reason is, that stereoisomers are often not distinguishable solely by mass spectrometry. Orthogonal information (e.g. retention time) is necessary to separate different stereoisomers. Another reason is, that we cannot be sure that stereo information is always annotated correctly within public spectral libraries.

In this thesis we focus on high resolution mass spectrometry data. Our primarily analysis platforms are Q-ToF, Orbitrap and FTICR. We exclude measurements from low resolution instruments like quadrupoles or ion traps. Furthermore, we exclude GC-MS and electron ionisation data. We also exclude compounds from isotopic labelling experiments, as well as compounds with a mass above 1200 Da. When using public libraries we have to deal with missing or inconsistent metadata. We exclude all spectra from our training and evaluation where we cannot assign an InChI. Some spectra in these libraries also have very low quality: We exclude spectra where the fragmentation spectrum containing less than 5 peaks with relative intensity 2 % or above. For the training data of CSI:FingerID we applied more stringent filters: we remove spectra for which the fragmentation tree explains less than three peaks as well as spectra that have a low prediction quality during cross-validation.

As the number of data was constantly growing during writing these thesis, we have used different datasets for different experiments. We did not repeat every experiment with the newest data. See Table 5 for an overview of the number of spectra in all datasets that were used throughout this study. CSI:FingerID 1.1 is the up-to-date version of the method CSI:FingerID that is presented in Chapter 7. It is trained separately on positive mode and negative mode spectra. The evaluation set is the complete set of of MS^2 spectra that pass the above quality criteria. It contains spectra from Agilent which we are not allowed to use for CSI:FingerID, but also spectra which we filtered out from the CSI:FingerID 1.1 training set due to low spectral quality. The smaller evaluation set we used in [58] can be downloaded at `https://bio.informatik.uni-jena.de/data/evaluation_data-tar/`.

We received two versions of the Agilent dataset. The first one contains 2 046 MS^2 spectra but no isotope pattern and is used in [22]. Thanks to Frank Kuhlmann, we have now also access to a second version of this dataset that contains more compounds (3 387 positive mode spectra and 581 negative mode spectra) but also isotope pattern for most of the compounds. To distinguish both datasets, we will call the second dataset Agilent 2.0.

"Current dataset" denotes the set of all MS^2 spectra from GNPS, MassBank, Agilent 2.0 and NIST.

Table 5.2: Structure databases of biomolecules or compounds that can be expected in biological samples. We refer to compounds existing in any of these databases as *biocompounds*. The *biological database* is the union of these databases; the database containing all these biocompounds.

Database	Ref.	URL
KNApSAcK	[199]	http://kanaya.naist.jp/knapsack
HMDB	[240]	http://www.hmdb.ca
ChEBI	[76]	http://www.ebi.ac.uk/chebi
KEGG	[100]	http://www.kegg.jp
HSDB	[65]	https://toxnet.nlm.nih.gov/cgi-bin/sis/htmlgen?HSDB
MaConDa	[232]	http://www.maconda.bham.ac.uk
BioCyc	[35]	http://metacyc.org
UNPD	[72]	http://pkuxxj.pku.edu.cn/UNPD
biological subset of ZINC	[94]	http://zinc.docking.org
structures from GNPS	[227]	http://gnps.ucsd.edu
structures from MassBank	[89]	http://www.massbank.jp
MeSH-annotated PubChem	[105, 142]	http://pubchem.ncbi.nlm.nih.gov

Spectral databases cover only a small amount of existing metabolites. Structure databases record structural information and metadata of molecules. The largest structure database is PubChem with over 94 000 000 molecules [105, 142]. It provides large amount of metadata for each molecule, like references to PubMed entries or patents. Some molecules in PubChem are annotated with MeSH terms to link them to biological studies in PubMed [175]. Biological databases focus on metabolites and are often specialised on certain organisms. The HMDB database, for example, contains over 100 000 metabolites that either are produced in human, or might be detected in human [240]. BioCyc is a collection of databases that focus on different organisms, like EcoCyc that lists metabolites from Escherichia coli, or AraCyc that lists metabolites from Arabidopsis thaliana [35]. KNApSAcK lists metabolites from different organisms together with taxonomic metadata [199]. Chemical Entities of Biological Interest (ChEBI) is a dictionary of molecular entities focused on small compounds [76]. It provides large amount of metadata and a structural classification of all compounds. The Kyoto Encyclopedia of Genes and Genomes (KEGG) contains metabolomic networks together with gene, protein and metabolite information [100]. The Universal Natural Products Database (UNPD) is a database for natural products [72]. Not all molecules from these "biological databases" have to be produced in living organisms. The Hazardous Substances Data Bank (HSDB) is a toxicology database [65]. As mass spectrometry is used for detecting hazardous molecules in environment and drinking water, it makes sense to consider databases of such toxic compounds, too. Similarly, the Mass spectrometry Contaminant Database (MaConDa) lists molecules that regularly appear in mass spectrometry experiments (e.g. solvents) [232].

These biological databases are several magnitudes smaller than PubChem. In this thesis we will refer to compounds from any of these biological databases as *biomolecules* and we will denote the union over all these biological databases as *biological database*. This also includes molecules from HSDB and MaConDa. In total, the biological database contains 492 921 different structures. See Table 5.2 for a list of all used biological databases.

Critical Assessment of Small Molecule Identification

The Critical Assessment of Small Molecule Identification is a blind contest for identifying the molecular formula and structure of compounds from MS^2 data. The contest is hold every year since 2012. We attended with SIRIUS and CSI:FingerID several times at these contests. CASMI 2016 provided 208 challenges with 127 compounds recorded in positive mode and 81 in negative mode. We will use this large dataset for evaluating the methods presented in this thesis. The CASMI 2016 data can be downloaded from the CASMI 2016 website at `http://casmi-contest.org/2016/`. Furthermore, we uploaded the mass spectra from CASMI 2016 in a SIRIUS readable format at `https://bio.informatik.uni-jena.de/wp/wp-content/uploads/2017/11/casmi2016.tar.gz`.

6 Maximum A Posteriori Probability Estimation for Computing Fragmentation Trees

The scoring scheme by Böcker and Rasche [25] has certain disadvantages: It relies on an expert list of common losses which might be incomplete and it has several hyperparameters which were chosen ad hoc and were never evaluated on larger datasets. We refined the optimisation problem as maximum a posteriori probability estimation which we will discuss in Section 6.2. A statistical model has the advantage that its hyperparameters can be directly estimated from data. We also developed a method for learning the list of common losses directly from reference data using an approach similar to expectation maximisation. In Section 6.4 we refined the statistical model for scoring isotope patterns and show how to integrate the scoring of isotope pattern in MS^2 into the maximum colourful subtree problem. In Section 6.7 we compare the identification rates of the new scoring with the method from [163] and show that we significantly improved on this task.

The maximum a posteriori estimation was published in [22] and was presented at the annual international conference on 'Research in Computational Molecular Biology' (RECOMB) in Warsaw, 2015 [54]. Since then, we further refined the scoring and introduced the new isotope pattern scoring.

6.1 Formal Introduction of Fragmentation Trees

First we will give a formal notation of the fragmentation trees, fragmentation graphs, and the underlying maximum colourful subtree problem. Note that most of this work was already done in [25], but we will give some more details that are necessary to formulate the problem as maximum a posteriori probability estimation.

Our *data* $\mathcal{D} = (\mathcal{M}, I)$ is a measured fragmentation spectrum with peak masses $\mathcal{M} = \{m_1, \ldots, m_L\}$ and peak intensities $I : \mathcal{M} \to \mathbb{R}_{>0}$. Masses are not measured with arbitrary precision: To decide whether some theoretical molecular formula may coincide with some measured peak, we use a relative mass accuracy parameter MA provided by the user. Some peak with mass m and a molecular formula with mass m' match if $|m' - m| \leq MA \cdot m$. Usually, the mass accuracy parameter MA is provided as "parts per million" (ppm); for mass accuracy 5 ppm we have $MA = 5 \cdot 10^{-6}$. For small masses below some threshold parameter $m < m_{MA}$, we instead check $|m' - m| \leq MA \cdot m_{MA}$. Fragmentation spectra are relatively sparse: For any interval of 1 Da in the spectrum, there are at most a few peaks present. On the other hand, we demand that the mass accuracy of the measurement is high, say, 20 ppm or better. To this end, almost all theoretical molecular formula can explain *at most one* peak in the measured spectrum. See below for the very rare exceptions to this rule.

A *fragmentation tree* $\mathcal{T} = (V, E)$ consists of a set of nodes V which are molecular formulas over some alphabet of elements, and directed edges connecting these nodes. All

edges are directed away from the root of the tree, and every node can be reached from the root via an unique series of edges. Therefore, the FT is an arborescence. In small compound fragmentation, many fragments result from fragmentation cascades, that is, series of subsequent fragmentation events; these cascades are modelled by the tree structure of the FT. Nodes of the FT are molecular formulas of the parent ion and its fragments. For any FT, each molecular formula can appear at most once as a node of the tree. Edges are labelled with the molecular formula of the loss. In especially, the molecular formula of an edge is the difference of the molecular formula of its adjacent vertices. We say that for each edge $(u, v) \in E$, $u - v$ is the molecular formula of the corresponding loss; we demand that $u \geq v$ (the molecular formula u is a superset of v) holds for each component. Let $\mu(f)$ denote the theoretical mass of the molecular formula f (either fragment or loss). This will be usually the mass of the lightest naturally occurring isotope of an element, such as $\mu(\mathrm{H}) = 1.007825$. For a given FT, we can simulate a fragmentation spectrum (without intensities), simply using the masses of all nodes' molecular formulas. For the inverse direction, a FT is supported by a fragmentation spectrum of a compound if, for every node of the tree, we find a peak in the spectrum such that the mass difference between the molecular formula of the node and the peak mass is below some user-defined threshold. Recall from the above that there can be at most one such peak. Not all peaks of the fragmentation spectrum have to be explained by the tree, as we also have to model noise peaks in the spectrum. But we demand that for every node of the FT, there is a peak in the spectrum.

By modelling the compound fragmentation as a tree, we make the implicit assumption that each fragment in the fragmentation spectrum is generated by a single fragmentation pathway. In practice, different fragmentation pathways may lead to fragments with identical molecular structure. The most prominent example is that two fragmentation events happen independently and in arbitrary order: We call this a "parallelogram" spanned by the losses a, b, and $a + b$. For the FT, we focus on the most important fragmentation process that does possibly not contain all fragmentation events, but all major fragmentation events that mainly occurred. This is a slight oversimplification of the problem, but applying the parsimony principle is necessary to formulate the task as an optimisation problem. Regarding parallelograms, we note that these are implicitly encoded in the FT, as we can re-insert edges (losses) that correspond to such fragmentation events.

There is one additional requirement we need: We demand that every node of the FT explains a *unique* peak in the spectrum. In other words, no two nodes of the tree may correspond to the same peak. Allowing more than one node to explain a peak, would violate the vast majority of observations: In theory, it is possible that two fragments of a compound have different structure but very similar mass, so that both fragments explain the same peak. In practice, this situation is extremely rare, and excluding this "pathological" cases is again necessary to formulate our task as an optimisation problem: the improvement by making this assumption outweighs the cases where it leads to a possible incorrect interpretation.

We now formalise our above considerations. We say that a FT $\mathcal{T} = (V, E)$ is *supported by* the observed data $\mathcal{D} = (\mathcal{M}, I)$ if each node $v \in V$ is assigned a unique peak $m \in \mathcal{M}$ in the fragmentation spectrum that is within the chosen mass accuracy. Furthermore, no two nodes are assigned the same peak. We denote the *natural injective mapping* from the FT nodes to the peaks by $m : V \to \mathcal{M}$. All peaks in the spectrum not assigned to a node of the FT, are regarded as noise peaks. Our task is to find a FT that "best explains" the

observed data, where goodness-of-fit is measured by some scoring function (such as the posterior probability estimate considered below) that matches FT and mass spectrum.

This formulation of the problem is not easily accessible by algorithmic means; to this end, we use an alternative formulation which, for additive scorings, is equivalent to the above [25]: For each peak in the fragmentation spectrum, we find all molecular formulas with mass difference sufficiently small. These molecular formulas are the nodes of a directed acyclic graph (DAG) called *fragmentation graph*. Nodes are coloured so that all molecular formulas corresponding to the same peak have the same colour. Recall that we must use at most one node for each colour (peak) in our FT. Edges are inserted whenever one molecular formula is a subformula of another. Edges are appropriately weighted using some score function. It is straightforward to check that there is a 1-1 correspondence between *colourful* subtrees, that use every colour in the graph at most once, and FTs supported by the data. We search for a colourful subtree of this graph that has maximum weight.

To identify the molecular formula of the unknown compound, we can add a super-root that is connected to all molecular formula explanations of the parent ion peak. As all of the corresponding nodes share the same colour, only one interpretation of the parent ion peak will be present in the optimal solution. In practice, it turns out to be faster to instead consider one molecular formula for the parent ion peak at a time, compute for each such candidate an optimal FT, and rank the resulting trees according to their posterior probability.

We have deliberately ignored that the mass difference between two measured peaks in $\mathcal{D}$ may be smaller than twice the chosen mass accuracy; in this case, two peaks would be assigned the same molecular formula in the fragmentation graph and, possibly, also the maximum colourful subtree, violating our condition that all nodes have to be different molecular formulas. In practice, this situation will show up extremely rarely for mass accuracy of 10 ppm or better. If this "pathological" situation turns up, we split the mass range between the two measured peaks in half, so that any molecular formula is forced towards the closer measured peak.

6.2 Maximum A Posteriori Probability Estimation

Scorings in [25, 164, 165] were motivated by stochastic considerations, but only in an informal way. Here, we will strictly model the problem as a Maximum A Posteriori estimation, which allows us to make sensible choices for the (hyper)parameters of the method. Bayesian Statistics tell us that

$$\mathbb{P}(\mathcal{T}_j|\mathcal{D}) = \frac{\mathbb{P}(\mathcal{D}|\mathcal{T}_j) \cdot \mathbb{P}(\mathcal{T}_j)}{\mathbb{P}(\mathcal{D})} = \frac{\mathbb{P}(\mathcal{D}|\mathcal{T}_j) \cdot \mathbb{P}(\mathcal{T}_j)}{\sum_i \mathbb{P}(\mathcal{D}|\mathcal{T}_i)\, \mathbb{P}(\mathcal{T}_i)}\,, \tag{6.1}$$

where $\mathcal{D}$ is the data (the measured spectrum) and $\mathcal{T}_j$ are the models (the candidate FTs). We want to maximise the posterior probability $\mathbb{P}(\mathcal{T}_j|\mathcal{D})$ which is equivalent to maximising $\mathbb{P}(\mathcal{D}|\mathcal{T}) \cdot \mathbb{P}(\mathcal{T})$ over all possible models $\mathcal{T}$. Here, $\mathbb{P}(\mathcal{D}|\mathcal{T})$ is the probability of the data given the model $\mathcal{T}$, and $\mathbb{P}(\mathcal{T})$ is the *prior probability* of model $\mathcal{T}$, based on prior information that we have about FTs without considering the actual data $\mathcal{D}$. We have considerable background information about the prior probability of any given FT: For example, smaller losses are usually more frequent than larger losses for low and medium energy fragmentation, and certain losses such as H_2O or CO turn up very frequently. We will call all parameters related to these background information *hyperparameters*. Note,

that in the area of machine learning, this term has a different meaning and refers to parameters that are (often manually) chosen by the user; in contrast to model parameters which are optimised on data. In Bayesian statistics, however, the term hyperparameters refers to parameters of the prior distributions. We will name parameters from the likelihood model that can be directly estimated from spectral data (using maximum likelihood) as *parameters*.

We have stressed repeatedly that we are interested in those FTs only that are supported by the data. To this end, we demand $\mathbb{P}(\mathcal{D}|\mathcal{T}) = 0$ and, hence, $\mathbb{P}(\mathcal{T}|\mathcal{D}) = 0$ for any tree $\mathcal{T}$ that is *not supported by the data* $\mathcal{D}$. In the following, we assume that each considered FT is supported by the data.

We now introduce computations for prior probability and likelihood of the tree.

6.2.1 Prior Probability of the Tree

We first concentrate on the prior $\mathbb{P}(\mathcal{T})$. We will not demand that priors sum to one but only that the sum $\sum_i \mathbb{P}(\mathcal{T}_i)\,\mathbb{P}(\mathcal{D}|\mathcal{T}_i)$ converges, what is sufficient for optimising $\mathbb{P}(\mathcal{T}) \cdot \mathbb{P}(\mathcal{D}|\mathcal{T})$. But this is obviously true: The number of models $\mathcal{T}_i$ we are considering is finite, as we only consider trees supported by the data. We assume that, for all trees of constant size, prior probabilities of the nodes and edges of $\mathcal{T}$ are independent so that

$$\mathbb{P}(\mathcal{T}) = \mathbb{P}(\text{size } |E| \text{ of the tree}) \cdot \prod\nolimits_{v \in V} \mathbb{P}(v) \cdot \prod\nolimits_{e \in E} \mathbb{P}(e).$$

Here, $\mathbb{P}(v)$ is the prior probability to see a particular *fragment* in a FT, and $\mathbb{P}(e)$ is the prior probability to see a particular *loss* in a FT. The independence assumption is obviously violated in reality, but allows us to come up with simple yet meaningful priors. We can simplify this equation, noting that every node of the tree except the root has exactly one incoming edge. For molecular formulas u, v let $P_{\text{edge}}(u, v)$ be the prior that fragment v *and* loss $u - v$ are simultaneously seen in the tree, and let $P_{\text{root}}(u)$ be the prior that the tree is rooted with molecular formula u. Then,

$$\mathbb{P}(\mathcal{T}) \propto \mathbb{P}(\text{size } |E| \text{ of the tree}) \cdot P_{\text{root}}(r) \cdot \prod_{(u,v) \in E} P_{\text{edge}}(u, v)\,, \tag{6.2}$$

where r is the root of $\mathcal{T}$.

Prior of the Root For the prior $P_{\text{root}}(r)$ of the root r we use the neutral molecular formula r, and the fact that certain molecular formulas are observed more often in molecular databases [107].

First, we filter out structurally impossible molecular formulas that have a negative RDBE; see eq. (2.1) as well as the discussion about RDBE in Section 2.1. For molecular formulas with negative RDBE there exists no fully connected molecular graph. This is one of the "Senior rules" [193]. Molecular formulas from biomolecules look different from random molecular formulas or molecular formulas of synthetic molecules. For example, biomolecules usually have a backbone of carbon atoms, surrounded by a cloud of hydrogen atoms. This results in a hydrogen-to-carbon ratio that is quite specific for biomolecules. Other elements like nitrogen or oxygen occur less frequently. Therefore, the ratio of heteroatoms (atoms which are neither carbon nor hydrogen) to carbon atoms is usually small for biomolecules. Biomolecules contain a relative small number of rings and double

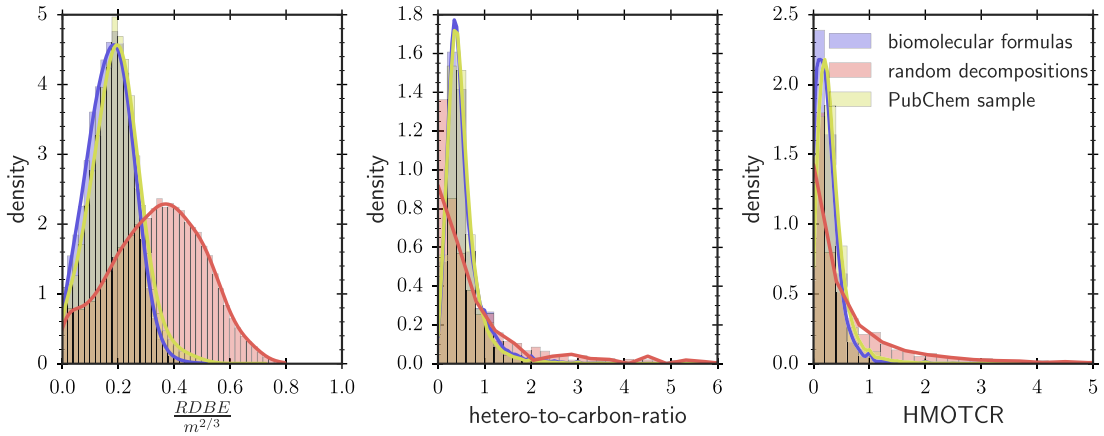


Figure 6.1: Histograms of different molecular properties with molecular formulas from the biomolecule database (blue), PubChem (yellow) and random decompositions of masses in the biomolecule database (red). Kernel density estimate is drawn as solid line. Left: Histogram of mass corrected RDBE. Middle: Histogram of hetero-to-carbon-ratio. Right: Histogram of hetero-minus-oxygen-to-carbon-ratio (HMOTCR).

bonds and, therefore, have a small RDBE value. The Seven Golden Rules [107] define upperbounds for the hydrogen- to-carbon ratio, the hetero-to-carbon ratio and the RDBE. However, strict upperbounds will omit some real existing molecules and will not penalise a lot of theoretical (non-bio)molecules that are close to the upperbound. Böcker and Rasche [25] found that the hetero-to-carbon ratio is normal distributed for biomolecules and suggested to use the density of this distribution as score. However, they did not use a scoring for the hydrogen- to-carbon ratio or the RDBE. We define our priors on the root based on the molecular formula properties that were reported as specific for biomolecules in [107] using probability distributions as it was done in [25]. The hyperparameters of these probability distributions can be estimated from databases of biomolecules. In [22] we used the Kyoto Encyclopedia of Genes and Genomes (KEGG) [99]. Later, we included additional biological databases (see Table 5.2) for hyperparameter estimation and classifier training.

We want to model the priors such that they penalise unlikely molecular formulas but not reward likely molecular formulas. For example, carbohydrates appear very frequently in biological databases. A molecular formula like $C_6H_{12}O_6$ would, therefore, receive a larger prior probability than $C_{10}H_{14}N_4O_4$. However, both molecular formulas are biomolecules. Rewarding one of them more than another without looking at the measured data cannot be justified. We can avoid such pitfalls by using two probability distributions per molecular formula property: one uniform distribution that covers 99 % of the biomolecules. And one non-uniform that penalises the remaining 1 %.

For example, we model the hetero-to-carbon ratio as a mixture of an uniform distribution for hetero-to-carbon ratios between 0 and 4 and as a Pareto distribution for larger values. We find that the hetero-to-carbon ratio becomes even more informative if we also exclude oxygen from the hetero atoms. We call this ratio "hetero minus oxygen to carbon ratio" (HMOTCR). For the RDBE value it is obvious that large molecules have more rings and double bonds than small molecules and, therefore, this value depends on the size of the molecule. We correct this size influence by dividing the RDBE by $m^{2/3}$ where m is the mass

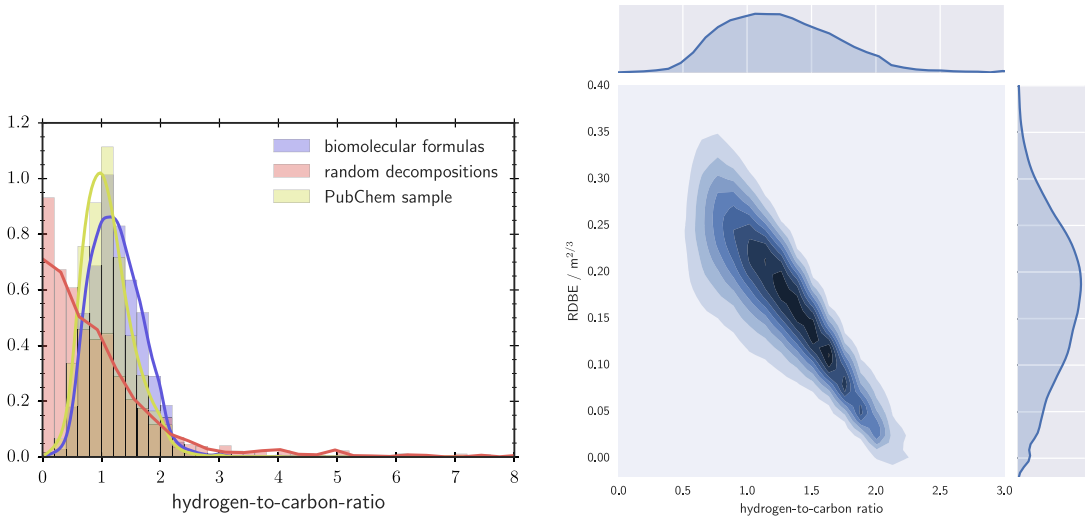


Figure 6.2: Left: Histogram of the hydrogen-to-carbon ratio with molecular formulas from the biomolecule database (blue), PubChem (yellow) and random decompositions of masses in the biomolecule database (red). Kernel density estimate is drawn as solid line. Right: Kernel density of hydrogen-to-carbon ratio and mass corrected RDBE with molecular formulas from the biomolecule database. Both molecular formula properties are correlated with a Pearson correlation coefficient of -0.76.

of the compound. We find that the resulting distribution is roughly normal distributed. See Fig. 6.1 for the histograms and kernel densities of these molecular formula properties.

In [22] we modelled the prior probability for a molecular formula as product of the densities of the corresponding probability distributions (with hyperparameters estimated on KEGG compounds). However, multiplying probabilities can only be justified if the probabilities are independent. Therefore, we have to exclude molecular properties from the prior that show high correlations. For example, a large hydrogen-to-carbon ratio usually results in a small RDBE, because the RDBE value becomes smaller for each hydrogen atom in the molecular formula and larger for each carbon atom (see eq. (2.1)). The Pearson correlation coefficient between the mass corrected RDBE and the hydrogen-to-carbon ratio is -0.76 within the biomolecule database (see Fig. 6.2). Therefore, the hydrogen-to-carbon ratio was ignored in [22].

Alternatively, we can train a linear support vector machine that combines several molecular properties via linear combination into a single scalar that is optimised for separating biomolecular formulas from random molecular formulas. See Table A.1 in the Appendix for the description of the used features. For training we use all molecular formulas from the biological database as positive set. The negative set is built by decomposing each mass from the positive set and randomly choose a decomposition which is not also contained in the PubChem structure database or any biological database. We further add molecular formulas to the negative set that clearly violate one of the Seven Golden Rules (e.g. which have an unusual high RDBE or a very large hetero-to-carbon ratio). We define the prior P_{SVM} as the posterior probability estimate of a molecular formula to be biological plausible by fitting a logistic function as proposed by Platt [155]. We evaluate this SVM predictor and the parametric distributions from [22] in a Receiver Operating Characteristic plot (see Fig. 6.3). The SVM predictor has an area under curve

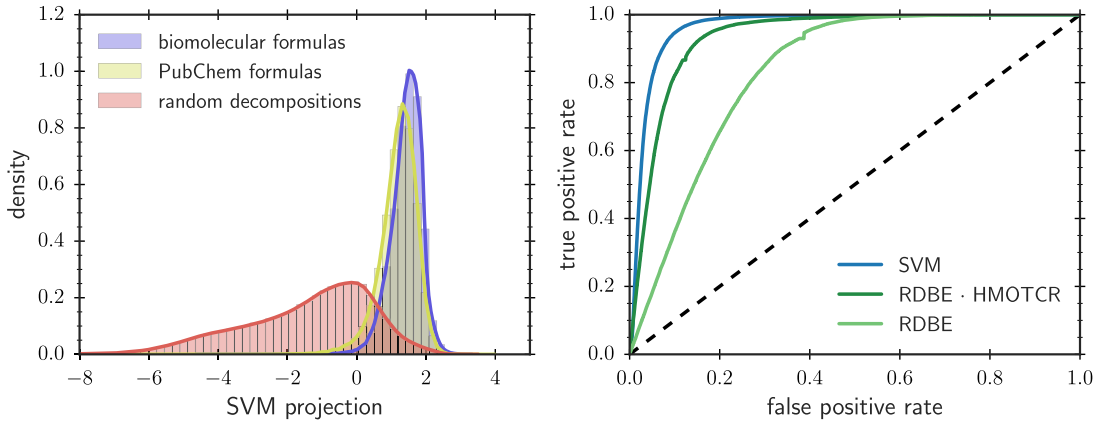


Figure 6.3: Left: Histogram of the linear combination of molecular formula properties learned by an SVM to separate biomolecular formulas (blue) from random molecular formulas (red). Molecular formulas from PubChem (yellow) show a similar distribution as the biomolecules. Right: Receiver Operating Characteristic (ROC) of the SVM predictor (blue) and the different estimated parametric distributions (green).

(AUC) of 0.965. The HMOTCR distribution has an AUC of 0.674 and the AUC of the mass corrected RDBE distribution is 0.833. The product of both parametric distributions has an AUC of 0.936, which is close to the SVM predictor. However, we are interested in a predictor with very high recall. If we enforce a recall of 99 %, the SVM predictor still has a false positive rate (FPR) of 21 %. This means that 4 out of 5 molecular formulas are correctly classified (and penalised) as biological implausible while only one of 100 biological molecular formulas is wrongly classified as implausible. Using parametric distributions we only reach an FPR of 50 %. Thus, the SVM predictor has only half as many false positive predictions. Note that the biomolecule database also contains contaminants, pesticides and other molecules that might look very different to most biomolecules. To reduce the number of false negatives, we add two additional tricks: First, we add all molecular formulas from the biomolecule database which were classified as biological implausible (550 formulas in total), into a 'whiteset'. These molecular formulas are never penalised by the SVM prior. Second, we never penalise molecular formulas with mass below 100 Da. This ensures that molecules like PO_4 or Na are not penalised.

Beside the machine learning approach, we can define some simple rules on molecular formulas based on chemical prior knowledge. If we assume that the compound is not a radical, then the sum of valences is even [193]. If the compound ion is protonated or has its charge due to some adduct, then the sum of valences of the ion is odd. Rejecting compounds with uneven sum of valences is also referred to as a "Senior rule" [193]. Free radicals such as nitrosyls have, in their protonated form, even sum of valences. But free radicals are comparatively rare; to this end, we use $P_{\text{RDBE-odd}} = 0.1$ for molecular formulas with even sum of valences, and $P_{\text{RDBE-odd}} = 1$ for all others. We note that this is not a true probability, because the probability to observe any particular value is zero for this continuous distribution; but this is not a problem when defining a prior, and will be repeatedly done in the following. Furthermore, the prior is not estimated from data but chosen ad hoc. We use a prior of 0.1 because this will penalise radicals while not forbidding them.

We add another prior for penalising molecular formulas containing "special" elements. We define all elements but C, H, N, O as *special*, as they occur less often in metabolites and natural products. We define $P_{\text{spec}} = 0.25^n$ with n being the number of special elements in the molecular formula. We stress that this is *not the number of atoms* corresponding to special elements: For example, $C_{17}H_{17}Cl_2N$ has $n = 1$. The basic idea behind this prior is to penalise the occurrence of special elements in the molecular formula if there are no further indications (in losses or fragments) for this element. We later add other edge and node priors to counter the effect of the special elements prior.

Additionally, we add a prior for penalising phosphorus-containing molecular formulas without oxygen or sulphur: We find that for more than 99 % of the phosphorus-containing compounds in the KEGG metabolite database, the sum of oxygen and sulphur atoms is at least twice the number of phosphorus atoms. We set $P_{\text{phos}} = 0.05$ for all compounds that violate this constraint, and $P_{\text{phos}} = 1$ otherwise.

The root prior

$$P_{\text{root}}(r) = P_{\text{SVM}}(r) \cdot P_{\text{RDBE-odd}}(r) \cdot P_{\text{spec}}(r) \cdot P_{\text{phos}}(r) \tag{6.3}$$

is the product of these four priors. Note that in [22] we used $P_{\text{HMOTCR}} \cdot P_{\text{RDBE}}$ instead of P_{SVM}. We stress that we never discard chemically feasible molecular formulas, and we never reward molecular formulas; we only penalise those that deviate too strongly from what we expect to see in a biomolecule.

Priors of Edges The prior probability $P_{\text{edge}}(u, v)$ of an edge $e = (u, v)$ is estimated from different factors, namely prior knowledge about implausible (and radical) losses, the mass of the loss, common losses, as well as common fragments:

$$\begin{aligned} P_{\text{edge}}(u, v) := & P_{\text{loss-impl}}(u, v) \cdot P_{\text{loss-mass}}(u, v) \cdot P_{\text{loss-comm}}(u, v) \cdot P_{\text{loss-spec}}(u, v) \cdot P_{\text{frag-spec}}(v) \\ & \cdot P_{\text{frag-chem}}(u, v) \cdot P_{\text{frag-mass}}(v) \cdot P_{\text{frag-comm}}(v) \cdot P_{\text{phos}}(v) \,. \end{aligned} \tag{6.4}$$

We first penalise *implausible losses* of an edge (u, v) using a prior $P_{\text{loss-impl}}(u, v)$ on the loss $u - v$. This is a small list of losses that repeatedly turned up during our combinatorial optimisation in [164], but that were rejected in the subsequent expert evaluation given there. In particular, we penalise losses that contain only nitrogen or only carbon; radical losses with certain exceptions; and few losses from a list of losses generated by expert knowledge. See Table 6.2.1 for the list of implausible losses and priors. Since these are losses that we *do not want to see*, there appears to be no reasonable way to learn such implausible losses from the data. Instead, we have to rely on expert knowledge and evaluation of FTs computed by the method, to collect this list. Also, priors for such implausible losses are chosen ad hoc as there appears to be no sensible way of learning such penalties from the data.

Regarding the mass of a loss, we assume that large losses are less likely than small losses. Unfortunately, there is only a very small number of annotated FTs available in the literature, and these are usually measured on different instruments (and instrument types) using different experimental setup and, hence, mostly incomparable. To this end, we estimate the loss mass distribution using FTs determined by SIRIUS. We will try to bring in agreement the observed distributions with the distribution used for scoring.

Different from [25, 164, 165] we do not penalise the relative size of the mass but rather the mass itself, as this allows for a more stringent incorporation of common losses.

Table 6.1: Priors for *implausible losses*. For an edge (u, v) with loss $u - v$ let $P_{\text{loss-impl}}(u, v)$ be the prior for $u - v$ chosen according to this table. Expert knowledge and evaluation of FTs from SIRIUS[2] resulted in the implausible losses listed here [164]. These losses should only very rarely (if ever) occur in a FT, so we manually select reduced priors.

Probability	Loss type and molecular formulas
10^{-3}	Implausible losses: C_2O, C_4O, C_3H_2, C_5H_2, C_7H_2
$\frac{1}{3^{RDBE}}$	Neutral losses with negative ring double bond equivalent RDBE
0.1	Nitrogen-only losses, carbon-only losses: for example, N_5 or C_3
1	All other neutral losses
0.9	Common radical losses: $H^{\cdot}$, $O^{\cdot}$, $^{\cdot}OH$, $^{\cdot}CH_3$, $CH_3O^{\cdot}$, $^{\cdot}C_3H_7$, $^{\cdot}C_4H_9$, $C_6H_5O^{\cdot}$
10^{-3}	All other radical losses

Combinatorics dictates that there exists only a small number of losses below, say, 30 Da. Besides certain common losses, this implies that the number of small losses is also small, but increases rapidly until some maximum is reached. Beyond this mass, we find that the probability to observe a loss drops rapidly in the beginning, but stays significantly above zero even for large masses. To model these observations, we use a log-normal distribution as a classical example of a long-tailed distribution. Let $\mu_{\text{ls}}, \sigma_{\text{ls}}^2$ be the parameters of the log-normal distribution, then the probability density function is

$$\frac{1}{x\sqrt{2\pi\sigma_{\text{ls}}^2}} \exp\left(-\frac{(\ln x - \mu_{\text{ls}})^2}{2\sigma_{\text{ls}}^2}\right)$$

for mass x. See Section 6.6 for the fitting of hyperparameters $\mu_{\text{ls}}, \sigma_{\text{ls}}$; there, we report an excellent fit of loss masses using the log-normal distribution. We use

$$P_{\text{loss-mass}}(\Delta) := \frac{1}{\Delta\sqrt{2\pi\sigma_{\text{ls}}^2}} \exp\left(-\frac{(\ln \Delta - \mu_{\text{ls}})^2}{2\sigma_{\text{ls}}^2}\right) \tag{6.5}$$

for mass delta $\Delta > 0$ as the loss mass prior, and set $P_{\text{loss-mass}}(u, v) := P_{\text{loss-mass}}(\mu(u - v))$.

Some losses turn up more often than we would expect from the loss mass distribution. Instead of relying on an expert-curated list we learn common losses and their prior probabilities from our training data, see Section 6.6; and see Table A.2 in the Appendix for the actual priors $P_{\text{loss-comm}}(u, v)$.

For a FT to be informative, it is useful that the FT includes fragments of small masses, even if the corresponding peaks have small intensities and, possibly as a result, larger mass deviations. In addition, one can relatively easy identify the fragment's correct molecular formula, as well as distinguish fragment peaks from noise, due to the small "combinatorial diversity": The chance that the mass of a noise peak coincidence with the theoretical mass of a molecular formula is very small for small masses. As a theoretical example, consider masses below 15 Da: In this mass region, reasonable molecular formulas are H, H_2, CH and CH_2. To this end, all peaks with other masses *must* be noise. The fragment mass prior favours peaks with small masses,

$$P_{\text{frag-mass}}(u) = \begin{cases} 1 & \text{if } m(u) > 200 \\ e^{2\frac{m(u)}{200}} & \text{otherwise,} \end{cases}$$

to encourage the integration of small peaks that allow for a mass decomposition. The threshold of 200 Da has been chosen ad hoc and without any further optimisation; we expect that choosing, say, a threshold of 100 Da will not result in significant differences.

Combinatorics dictates that the number of explanations for a mass increases with the size of the chemical alphabet. However, biomolecules usually are built from very few elements. If we consider a large chemical alphabet (e.g. the whole periodic table), chances are high that an implausible molecular formula fits better to our data (in the sense of mass deviation) than the real measured molecular formula. We introduced the $P_{\text{root-spec}}$ prior to make molecular formulas with many elements less likely. But we find that for very large trees the root prior has only marginal influence on the weight of the tree. Therefore, we also introduce a $P_{\text{frag-spec}}$ that penalises rare elements for every fragment. As we mentioned above, low mass peaks can be easily identified and there explanations are in general more reliable. In the end we define the $P_{\text{frag-spec}}$ as

$$P_{\text{frag-spec}}(u) = \begin{cases} \frac{5}{3} & m(u) < 100 \\ \frac{1}{3} & \text{otherwise} \end{cases}$$

such that it gives a prior above 1 for small fragments with special elements and a prior below 1 for large fragments with special elements. We observe that common losses and low mass peaks are reliable indicators for the presence of special elements in the compound. We set $P_{\text{loss-spec}} = 1.5$ for all fragments for which the incoming edge (loss) is a common loss containing a special element.

Although, we assume independence of losses and fragments, we already noted that this independence is clearly violated in every fragmentation tree. This is especially the case for rare elements: If the root contains a rare element, also all child fragments have to contain this rare element, or it has to be cleaved off in a loss. With the special element priors we favour rare elements if we either see common losses with rare elements and/or small fragments containing them, while we penalise trees that contain rare losses only in the larger mass region. We reuse the P_{phos} prior we previously have introduced for the root. Phosphorus usually appears in biomolecules as phosphate ester and, therefore, it should cleave of (or stay) together with its neighboured oxygens (or in rare cases: sulphur) atoms.

Finally, we notice that certain fragments turn up repeatedly in FTs. The explanation for this observation is simple and is known to MS experts for decades: Certain groups such as $C_6H_5^+$ (benzyne) or $C_4H_8N^+$ (pyrroline) can be cleaved off as ions, leading to characteristic peaks in the mass spectra. But giving priors for both common losses *and* common fragments, clearly violates the independence assumption: If we know the molecular formulas of a fragment and one of its losses, then this also tells us the molecular formula of the child fragment. To this end, we use a "cautious" prior that rewards only few and small common fragments which are observed very often, whereas the vast majority of fragments receive a flat prior. See Section 6.6 for how we learn the common fragments and their priors from the data; and see Table A.3 in the Appendix for the actual priors $P_{\text{frag-comm}}(u, v)$.

Prior of the Tree Size The FT we will compute should explain a large number of peaks; to this end, we want to favour large trees over small ones. The priors we have introduced so far do exactly the opposite: Many edges result in many probabilities we have to multiply, and small trees are favoured over large trees. To this end, we introduce one last prior: We assume

$$\mathbb{P}(\text{size } |E| \text{ of the tree}) \propto P_{\text{tree-size}}^{|E|} \quad \text{where} \quad P_{\text{tree-size}} := P_{\text{tree-norm}} \cdot P_{\text{tree-bonus}}\,. \tag{6.6}$$

Here, $P_{\text{tree-norm}}$ is chosen to counter the effects of the other priors on average, whereas $P_{\text{tree-bonus}}$ can be set by the user to favour smaller or larger trees. See Section 6.6.3 for how an appropriate default value of this prior is estimated from data.

6.2.2 Likelihood of the Tree

Recall that each considered FT $\mathcal{T} = (V, E)$ is supported by the data $\mathcal{D} = (\mathcal{M}, I)$. This implies the existence of a natural injective mapping $m : V \to \mathcal{M}$: Each node $v \in V$ is assigned a unique peak $m(v)$ in the fragmentation spectrum. All peaks in the spectrum not assigned to a node of the FT, are noise peaks and also contribute to the likelihood of the tree. Also recall that each node $v \in V$ is the molecular formula of the corresponding hypothetical fragment, whereas an edge (u, v) corresponds to a loss $v - u$.

To simplify our computations, we assume independence between the measured peaks in $\mathcal{M} = \{m_1, \ldots, m_L\}$:

$$\mathbb{P}(\mathcal{D}|\mathcal{T}) = \prod_l \mathbb{P}(m_l|\mathcal{T})$$

This simplifying assumption implies that mass deviations and intensities of the individual peaks are independent of each other. Such independence assumptions are commonly used to make a stochastical model computable. Here and in the following, m_l refers both to the l-th peak and to its mass. Furthermore, we may assume that for each peak, the probability of the tree to generate some peak depends only on the corresponding hypothetical fragment, so $\mathbb{P}(m(v)|\mathcal{T}) = \mathbb{P}(m(v)|v)$ for all $v \in V$. Then,

$$\mathbb{P}(\mathcal{D}|\mathcal{T}) = \prod_l \mathbb{P}(m_l|\mathcal{T}) = \prod_{v \in V} \mathbb{P}(m(v)|v) \cdot \mathbb{P}(\text{unassigned peaks}|\mathcal{T})$$

for appropriately chosen $\mathbb{P}(m(v)|v)$. Here, $\mathbb{P}(\text{unassigned peaks}|\mathcal{T})$ is the probability that all unassigned peaks $\mathcal{M} - \{m(v) : v \in V\}$, which cannot be explained by $\mathcal{T}$, are noise peaks.

Unassigned peaks cannot be scored in the FT optimisation, as only those nodes and edges are scored that are actually part of the tree. To get rid of the probability of unassigned peaks, note again that each node is assigned a unique peak, and that no two nodes are assigned the same peak. We reach

$$\mathbb{P}(\mathcal{D}|\mathcal{T}) = \mathbb{P}(\text{all peaks in } \mathcal{D} \text{ are noise}) \cdot \prod_{v \in V} \frac{\mathbb{P}(m(v)|v)}{\mathbb{P}(m(v) \text{ is noise})}$$

for appropriate $\mathbb{P}(m(v)|v)$. Again, for fixed data $\mathcal{D}$, the probability of all peaks being noise simultaneously is a constant, and can be ignored in the optimisation of $\mathbb{P}(\mathcal{T}|\mathcal{D})$.

We will now show how to compute the probability of signal peaks and noise peaks. Currently, there exists no general model for the intensity of signal peak in small compound MS. Here, the problem is even harder, as we do not know the fragment's molecular *structure* but only its molecular formula. Similarly, there exists no sensible model for the mass of noise peaks. To this end, we will use only the peak mass to assess the probability of signal peaks; and only peak intensity to assess the probability of noise peaks. The intensity of peak m is $I(m)$; for brevity we write $I(v) := I(m(v))$.

Probability of Signal Peaks It has been frequently observed that relative mass deviations are roughly normally-distributed [95, 245]. We found this to be the case for our datasets, see Section 6.6. We assume that the instrument is decently calibrated, so that no mass bias can be observed. Let MA be the mass accuracy parameter used to build the fragmentation graph. If we assume that 95.5 % of the normally-distributed masses fall within this range, then the standard deviation is $\sigma_{\text{ppm}} := \frac{1}{2}MA$; if we assume that 99.7 % of the masses fall within this range, then $\sigma_{\text{ppm}} := \frac{1}{3}MA$. Now, relative mass errors are distributed according to $\mathcal{N}(0, \sigma_{\text{ppm}})$. We ignore the fact that no mass errors above some threshold can be observed (truncated normal distribution) as this has a negligible effect on our computations. The probability to observe a peak with mass $m(v)$ for node/fragment v can be estimated as:

$$\begin{aligned}\mathbb{P}(m(v)|v) &= \mathbb{P}\left(|\mathcal{N}(0,\sigma_{\text{ppm}})| \geq \tfrac{|m(v)-\mu(v)|}{\mu(v)}\right) \\ &= 2 \cdot \int_{\frac{|m(v)-\mu(v)|}{\mu(v)}}^{\infty} \frac{1}{\sigma_{\text{ppm}}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x}{\sigma_{\text{ppm}}}\right)^2} dx = \operatorname{erf}\left(\tfrac{|m(v)-\mu(v)|}{\sigma_{\text{ppm}}\sqrt{2}\mu(v)}\right)\end{aligned} \tag{6.7}$$

This is the two-sided probability that a mass deviation larger than the observed relative mass deviation of peak $m(v)$ will occur by chance. Here, "erf" denotes the error function.

Probability of Noise Peaks As we have no model for the intensity of fragment peaks, $I(v)$ cannot be used for estimating the probability of fragment peaks. Similarly, we have no model for noise peak masses. But we can estimate the probability that a certain peak is noise, by observing that noise with high intensity are much rarer than noise peaks with small intensity.

Böcker and Rasche [25] proposed to directly use the peak intensity in the score calculation. Later, Rasche *et al.* [164] pointed out that this can be statistically justified by assuming that noise peak intensities are exponentially distributed. To this end, we analyse the intensity distribution of noise peaks, see Section 6.6. We observe that with increasing intensity, the probability to observe a noise peak of this intensity drops rapidly in the beginning, but stays significantly above zero even for large intensities. This is an example of a long-tailed distribution, and we use the Pareto distribution as a classical example of a long-tailed distribution. This distribution offers the additional advantage that a minimum peak intensity threshold, which is naturally applied in peak picking, can be directly integrated into the model.

Let I_{min} be the peak intensity threshold used for peak picking. Then, the probability for observing a noise peak with intensity larger or equal to I is $(\frac{I_{\text{min}}}{I})^{\alpha_{\text{i}}}$. See Section 6.6 for fitting hyperparameters α_{i} and I_{min}.

We want our probabilistic model to generalise well across different experimental setups and instruments. However, the absolute intensity values depend on the instrument type as well as the preprocessing software. We decide to use *normalised peak intensities* instead. We divide the intensity of each peak by the intensity of the most intensive peak in the spectrum, such that all peaks have an intensity between 0 and 1. This means that we will never record a peak with intensity above 1 and, therefore, we have to cut off the probability distribution at 1. Nevertheless, the probability distribution is not allowed to drop too close to zero, as even a peak with maximal intensity might still be a noise peak. We define $\beta_{\text{i}} = 10^{-5}$ as the probability that the most intensive peak in the spectrum is

noise. Then we can define the probability of observing a noise peak m with intensity $I(m)$ or higher, as

$$\mathbb{P}(m \text{ is noise}) = 1 - \frac{\left(\frac{I_{\min}}{I(m)}\right)^{\alpha_{\mathrm{i}}} - I_{\min}^{\alpha_{\mathrm{i}}} + \beta_{\mathrm{i}}}{1 - I_{\min}^{\alpha_{\mathrm{i}}} + \beta_{\mathrm{i}}}\,. \tag{6.8}$$

We found that the Pareto distribution agrees well with the experimental data, see Section 6.6.

6.2.3 Posterior Probability of the Tree

From the above we infer that

$$\mathbb{P}(T) \cdot \mathbb{P}(\mathcal{T}|\mathcal{D}) \propto P_{\text{root}}(r) \cdot \prod_{e \in E} (P_{\text{edge}}(e) \cdot P_{\text{tree-size}}) \cdot \prod_{v \in V} \left(\frac{\mathbb{P}(m(v)|v)}{\mathbb{P}(m(v) \text{ is noise})} \right) \tag{6.9}$$

for FT $\mathcal{T} = (V, E)$ with root $r \in V$. The probability that all peaks in the spectrum are noise, is independent of the actual tree $\mathcal{T}$ and, hence, can be disregarded. We define

$$\begin{aligned} \mathcal{L}(\mathcal{T}) := \log P_{\text{root}}(r) + \sum_{e \in E} \log(P_{\text{edge}}(e) \cdot P_{\text{tree-size}}) + \\ \sum_{v \in V} \left(\log \operatorname{erf}\left(\tfrac{|m(v)-\mu(v)|}{\sigma_{\text{ppm}}\sqrt{2}\mu(v)}\right) - \log\left(1 - \frac{\left(\frac{I_{\min}}{I(v)}\right)^{\alpha_{\mathrm{i}}} - I_{\min}^{\alpha_{\mathrm{i}}} + \beta_{\mathrm{i}})}{1 - I_{\min}^{\alpha_{\mathrm{i}}} + \beta_{\mathrm{i}}}\right)\right) \end{aligned} \tag{6.10}$$

then $\log(\mathbb{P}(T) \cdot \mathbb{P}(\mathcal{T}|\mathcal{D})) = \mathcal{L}(\mathcal{T}) + c$ for some constant $c \in \mathbb{R}$. To this end, the posterior probability of tree $\mathcal{T}$ is maximum if and only if $\mathcal{L}(\mathcal{T})$ is maximum.

Given a fragmentation spectrum $\mathcal{D}$ we proceed as follows: First, for each peak $m \in \mathcal{D}$ we search for all molecular formulas v that are within the specified mass accuracy MA,

$$\mu(v) \in \big[m - \delta, m + \delta\big]$$

with $\delta = MA \cdot \max\{m, m_{MA}\}$. In case two of these intervals overlap, we shrink them accordingly. We use these molecular formulas v as the nodes of the fragmentation graph, each coloured with the corresponding mass m, and set $m(v) = m$. We introduce an edge (u, v) for each pair $u \geq v$. For each edge (u, v) we set its edge weight to

$$\begin{aligned} w(u,v) := \log P_{\text{edge}}(u,v) + \log P_{\text{tree-size}} + \log \operatorname{erf}\left(\tfrac{|m(v)-\mu(v)|}{\sigma_{\text{ppm}}\sqrt{2}\mu(v)}\right) - \\ \log\left(1 - \frac{\left(\frac{I_{\min}}{I(v)}\right)^{\alpha_{\mathrm{i}}} - I_{\min}^{\alpha_{\mathrm{i}}} + \beta_{\mathrm{i}}}{1 - I_{\min}^{\alpha_{\mathrm{i}}} + \beta_{\mathrm{i}}}\right)\,. \end{aligned} \tag{6.11}$$

We also introduce a super-root sr which is connected to all nodes corresponding to the parent mass M. These $v \in V$ with $m(v) = M$ are the potential roots of the FT, and for each we set

$$w(sr, v) := \log P_{\text{root}}(v) + \log \operatorname{erf}\left(\tfrac{|m(v)-\mu(v)|}{\sigma_{\text{ppm}}\sqrt{2}\mu(v)}\right)\,. \tag{6.12}$$

With these edge weights, ordering colourful subtrees with respect to their weight, is equivalent to ordering the corresponding FTs by posterior probability.

6.3 Hypothesis-driven Recalibration

To improve the quality of FTs, and to increase the chance that the FT with the correct molecular formula root will receive the highest score, we use a hypothesis-driven recalibration [24]. We are given one fragmentation spectrum at a time. For each candidate molecular formula explaining the root, we compute a FT, and then use the theoretical masses of all nodes in the FT as references to recalibrate the sample spectrum. Some of the molecular formulas assigned to peaks may be wrong, even for the correct candidate molecular formula. To this end, we use recalibration methods which are robust to outliers, and automatically discard such wrong assignments when computing the recalibration.

Recalibration is carried out using an affine mass correction [24] $f(x) := ax+b$. Let (x_i, y_i) be the pairs of potentially matching masses: x_i is a mass in the measured spectrum, and y_i is a mass in the reference spectrum simulated using the FT. Note that for any measured (reference) mass there can be multiple elements with different reference (measured) masses. We use the Theil-Sen estimator [192, 210] to find the slope a of f as the median of the slopes $(y_j - y_i)/(x_j - x_i)$ determined by all pairs of sample points with distinct x-coordinates. Next, we set b to be the median of the values $y_i - mx_i$. We recalibrate the measured spectrum by applying f to all masses.

We then compute the optimal FT for the recalibrated sample spectrum and the candidate molecular formula, and use this score to evaluate which root molecular formula best explains the data. Then, the recalibration is *discarded*, returning to the original measured sample spectrum, and the next root molecular formula is processed.

We observed that peaks in the low mass region often have a very different mass deviation than peaks in the high mass region. This indicates that the underlying bias cannot be explained by a linear function. However, more complex functions like quadratic or high order polynomial functions have too many degrees of freedom such that also wrong FT explanations result in recalibrated spectra with low mass errors. A problem occurs when the number of peaks in low mass region is much higher than in high mass region as the median is then mostly influenced by low mass peaks. For this reason we propose the following approach: For each 100 Dalton interval in the mass spectrum that contains at least one peak we count the number of nodes in the tree with this mass. We then take the median k over all counts. We build the reference spectrum by choosing the k-most intensive explained peaks in each interval and insert their exact mass (known from the hypothetical tree) in the reference spectrum.

We note that our hypothesis-driven recalibration is fundamentally different from, say, the recalibration proposed in [206]: In our approach, each spectrum is recalibrated individually, using each peak's best theoretical explanation as anchors for the mass correction. In this way, we do not require a homogeneous dataset of mass spectra to start the recalibration process.

We do not apply the recalibration to the precursor ion, such that we will never change the set of possible molecular formula explanations of the precursor ion by recalibration. Fragment ions may, in some cases, change their molecular formula annotation after recalibration. In the most simple case, recalibration only effects the scoring of the fragment peaks. Note that an alternative to hypothesis-driven recalibration is a scoring that favours peak explanations that have a mass deviation in the same direction. However, in such a scoring the likelihood of a peak would depend on other peaks, which violates our independence assumption. In especially, we could not longer solve such a problem with an

integer linear program. From this perspective it seems that hypotheses driven recalibration is just a workaround - but there are some advantages in this approach: If the mass bias is very large, some peaks might be annotated wrongly or cannot be annotated at all because they are outside of the allowed mass deviation window. Hypotheses driven recalibration may correct these annotations as long as there are enough correctly explained peaks in the spectrum.

6.4 Isotope Patterns

Isotope patterns in MS and MS^2 can be naturally integrated into the maximum a posteriori estimation. We will first introduce a scoring of isotope peaks based on maximum likelihood. We then show how to integrate this scoring into the maximum a posteriori probability estimate. In some experimental setups, for example MS^{all} or SWATH, we have to expect isotope peaks within the MS^2 spectrum. We will show how we can extend the FT model such that it includes isotopic peaks. This new model can still be represented as colourful subtree in an acyclic colourful graph - such that we do not have to change anything on computational side. Finally, we will present a detector for uncommon elements based on the shape of isotope pattern [133].

While isotope pattern analysis of the precursor ion in MS is an integral part of SIRIUS since its first release [28], the support for element detection, isotope patterns of in-source fragments as well as isotope patterns within MS^2 was developed throughout this thesis and released with SIRIUS 4.

6.4.1 Maximum Likelihood Scoring for Isotope Patterns

To reveal the molecular formula of an ion we first enumerate over all possible molecular formulas, simulate an isotope pattern for each molecular formula and then compare the simulated pattern with the measured one. Let $m = m_1, \ldots, m_n$ be the measured masses and $p = p_1, \ldots, p_n$ the measured intensities, $\hat{m} = \hat{m}_1, \ldots, \hat{m}_n$ the simulated masses and $\hat{p} = \hat{p}_1, \ldots, \hat{p}_n$ the simulated intensities. We assume that intensities are relative values such that their maximum is 1. Zhang *et al.* [244] suggested to use Bayesian statistics to score the similarity of simulated and measured peaks. If we assume independence between the peaks we can compute the similarity as

$$\mathbb{P}(\hat{m}, \hat{p} \mid m, p) = \prod_{i=1}^{n} \mathbb{P}(m_i \mid \hat{m}_i) \cdot \mathbb{P}(p_i \mid \hat{p}_i)\,. \tag{6.13}$$

It is obvious that the independence assumption is clearly violated as all peak masses share a similar recalibration error and peak intensities are relative to the maximum intensity. But nevertheless this gives us a reasonable scoring function. We can get rid of correlated mass deviations by looking at mass differences (see eq. (6.15)). Again we assume a normal distributed error on measured masses. Let $\epsilon_i = m_i - \hat{m}_i$ be the mass difference between a measured and simulated peak. When mass errors are normally distributed $\epsilon \sim \mathcal{N}(0, \sigma)$ with sigma is the average mass error, we can compute the probability

$$\mathbb{P}(\hat{m}_i \mid m_i) = \operatorname{erfc}\left(\tfrac{m_i - \hat{m}_i}{\sqrt{2}\pi\sigma}\right).$$

Mass errors are larger for larger masses. As done in Section 6.2.2 we use a relative measure $\sigma = m_1 \cdot 10^{-6} \cdot \sigma_{\text{ppm}}$.

So far, we ignored that measurements usually come with some systematic shift (a recalibration error). This error will be roughly the same for all isotope peaks and, due to the independence assumption, will affect the probability for every peak. We can remove the recalibration error by subtracting m_i from m_1. The average error for mass differences is a user specified parameter σ_{diff}. We observe that low intensive peaks have larger mass errors than high intensive peaks. We, therefore, define a intensity dependent mass error as a piecewise linear function:

$$\sigma_{\text{diff}}(p) := \begin{cases} 10^{-6} \cdot m_1 \cdot 1 \cdot \sigma_{\text{diff}} & \text{if } p > 0.2 \\ 10^{-6} \cdot m_1 \cdot \left(1 + \frac{0.2-p}{0.1}\right) \cdot \sigma_{\text{diff}} & \text{if } 0.1 < p \leq 0.2 \\ 10^{-6} \cdot m_1 \cdot \left(2 + \frac{0.1-p}{0.09}\right) \cdot \sigma_{\text{diff}} & \text{if } 0.01 < p \leq 0.1 \\ 10^{-6} \cdot m_1 \cdot 3 \cdot \sigma_{\text{diff}} & \text{otherwise} \end{cases} \tag{6.14}$$

We then come up with a more accurate estimation of the probability by replacing eq. (6.13) by

$$\mathbb{P}(\hat{m}, \hat{p} \mid m, p) = \operatorname{erfc}\left(\frac{m_1 - \hat{m}_1}{\sqrt{2}\pi\sigma_{\text{ppm}}}\right) \cdot \prod_{i=2}^{n} \operatorname{erfc}\left(\frac{(m_i - m_1) - (\hat{m}_i - \hat{m}_1)}{\sqrt{2}\pi\sigma_{\text{diff}}(p_i)}\right) \cdot \prod_{i=1}^{n} \mathbb{P}(\hat{p} \mid p)\,. \tag{6.15}$$

Böcker et al. suggest using a log-normal distribution for relative intensity errors [26]. The probability $\mathbb{P}(p_i \mid \hat{p}_i)$ is then calculated via the complementary error function of logarithmised intensity ratios. The underlying assumption is that intensity errors are relative. However, we know that low intensive peaks are measured less accurate, especially if they are near noise level. A relative error underestimates the error on low intensive peaks. Kind et al. and Pluskal et al. suggest to sum up the intensity error between a simulated and a theoretical isotope pattern and use this error directly as scoring function [107, 158]. This approach can be adopted for our stochastical model using exponential distributed absolute noise. However, we can observe that intensity errors are larger for high intensive peaks (as we will see in Section 6.6.2).

To simultaneously model absolute and relative intensity deviations, we propose a simple maximum likelihood estimator that requires only two parameters, namely, absolute error $\sigma_{\text{abs}} > 0$ and relative error $\sigma_{\text{rel}} > 0$ of peak intensities. We statistically model the intensity error as

$$Y = x + D + E\,,$$

where x is the expected (theoretical) intensity, Y is the random variable modelling the observed intensity, and D, E are random variables for relative and absolute noise, respectively. We assume that both relative noise $D \sim x \cdot \mathcal{N}(0, \sigma_{\text{rel}}^2) = \mathcal{N}(0, x^2\sigma_{\text{rel}}^2)$ and absolute noise $E \sim \mathcal{N}(0, \sigma_{\text{abs}}^2)$ are normally distributed: In detail, we assume that D, E have densities

$$f_D(\delta) = \frac{1}{\sqrt{2\pi x^2 \sigma_{\text{rel}}^2}} \exp\left(-\frac{\epsilon^2}{2x^2\sigma_{\text{rel}}^2}\right) \quad \text{and} \quad f_E(\epsilon) = \frac{1}{\sqrt{2\pi\sigma_{\text{abs}}^2}} \exp\left(-\frac{\epsilon^2}{2\sigma_{\text{abs}}^2}\right)$$

for $\delta, \epsilon \in \mathbb{R}$. We are using the probability density function to estimate these probabilities, which can be interpreted as the limit of an arbitrary small interval around the values δ, ϵ, respectively. Note that this model can result in negative observed peak intensities; we found that this limitation is not relevant in application, where relatively weak noise is observed.

We further assume that relative and absolute noise are independent. Given an observed intensity y and an expected intensity x, the likelihood of some model $\theta = (\delta, \epsilon)$ is

$$\mathcal{L}_{y|x}(\delta,\epsilon) = \tfrac{1}{\sqrt{2\pi}x\sigma_{\text{rel}}} \exp\left(\tfrac{-\delta^2}{2x^2\sigma_{\text{rel}}^2}\right) \cdot \tfrac{1}{\sqrt{2\pi}\sigma_{\text{abs}}} \exp\left(\tfrac{-\epsilon^2}{2\sigma_{\text{abs}}^2}\right) = \tfrac{1}{2\pi x\sigma_{\text{rel}}\sigma_{\text{abs}}} \exp\left(-\tfrac{\delta^2}{2x^2\sigma_{\text{rel}}^2} - \tfrac{\epsilon^2}{2\sigma_{\text{abs}}^2}\right). \tag{6.16}$$

We now want to find the model that best explains the observed data, in the sense that we maximise likelihood. We substitute $\epsilon = y - x - \delta$ and reach

$$\mathcal{L}_{y|x}(\delta,\epsilon) = \frac{1}{2\pi x\sigma_{\text{rel}}\sigma_{\text{abs}}} \exp\left(-\frac{\delta^2}{2x^2\sigma_{\text{rel}}^2} - \frac{(y-x-\delta)^2}{2\sigma_{\text{abs}}^2}\right)$$

which does no longer depend on ϵ. Due to the monotonicity of the exponential function, this likelihood is maximum if and only if

$$g(\delta) := -\frac{\delta^2}{2x^2\sigma_{\text{rel}}^2} - \frac{(y-x-\delta)^2}{2\sigma_{\text{abs}}^2}$$

is maximum. Differentiating g we reach

$$\begin{aligned} g'(\delta) &= -2\frac{1}{2x^2\sigma_{\text{rel}}^2}\delta + \frac{1}{2\sigma_{\text{abs}}^2}2(y-x-\delta) = -\frac{\delta}{x^2\sigma_{\text{rel}}^2} + \frac{y-x-\delta}{\sigma_{\text{abs}}^2} \\ &= \frac{1}{\sigma_{\text{abs}}^2 x^2\sigma_{\text{rel}}^2}\left((y-x)x^2\sigma_{\text{rel}}^2 - (\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2)\delta\right). \end{aligned}$$

Now, $g'(\delta_0) = 0$ if and only if

$$(y-x)x^2\sigma_{\text{rel}}^2 - (\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2)\delta_0 = 0\,,$$

so the maximum is reached for

$$\delta_0 := \frac{(y-x)x^2\sigma_{\text{rel}}^2}{\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2}\,.$$

This allows us to compute the Maximum Likelihood as $\mathcal{L}_{y|x}(\delta_0, \epsilon_0)$ using (6.16), where $\epsilon_0 = y - x - \delta_0$: Note that

$$-\frac{\delta_0^2}{2x^2\sigma_{\text{rel}}^2} = -\frac{(y-x)^2 - x^2\sigma_{\text{rel}}^2}{2(\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2)^2}$$

and

$$y - x - \delta_0 = (y-x)\left(1 - \frac{x^2\sigma_{\text{rel}}^2}{\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2}\right) = (y-x)\frac{\sigma_{\text{abs}}^2}{\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2}\,.$$

We reach:

$$\begin{aligned} \mathcal{L}_{y|x}(\delta_0,\epsilon_0) &= \frac{1}{2\pi x\sigma_{\text{rel}}\sigma_{\text{abs}}} \exp\left(-\frac{(y-x)^2x^2\sigma_{\text{rel}}^2}{2(\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2)^2} - \frac{(y-x)^2\sigma_{\text{abs}}^2}{2(\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2)^2}\right) \\ &= \frac{1}{2\pi x\sigma_{\text{rel}}\sigma_{\text{abs}}} \exp\left(-\frac{(y-x)^2}{2(\sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2)}\right). \end{aligned} \tag{6.17}$$

But this implies that for the computation of the maximum likelihood, we actually do not have to compute δ_0 or ϵ_0; instead, it is sufficient to directly insert theoretical intensity x

and observed intensity y into equation (6.17) to estimate the maximum likelihood of the data. We note that (6.17) is very similar to the probability density function of the random variable $D + E \sim \mathcal{N}(0, \sigma_{\text{abs}}^2 + x^2\sigma_{\text{rel}}^2)$.

Finally, the likelihood for a measured isotope pattern to be explained by the given simulated pattern is

$$\mathbb{P}(\hat{m}, \hat{p} \mid m, p) = \operatorname{erfc}\left(\frac{m_1 - \hat{m}_1}{\sqrt{2}\pi\sigma_{\text{ppm}}}\right) \cdot \prod_{i=2}^{n} \operatorname{erfc}\left(\frac{(m_i - m_1) - (\hat{m}_i - \hat{m}_1)}{\sqrt{2}\pi\sigma_{\text{diff}}(p_i)}\right) \cdot \prod_{i=1}^{n} \frac{1}{2\pi\hat{p}_i\sigma_{\text{rel}}\sigma_{\text{abs}}} \exp\left(-\frac{(p_i - \hat{p}_i)^2}{2(\sigma_{\text{abs}}^2 + \hat{p}_i^2\sigma_{\text{rel}}^2)}\right). \tag{6.18}$$

For a candidate molecular formula, we simulate an isotope pattern with peak intensities and mean peak masses as described in [26, 28]. We normalise both spectra using the first isotope peak. Assuming statistical independence [26, 28], the likelihood of the candidate molecular formula is simply the product of the individual likelihoods for peak mass differences and peak intensity differences as shown in eq. (6.18). Peaks which are not observed but expected by the theoretical pattern get no likelihoods for mass deviations but still a likelihood for intensity deviation where they are counted as zero.

Isotope pattern are extracted from MS1 by searching for the most intensive peak around the precursor mass within the allowed mass deviation and then extending the pattern gradually by picking the next isotope peak that has a reasonable mass difference. If several such peaks exist, they are merged using the weighted mean of their masses and the sum of their intensities. To prevent that we include a peak into the pattern which is not part of the isotope pattern, but, for example, another coeluting ion, we compute scores for all possible lengths of the pattern (by successively removing the last peak of the pattern) and only report the maximum score. Note that such an approach is only possible, if the isotope pattern score is positive for reasonable isotope pattern and gets negative for unreasonable ones. In our case, the isotope pattern score, although a log likelihood, is positive for reasonable patterns because we use the densities of two normal distributions to estimate the likelihood. As long as the standard deviations σ_{abs} and σ_{rel} are very small (which is the case for any reasonable isotope scoring), the density becomes a value above 1 at maximum. Alternatively, we could estimate a normalisation factor to make our score positive.

6.4.2 Integrating Isotope Patterns into Fragmentation Tree Computation

In most experimental setups, the precursor ion is isolated and fragmented without its isotope peaks. Even if the isolation window is slightly larger than one Dalton, we will rarely observe isotope peaks in MS^2. This is because ions close to the border of the isolation window are collected with much lower efficiency. In this case we only have to deal with isotope peaks in MS. We first look for peaks in MS which have an isotope pattern and also occur in MS^2. These peaks are likely candidates for in-source fragments. We score their isotope pattern using eq. (6.18) and add this log likelihood to log likelihood of the node in the fragmentation graph with same molecular formula explanation. We do only add positive log likelihoods. We do the same for the precursor ion. Note that we do not have to score the likelihood of the root node when we have an isotope pattern available. In most cases, mass accuracy in MS is better than in MS^2. Therefore, we use the mass of

the MS ion for generating the list of candidate formulas and use the isotope pattern in MS to score the likelihood of the root node.

In cases where we have to expect isotope patterns in MS^2 we cannot longer simply add the log likelihood to the nodes: We do not know which peak in the spectrum is an isotope peak and which peak is a fragment ion. A simple heuristic would be to first score all isotope patterns and then remove all isotope peaks which belong to a pattern with positive score. However, a more elaborated approach would be to formulate this decision within the maximum colourful subtree problem. We add a special kind of nodes in our tree model, which we call isotope nodes. An isotope node explains a peak in the spectrum, like a fragment node. A tree is supported by the data if all its fragment nodes and all its isotope nodes correspond to a peak in the spectrum. Each isotope node v is either the child of a fragment node u in which case the mass of v is the mass of the second isotope peak of the theoretical isotope pattern of u. Otherwise, it must be a child of another isotope node, in which case its mass is the mass of the successive isotope peak of its parent. Note that we can add isotope nodes to any existing fragmentation tree and use them to simulate how the spectrum would look like when measured in MS^{all}. To obtain the likelihood of the tree we just have to simulate the corresponding mass spectrum and compare the simulated and measured mass spectrum using the approach in Section 6.4.1. It still makes sense for each monoisotopic peak to divide its likelihood by the probability that this peak is noise using eq. (6.8).

This extended statistical model can be easily integrated into the maximum colourful subtree problem: For each node u in the fragmentation graph we check if there is an isotope pattern for the corresponding peak in the spectrum. Let $p_1, \dots p_n$ be the isotope pattern with p_1 is the colour of the node u. Let $\hat{p}_1, \dots, \hat{p}_n$ be the simulated isotope pattern using the molecular formula of u. We now add nodes $v_2 \dots v_n$ and colour them by the corresponding isotope peaks $p_2 \dots p_n$. We then add edges between consecutive nodes as well as an edge uv_2. We weight uv_2 with $\log \mathbb{P}(\hat{p}_1, \hat{p}_2 \mid p_1, p_2)$. We score the edges $v_i v_{i+1}$ between the isotope peaks with

$$\log \frac{\mathbb{P}(\hat{p}_1, \dots, \hat{p}_{i+1} \mid p_1, \dots, p_{i+1})}{\mathbb{P}(\hat{p}_1, \dots, \hat{p}_i \mid p_1, \dots, p_i)}.$$

We have shown how to integrate isotope patterns into the FT computation without changing the underlying algorithmic problem. However, we changed a small but possibly important property of the fragmentation graph: Without isotope patterns, for each directed path in the graph the masses of its nodes are strictly decreasing. For edges between isotope nodes, however, the mass is increasing. While this is not changing the problem in general, reduction rules or heuristics that are developed for solving the problem specific for fragmentation graphs might rely on this property. For example, the critical path heuristic [59] does not work when adding isotope nodes.

6.4.3 A Deep Neural Network for Estimating Elemental Composition

Isotope patterns of small molecules that consist solely of carbon, hydrogen and oxygen have a monotonic decreasing isotope pattern. When elements like sulphur, chlorine, bromine or selene are present, the shape changes, sometimes into a zig-zag pattern. While our maximum likelihood scoring is very efficient in finding the correct molecular formula explanation, its performance depends exponentially on the number of elements in the chemical alphabet. Although, most biomolecules consist of CHNOPS, there are also

some biomolecules as well as pesticides and pharmaceutics which contain halogens ClBrIF. Allowing the whole alphabet CHNOPSClBrIF is not possible for larger masses; in fact we would not even be able to keep all molecular formula explanations in memory. Therefore, we need some preprocessing step that filter out unreasonable explanations. It was shown that machine learning methods (namely, random forests) are able to exclude elements from the chemical alphabet based on the shape of the isotope pattern [133]. We do not use Random Forests to predict the presence/absence of individual elements, as this would require long startup times of SIRIUS for loading the decision trees from hard disk. Instead, we train Deep Neural Networks [116] for this task. We predict the presence/absence for elements sulphur,chlorine, bromine, boron, and selene. The DNNs have three hidden layers with 48, 32, and 32 neurons using the hyperbolic tangent activation function; the input layer is using the same features as the random forests from Meusel *et al.* [133], see Table A.4 and Table A.5 in the Appendix. All features are centred and normalised. We also use predictors for upper bounds on the number of atoms of a particular element in the query compound ("there are at most two sulphur atoms present"), which further speeds up computations. As loss function, we choose the mean squared error on the maximum of the logarithmised quantity and 0.2, for each element. The logarithm results in small costs for overestimating the amount of an element, but large costs for predicting the presence of an absent element (or vice versa).

As in [133], we do not use measured MS^2 spectra for training, but simulate theoretical isotope pattern of one million molecular formulas from PubChem. To prevent overfitting, we use l_2 regularisation and add normal distributed random noise on the training data [133]. Performance of the network is comparable to that of the Random Forests reported in Meusel *et al.* [133], but model parameters consume only 75 KB of memory, compared to more than 200 MB for the Random Forests [133].

6.5 Identify Molecular Formulas with SIRIUS: the Complete Workflow

After plenty of details about the scoring and integration of isotopes, we will now shortly summarise the workflow of SIRIUS.

The input of SIRIUS is an MS^2 spectrum and, optionally, an isotope pattern in MS. SIRIUS aims to identify the molecular formula of the analyte and annotate the spectrum with a fragmentation tree. The tree assigns molecular formulas to each peak in the MS^2 spectrum. The computation is done as follows:

1. If an isotope pattern is given, use the deep neural network to restrict the set of elements that are allowed for the analysis.

2. Enumerate over all molecular formulas that explain the precursor ion peak, or (if given) the monoisotopic peak of the isotope pattern using the Round Robin algorithm [23, 27, 55].

3. If an isotope pattern is given, score each of these molecular formulas using the maximum likelihood scoring in Section 6.4.1. If at least one pattern gets reasonable score, reject all molecular formula with very low score.

4. Now we perform the MS^2 analysis. For each molecular formula explanation of the precursor ion we create a fragmentation graph.

a) We add the molecular formula as root into the graph and weight it with the root prior in eq. (6.3) and, if an isotope pattern is given, by the maximum likelihood score from the isotope pattern analysis.

b) We now decompose the sixty most intensive peaks in the spectrum; this means we enumerate over all molecular formula explanations. For each molecular formula we add a node into the fragmentation graph using the molecular formula as label and the peak as colour.

c) For each two fragments u and v with $u - v > 0$ we add an edge into the graph and label it with the loss $u - v$.

d) Optionally: We search for isotope patterns in MS^2 and add them to the graph as described in Section 6.4.2. We also search for in-source fragments in MS and add their isotope score to the likelihood of the corresponding node.

e) We weight the graph with posterior probabilities as described in Section 6.2.3.

f) We compute the best fragmentation tree using the heuristic algorithm in [59].

5. We sort the heuristic fragmentation trees according to their scores.

6. For the top k fragmentation trees:

   a) We use hypothesis driven recalibration as described in Section 6.3 to recalibrate the spectrum.

   b) We repeat steps from 4a) to e) using the recalibrated spectrum.

   c) We compute the optimum fragmentation tree, this time using an exact algorithm (the integer linear program as described in eq. (4.1)).

7. We sort the k fragmentation trees according to their scores. The top scoring trees are reported to the user.

6.6 Statistics and Fitting the Model

We now describe how to estimate the hyperparameters and parameters for priors and likelihood. The parameter estimation was done using 2 005 spectra from GNPS and 2 046 spectra from the Agilent dataset [22]. For isotope pattern related parameters, we had to choose parameters ad hoc as we only had very few spectra with isotope patterns available at this time. Nowadays, we have the Agilent 2.0 dataset with 3 502 isotope pattern. However, instead of estimating new parameters for isotope pattern analysis, we will show in Section 6.6.2 that the ad hoc parameter show a good performance on Agilent 2.0 data.

Although the number of available training spectra has increased by several times, we did not repeat the (hyper)parameter estimation. We find that the (hyper)parameters are very robust and effective for any other dataset measured with a high mass accuracy instrument and electrospray ionisation. The (hyper)parameters estimated on positive ion mode data can even be applied on negative ion mode data. This is surprising, as negative ion mode spectra look quite different and usually require a separate training (as in Section 7.9.4). Note that parameter fitting is very different from training the weights of support vector machines or neural networks: First, we do not optimise the parameters for maximising the score difference between correct and incorrect answers. In fact, we only define a generative model that yields high probabilities for correct trees. Second, the degree of freedom in our

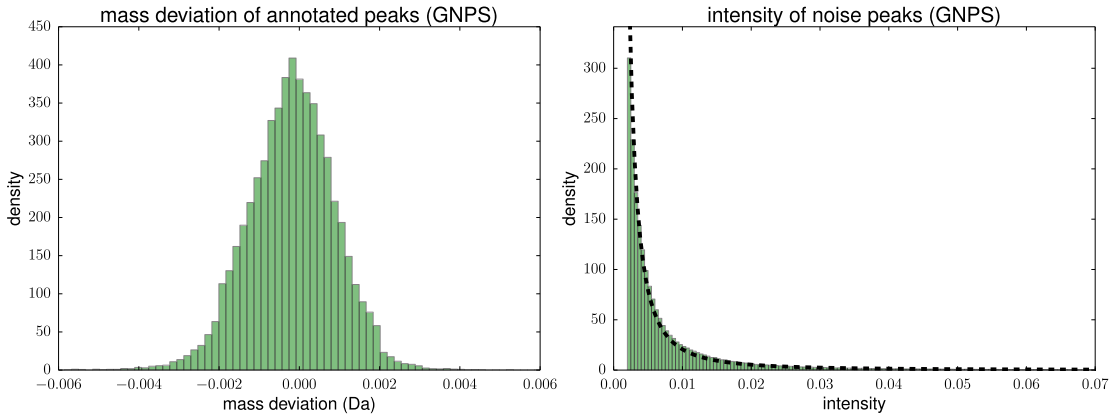


Figure 6.4: Left: Normalised histogram of the mass error distribution, for the GNPS dataset. Right: Normalised histogram of the noise peak intensity distribution and fitted Pareto distribution (dashed line), for the GNPS dataset.

model is very low, as we only optimise a comparatively small number of parameters. This is the reason why this model can be applied across many instruments and experimental settings. But this also means that some parameter choices that are estimated from data are worse for ranking the correct FT on top position than manually chosen parameters. We will, therefore, adjust some of the parameters, especially for likelihoods of peaks and isotopes, when this results in a better ranking performance.

6.6.1 Mass Error and Noise Peak Intensity.

The mass accuracy parameters σ_{ppm}, MA, m_{MA} are one of the few parameters that seem to be highly dependent on the chosen instrument. Fourier transform instruments like Orbitrap and FTICR have a better mass accuracy than Q-ToF instruments. While it is possible to adjust these parameters individually for each dataset, SIRIUS can achieve good performance even if parameters deviate significantly from the experimental truth. We find that $MA = 10$ ppm and $m_{MA} = 250\,\text{Da}$ gives good results on Q-ToF instruments. These are also the parameters we used in [22] (note that Agilent spectra as well as most GNPS spectra are measured on a Q-ToF instrument). For σ_{ppm}, we find that we get better results when using a very large standard deviation of $\sigma_{\text{ppm}} = 10$. This is somewhat counter intuitive, as we set the average mass error to the maximal allowed mass error. However, using a very large σ_{ppm} is somewhat comparable to "downweight" the mass deviation score. It seems that mass deviation is just not as informative as priors like the common loss prior. For Orbitrap and FTICR instruments we choose $MA = 5$ ppm and $\sigma_{\text{ppm}} = 5$ instead.

However, is our assumption even correct that mass errors are normally distributed? To verify this claim, we have to know the true (theoretical) mass of the fragments that resulted in some peak in the spectrum. To estimate the true mass, we use FTs computed for the correct root molecular formula, after setting all hyperparameters as described below. We compare the theoretical mass of each FT node with the observed mass of the peak. For both datasets, we observe that mass errors roughly follow a normal distribution, see Fig. 6.4. We also see that the distribution of mass errors has a mean different from zero. This suggests, that it might be possible to recalibrate spectra across a whole dataset.

In [22] we determined the noise intensity parameters individually for each of the two datasets, because spectra in the Agilent dataset only provide relative intensities, whereas GNPS spectra provide absolute intensities.

The parameter $I_{\min}$ can be easily estimated by taking the minimum intensity over all spectra. However, we can safely increase this parameter. For performance reason, SIRIUS cannot process hundreds (or even thousands) of peaks in a spectrum, anyways. We used the peak intensity threshold $I_{\min} = 0.002$ for GNPS and $I_{\min} = 0.005$ for Agilent as the first parameter of the noise intensity Pareto distribution. Peaks with lower intensity are removed from the spectrum. We find that $I_{\min} = 0.002$ is a good parameter for any other dataset, too. Parameter α_{i} can be estimated from the data using peaks that have no decomposition as a sub-formula of the known molecular formula of the compound, or masses larger than the mass of the precursor peak: These peaks are generally noise peaks. (In case no reference compounds are known in the dataset, we can instead choose those peaks that have no decomposition whatsoever.) We plot relative noise peak intensities in Fig. 6.4 (right). In both datasets, we observe a rapid decay of noise peaks with increasing intensity. The parameters of the Pareto distribution can then be estimated using maximum likelihood estimation. We estimated $\alpha_{\mathrm{i}} = 0.34$ for GNPS and $\alpha_{\mathrm{i}} = 0.5$ for Agilent, see Fig. 6.4. The larger α_{i} for Agilent is probably an artefact of intensity normalisation: If the most intense peak in a spectrum has a low intensity, which happens frequently in high-energy spectra, all other peaks (including noise peaks) have comparatively large relative intensities. We find that the noise parameters $I_{\min} = 0.002$ and $\alpha_{\mathrm{i}} = 0.34$ we used for GNPS are also a good choice for many other datasets. We use these parameters as default in SIRIUS.

6.6.2 Parameters for Isotope Pattern Analysis

Due to a lack of training data, we have chosen the parameters for the isotope pattern analysis by expert guess. We set $\sigma_{ppm} = 5$ and $\sigma_{\mathrm{diff}} = 0.0005$ for Q-ToF instruments and $\sigma_{ppm} = 2$ and $\sigma_{\mathrm{diff}} = 0.0003$ for Fourier transform instruments. The intensity deviation was chosen $\sigma_{\mathrm{rel}} = 0.08$ and $\sigma_{\mathrm{abs}} = 0.02$. Because Fourier transform instruments have larger intensity errors, we increase the absolute intensity error for such instruments to $\sigma_{\mathrm{abs}} = 0.03$. As we now have a large dataset with isotope pattern, we can evaluate how good our chosen parameters fit to the measured data. We compare simulated isotope pattern on the Agilent 2.0 dataset with the measured isotope pattern (both normalised such that the largest peak has intensity 1). Is our assumption correct that low intensive peaks have larger mass deviations? We group isotope peaks in three groups: high intensive peaks (above 20 % intensity), low intensive peaks (below 10 % intensity) and medium peaks in between. See Fig. 6.5 for a histogram of mass deviations within each of these groups. While our basic assumption is correct, we see that the distribution of mass deviations between medium and high intensive peaks are quite similar. The average mass deviation is 0.58 mDa for high intensive peaks, 0.53 mDa for medium intensive peaks, and 2.5 mDa for low intensive peaks. This is quite close to our ad-hoc chosen parameters of 0.5 mDa for Q-ToF instruments. For low intensive peaks, we only increase the mass deviation by three times, while it seems that a larger increase of 5 times would be appropriate.

We cannot measure relative and absolute intensity errors directly, but only the total intensity error. However, on low intensive peaks the absolute error will dominate. Similarly, we can estimate the relative error on high intensive peaks by dividing the intensity error by the measured peak intensity. See Fig. 6.6 for a histogram of relative and absolute intensity error for high and low intensive peaks. On the low intensive peaks the average intensity

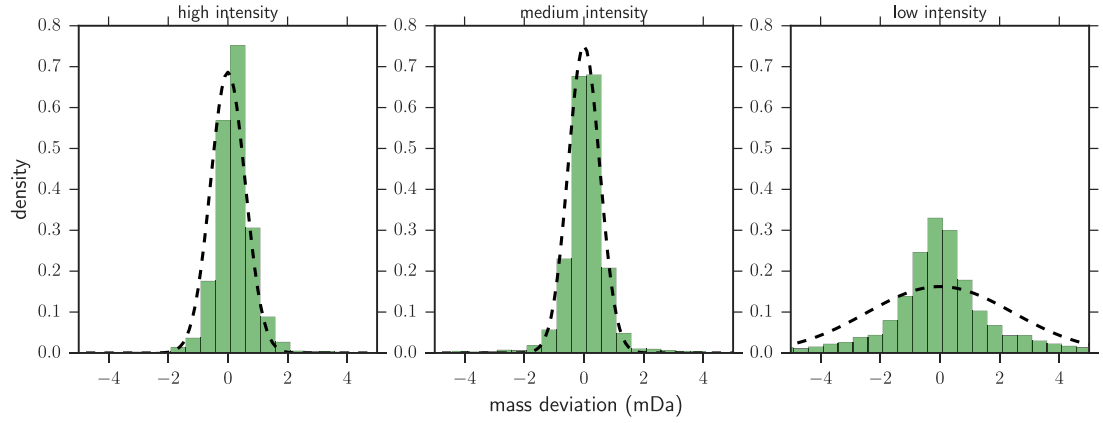


Figure 6.5: Histogram of absolute mass errors of isotope peaks. From left to right: Peaks are grouped in high intensive (above 20 % intensity), medium intensive (between 10 % and 20 % intensity), and low intensive (below 10 % intensity) peaks. Normal distributions with $\mu = 0$ and various σ are estimated from the data and drawn in dashed black lines.

error is 0.004. On large intensive peaks the average relative intensity error is 0.049. Both errors are significantly lower than our chosen parameters. However, the chosen parameters seem to work better on the Agilent 2.0 dataset. This is similar to the σ_{ppm} parameter in Section 6.6.1, where we also use a much larger parameter than we have estimated from data.

6.6.3 Iterative Estimation of Hyperparameters

For the noise intensity estimation we can detect a subset of peaks that are guaranteed to be noise because there is no molecular formula explanation within a sensible mass range or because the peak mass is larger than the precursor ion mass. Of course there might

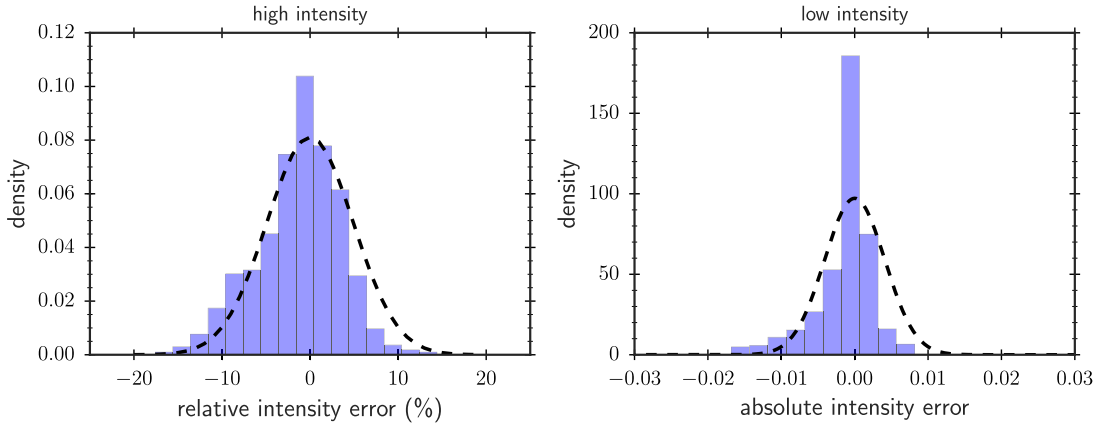


Figure 6.6: Left: Histogram of relative intensity errors from high intensive peaks with more than 20 % intensity. Right: Histogram of absolute intensity errors from low intensive peaks with less than 10 % intensity. Normal distributions with $\mu = 0$ and $\sigma = 0.049$ (left) and $\sigma = 0.004$ (right) are drawn in black dashed lines.

be explanations for these peaks: they might be dimeres or stem from another compound. But for our statistical model it is only relevant that they are not and cannot be part of the tree. The opposite, however, is not possible: We do not know which peaks in a spectrum are signal, and even worse: we do not know their real molecular formula (except for very low mass peaks that only have a single possible explanation). The lack of any "ground truth" makes it difficult to estimate hyperparameters for our statistical model. While there are annotated spectra in the literature as well as lists of common losses and common fragments [124] we have to assume that they are vastly incomplete. Therefore, we estimate the hyperparameters by an approach inspired by expectation maximisation: We estimate the hyperparameters in an iterative procedure, consisting of *rounds*. Each round consists of an expectation step where we compute the fragmentation trees with maximum posterior probability, enforcing the correct molecular formula as root. We then update in the *maximising* step our statistical model (the loss mass prior, common loss prior, and the tree size prior) using the computed trees as ground truth. Afterwards, we restart computation in the next round.

We estimated the hyperparameters on the combined Agilent and GNPS dataset [22]. We started the first round already with some meaningful parameter values: we manually set parameters $\mu_{\text{ls}} = 4$ and $\sigma_{\text{ls}} = 0.5$ for the loss mass distribution, which are estimated from FTs computed using SIRIUS^2-ILP [25, 164, 165]. We also used the manually derived list of common losses [165] with scores that compensate for 75 % of the penalty through the loss mass distribution. In this first round, the list of common fragments was empty, and the tree size prior was set to $P_{\text{tree- size}} = e^5 = 148.41$ to counter the effect of the other priors. After seven rounds we observed that no new common losses and common fragments were found and, therefore, stopped the optimisation routine after the tenth round.

Estimating the Loss Mass Distribution and Common Losses We consider the set of all losses that have been observed in at least one tree, together with their number of appearances (frequency). But instead of purely counting losses, we want to give more weight to losses that correspond to intense peaks. To this end, any loss receives weight corresponding to the maximum peak intensity of the two peaks that are responsible for this loss.

Loss mass distribution and the list of common losses are jointly determined in an inner loop: The loss mass distribution dictates what losses we regard as being "more common than expected". But these common losses, in turn, have to be made "uncommon" for determining the loss mass distribution. We proceed in 6 runs.

Let $l_1, \ldots, l_N$ be the observed losses, $x_1, \ldots, x_N$ the loss masses, and $w_1, \ldots, w_N$ the corresponding weights reflecting peak intensities. We may assume that all losses l_k are pairwise different, summing up weights. Let $w(l)$ be the total weight of some loss $l := u - v$. Further, set $w'_k \leftarrow w_k$ for all $k = 1, \ldots, N$; these will be the weights that are updated in each run. Maximum likelihood estimates of $\mu_{\text{ls}}, \sigma^2_{\text{ls}}$ are

$$\widehat{\mu} = \frac{\sum_k w'_k \ln x_k}{W}, \quad \widehat{\sigma}^2 = \frac{\sum_k w'_k \left(\ln x_k - \widehat{\mu}\right)^2}{W}, \tag{6.19}$$

where $W := \sum_k w'_k$ is the *total weight* of all observed losses. We set $\mu_{\text{ls}} = \widehat{\mu}$ and $\sigma^2_{\text{ls}} = \widehat{\sigma}^2$ for eq. (6.5).

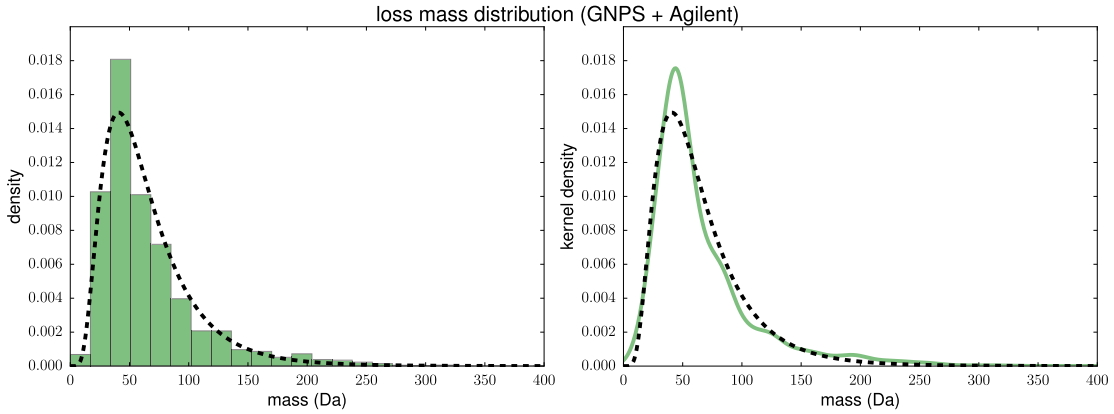


Figure 6.7: Loss mass distribution, after the final round of parameter estimation. Frequencies of the losses are weighted by the intensity of their peaks. The frequency of the identified common losses have been decreased to the value of the log-normal distribution. Left: Normalised histogram for bin width 17 Da (green). Right: Kernel density estimation (green). Maximum likelihood estimate of the log-normal distribution drawn in both plots (black, dashed).

Certain losses appear significantly more often than we would expect from the loss mass distribution. To this end, we use the following two rules to decide whether some loss $l := u - v$ is termed "common":

1. The observed sum of weights for this loss is at least 1.3-fold of what we would expect of eq. (6.5).

2. Large losses will be very rare and, using only the above rule, all of them would be regarded as "common". To this end, we also demand that the frequency has to be at least five above the expected value from eq. (6.5).

Common losses are outliers, in the sense that their frequency is far higher than we would expect for a loss of this mass. To this end, we now correct their weight in a straightforward manner: For each identified common loss l_k, we set its weight to exactly the value we would expect from the loss mass prior, namely $w'_k \leftarrow P_{\text{loss-mass}}(\mu(l)) \cdot W$.

After the *final round* of fitting the hyperparameters, we reached $\mu_{\text{ls}} = 4.02$ and $\sigma_{\text{ls}} = 0.31$. The mode of the log-normal distribution was $e^{\mu_{\text{ls}}} = 55.84$ Da. For each loss l in the identified list of common losses, we set:

$$P_{\text{loss-comm}}(l) := \frac{w(l)}{P_{\text{loss-mass}}\big(\mu(l)\big) \cdot W}$$

Losses H and H_2 are special cases, as they have very low prior probabilities due to the loss mass prior being a log normal distribution. We set the common loss prior for both losses such that the losses are neither penalised nor favoured: In detail, the product of the priors is equal to the geometric mean of the product for all other losses.

See Fig. 6.7 for the agreement between the observed distribution of loss masses (corrected for common losses as indicated above), and the fitted log-normal distribution. See Table A.2 in the Appendix for the list of identified common losses. We find that the resulting list of common losses shows high agreement with the expert-curated lists from [164, 165].

Common Fragments Next to common losses we also observe the same fragments repeatedly in the trees. For each fragment, we compute its weight as the sum of peak intensities of the corresponding peaks. Then, we compute a frequency of each fragment, dividing its weights by the total weight of all fragments. Unlike losses, the diversity of fragments is very high (we observed 13 537 different fragments in our datasets, most of them occurring only one time). To avoid overfitting, we use only the 40 most common fragments and set the common fragment prior to their weight divided by the weight of the 80th most common fragment (39). Both numbers are chosen ad hoc. All other fragments get a flat prior of 1. See Table A.3 in the Appendix for the common fragments that were found in the final round.

Tree Size Prior Finally, we have to determine tree size priors $P_{\text{tree-norm}}$ and $P_{\text{tree-bonus}}$: $P_{\text{tree-norm}}$ is chosen as the inverse of the geometric mean of the priors that any edge in any FT receives. The more interesting prior is $P_{\text{tree-bonus}}$ that can be used to control the size of the trees. We want to ensure that a high percentage of peaks in the fragmentation spectra are explained by our FTs. For the first round we set $P_{\text{tree-bonus}} \leftarrow 1$. In the following rounds we decrease $P_{\text{tree-bonus}}$ by dividing it with $e^{0.25}$. We then re-compute FTs with the new priors of the current round. To decide whether we have explained "enough" peaks, we use the following criteria: We compute the sum of intensities of all peaks that are explained by the FTs. We also compute the sum of intensities of all peaks that *could be* explained by a theoretical fragment, that is, $|m - m'| \leq MA \cdot \max\{m, m_{MA}\}$ for peak mass m and molecular formula mass m'. If the ratio of explained intensities vs. intensities that could be explained, drops below 85 % then we increase $P_{\text{tree-bonus}}$ by multiplying it with $e^{0.5}$, and re-start the computation of FTs. As soon as this ratio is above 85 %, we keep the FTs and proceed to the next round.

After the final found, we reached tree size priors $P_{\text{tree-norm}} = e^{1.46}$ and $P_{\text{tree-bonus}} = e^{-0.5}$.

6.7 Evaluation of the Statistical Model

We have defined the scoring of fragmentation trees as a maximum a posteriori probability estimator. In this section we will evaluate how good this model fits on different datasets, and if the model results in better identification rates than the old scoring. Most parts of the evaluation were done in [22]. However, in [22] we could not evaluate the performance of the isotope pattern analysis as neither the GNPS nor the Agilent dataset contained isotope pattern. With Agilent 2.0 dataset such an evaluation is now possible: 3 502 of the 4 048 compounds in Agilent 2.0 have an isotope pattern with more than two peaks. In Section 6.7.2 we will evaluate the isotope pattern analysis on Agilent 2.0. In [22] we also only had very small independent datasets. Therefore, we will evaluate on NIST as independent dataset with 20 462 compounds.

6.7.1 Evaluation of Hyperparameters

The NIST dataset is almost one order of magnitude larger than the datasets we used to estimate the parameters. Instead of re-estimating the hyperparameters on a larger dataset, we will show that our estimated hyperparameters fit well the distributions on the NIST data. Note that the NIST dataset is measured on Orbitrap instruments and contains positive mode and negative mode spectra, while we estimated the parameters on

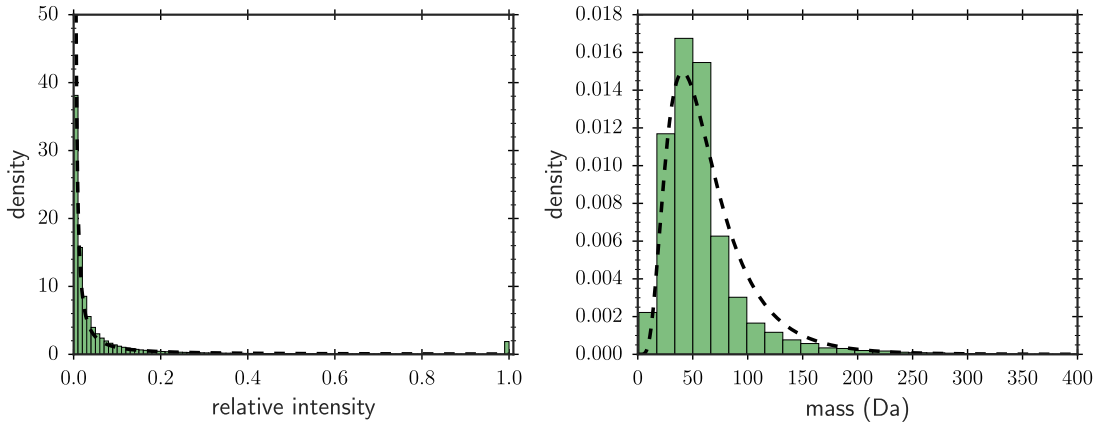


Figure 6.8: Left: Normalised histogram of intensities of noise peaks in the NIST dataset. The Pareto distribution with $I_{\text{min}} = 0.002$, $\alpha_{\text{i}} = 0.34$, and $\beta_{\text{i}} = 10^{-5}$ estimated on the GNPS data is drawn as dashed line. Right: Normalised histogram of loss masses from all trees in NIST dataset. The frequency of the identified common losses have been divided by the $P_{\text{common-loss}}$ prior. The log-normal distribution learned on GNPS and Agilent data is drawn as dashed lines.

spectra measured on Q-ToF instruments in positive mode. We compute fragmentation trees using the same parameters as for the GNPS dataset: $MA = 10$ ppm, $m_{MA} = 250\,\text{Da}$, $I_{\text{min}} = 0.002$ and $\alpha_{\text{i}} = 0.34$. We enforce the correct molecular formula for the root of the trees. As it is described in Section 6.2.1, we start with $P_{\text{tree-bonus}} = -0.5$ and increase the prior until we explain above 80 % of the intensity in the spectrum.

First, we collect all peaks in the spectra that have no explanation in the tree. See Fig. 6.8 (left) for a plot of the intensities of these peaks that are likely noise. Our Pareto distribution fits very well to the noise peak distribution. However, we find two issues: First, there are very few noise peaks in low intensity regions. This can be observed also in other datasets that are measured on Orbitrap instruments. It seems that Orbitrap instruments measure much less peaks (and also much less noise) than Q-ToF instruments. But the same effect also happens for other instruments and can be easily explained: Although there might be a very large number of peaks with intensity close to zero (for example: electronical noise), they are not picked as peaks during the peak picking step. Another problem is for peaks with intensity 1: it seems that there are sometimes very high intensive peaks in the spectrum that cannot be explained by the tree. Interestingly, this happens very often for peaks with maximum intensity. We did not observe such a hump for the GNPS and Agilent data. To ensure that this hump is not an error in our tree computation, we again collect noise peaks, but this time we only choose peaks that do not have a molecular formula explanation within 10 ppm that is a subset of the compounds molecular formula. We again find the same hump: 7 196 unexplainable peaks (1.33 % of all unexplainable peaks) have an intensity of 100 %. We assume that these are artefacts of the Fourier transformation. In most cases, these peaks have very strange masses which do not lead to meaningful molecular formula decompositions.

Is the tree size prior estimated on Agilent and GNPS still a good choice for NIST? We find that 74.5 % of the trees are computed with $P_{\text{tree-bonus}} = -0.5$ and. The default value for the tree size prior is sufficient for most of the spectra.

Next, we report the distribution of loss masses in Fig. 6.8 (right). We find that it looks very similar to the loss mass distribution we learned from Agilent and GNPS data. We learned common losses from data instead of relying solely on literature because we assume that any list of common losses we can find in literature will be incomplete. Now, with the number of compounds in our dataset increased from 4 051 (GNPS and Agilent from [22]) to 20 462 (NIST) we have to ask: Do we still have a comprehensive list of common losses, or do we observe many new losses in this much larger dataset? To answer this question we count for each loss how often it occurs within trees in the NIST dataset. We then compare this number with the expected number of occurrences computed with our statistical model (the loss mass prior times the common loss prior). In total, the trees in NIST have 520 420 losses. Surprisingly, there is not a single loss that occurs very frequently and is underestimated by our statistical model. Note, that the prior probability cannot be directly used as "expected frequency" of a loss. In particular, if we would sum up all expected relative frequencies, we end up with a sum above 1.

Finally, we can conclude that the set of common losses we learned from the GNPS and Agilent dataset is still valid and cannot be extended even when looking into a dataset that is five times larger. In total, we observed 4 899 different losses. Only 68 of them are contained in our set of the common losses. However, 83.7 % of all losses in the computed fragmentation trees are common losses. This demonstrates that the set of common losses is quite comprehensive, while our statistical model still allows the insertion of many 'non-common' losses in the tree.

6.7.2 Molecular Formula Identification

SIRIUS was developed with the objective of identifying the molecular formula of the precursor ion. Here, we evaluate how good SIRIUS achieve this task with the new maximum a posteriori probability estimate scoring. The evaluation on the Agilent and GNPS data was initially done in [22] using SIRIUS 3. Here, we repeat this evaluation using the up-to-date version SIRIUS 4. With the Agilent 2.0 dataset we can now also evaluate how good SIRIUS performs when combining MS isotope pattern analysis with MS^2 fragmentation pattern analysis. Finally, we evaluate on four independent datasets.

As most reference spectra from database do not contain MS information, we have to evaluate such instances using solely the fragmentation pattern analysis. This means we cannot use the automatic element detection to restrict the set of possible chemical elements. As said earlier, allowing the large alphabet CHNOPSClBrIF is not feasible. Therefore, we recommend to use the alphabet CHNOPS when no MS is available. Other elements should only be added if the presence of such elements is highly expected in the sample. In our evaluation, we will therefore split the datasets without MS into two batches: The "CHNOPS" batch contains only instances composed of the elements C,H,N,O,P, and/or S. Other compounds which contain halogen elements are put into the "contains FClBrI" batch. For batch CHNOPS, SIRIUS is run using this alphabet of elements without any further restrictions. For batch "contains FClBrI" we assume that we know upfront which of the elements, besides CHNOPS, *may* be contained in the compound: For example, for a compound with molecular formula $C_{18}H_{13}ClFN_3$ we start our analysis over the alphabet CHNOPSClF, but SIRIUS may still (wrongly) decide that the compound contains no chlorine or fluorine. This covers the case where we have some indications for the presence of these elements, but have to consider false positives. We do not restrict the number of atoms for each element. When we have MS data available, we will use the alphabet

C,H,N,O,P,F, and I instead, but use automatic element detection to add any of S,Cl,Br to the set of allowed elements.

We evaluate the performance of SIRIUS against existing methods for determining the molecular formula using MS/MS data. As a baseline method to evaluate against, we use the *naïve method* that returns the molecular formula with the smallest mass difference to the measured parent mass. This method completely ignores all fragmentation data, but will nevertheless in some cases find the correct answer, in particular if there are only few possible explanations of the parent mass. This strategy identifies the correct molecular formula for 14.6 % of the instances, and in 31.2 % the correct formula can be found in the top 5. Both of our datasets have no systematic mass error, see Fig. 6.4 for the GNPS dataset; for datasets that show a systematic mass error, we expect worse identification rates for the naïve method.

Another common approach is to search the neutral parent mass in a compound database. If we restrict our search to molecular formulas that are contained in PubChem, and again rank molecular formula candidates by the mass difference to the measured parent mass, we find the correct molecular formula for 17.1 % of the instances in top rank, and 59.8 % in the top 5. This approach is, by design, restricted to molecular formulas that are already known, and must naturally miss cases where no molecular formula is contained in a structure database. The improved performance is, hence, solely based on the reduced number of candidate molecular formulas, in particular for larger masses. We stress again that SIRIUS is *not* restricted to molecular formulas from any database.

We compare SIRIUS 4 against its predecessor, the computational method from [164] with the score modifications from [165]. This method has been released as "SIRIUS^2 (version 1.0)", and will be referred to here as "SIRIUS^2-DP". SIRIUS^2 does not use the Integer Linear Program proposed in [166] for computing FTs but instead, combines Dynamic Programming (DP) with a heuristic. This combination of algorithms is possibly inferior to the ILP from [166] used here, so we also combined the old SIRIUS^2 scoring with the ILP from [166]; this method is referred to as "SIRIUS^2-ILP" in the following.

In Fig. 6.9 we report whether the true molecular formula is contained in the top k output of the different methods, for varying k. We find that SIRIUS 4 can correctly identify the molecular formula for 73.8 % of the instances, compared to 31.1 % for SIRIUS^2-DP and 39.1 % for SIRIUS^2-ILP. Using an ILP [166] instead of the original dynamic programming algorithm does result in both better identification rates and decreased running times. But the better part of performance improvements must be attributed to the new scoring presented here: We observe a 2.4-fold increase of correct identifications when compared to SIRIUS^2-DP, and roughly a 2-fold increase when compared to SIRIUS^2-ILP.

Fig. 6.10 shows identification rates as a function of compound mass. Identification rates of SIRIUS 4, SIRIUS^2-ILP, and SIRIUS^2-DP decrease with increasing mass: This can be attributed to the fact that more candidate molecular formulas have to be considered for larger masses, compare to Fig. 4.1. Searching the precursor peak mass in PubChem, we observe better identification results for mass bins 600–800 Da and 800+ Da than for mass bin 400–600 Da. As mentioned above, this can be interpreted as an artefact of the distribution of molecular formulas in PubChem: As seen in Fig. 4.1, the number of candidate molecular formulas in PubChem reaches its maximum for mass bin 400–600 Da. Regarding the distribution of compound masses in the two datasets, we observe that the vast majority have masses below 650 Da, see again Fig. 6.10.

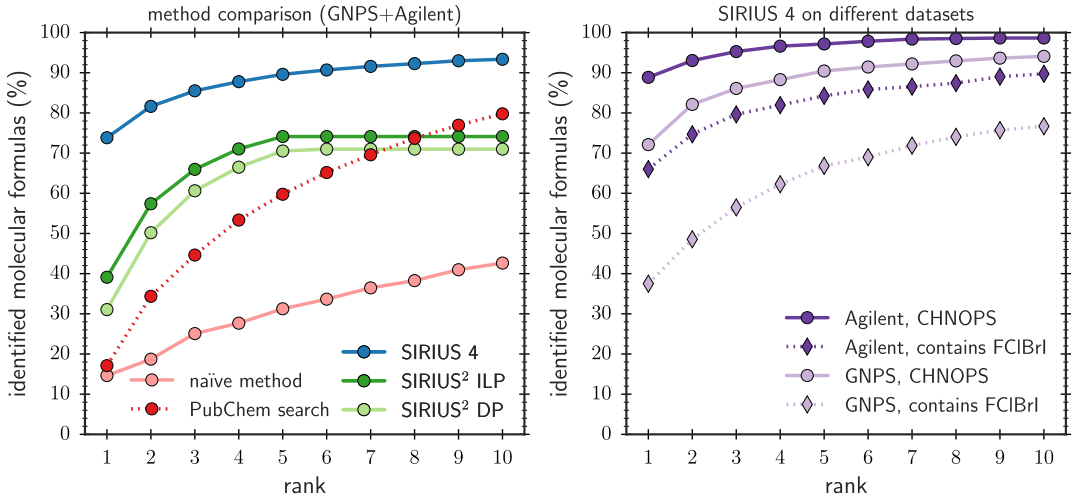


Figure 6.9: Performance evaluation, percentage of instances (y-axis) where the correct molecular formula is present in the top k for $k = 1, \ldots, 10$ (x-axis). Left: Performance evaluation for different methods on both datasets. Methods are "SIRIUS 4" (method presented here), "SIRIUS^2-ILP" (scores from [164, 165] solved by Integer Linear Programming), "SIRIUS^2-DP" (scores from [164, 165] solved by Dynamic Programming), the "naïve" method (searching all possible molecular formula for the closest precursor mass), and "PubChem search" (searching PubChem for the closest precursor mass). Right: Performance of SIRIUS 4 for the two compound batches (CHNOPS as solid line, "contains FClBrI" as dashed line) and the two datasets (GNPS in light violet and Agilent in dark violet).

To support our claim that the observed performance gain, when searching large compounds in PubChem, is simply an artefact, we also restricted SIRIUS 4 to molecular formulas from PubChem (Fig. 6.10). We stress once more that unless explicitly stated, SIRIUS 4 will consider *all* molecular formulas. Up to 600 Da, identification rates are roughly on par with SIRIUS 4 considering all possible molecular formulas. For larger mass, identification rates for the smaller PubChem candidate lists outperform the regular SIRIUS 4.

To the best of our knowledge, the only other method that we could evaluate against, is GENFORM [131], initially introduced in 2011 by MOLGEN (Bayreuth, Germany) and later released as open source software. In GENFORM, the fragments are inserted directly below the parent ion, but peak intensities as well as mass deviations of the fragments are taken into account in the scoring. However, GENFORM requires both, MS^2 and isotope patterns, to identify the molecular formula. The MS^2 analysis alone is not discriminative enough for this task. Evaluations by Stravs *et al.* [206] indicated that GENFORM when run with MS^2 and isotope pattern data is roughly on par with SIRIUS^2-DP. We repeat this evaluation with SIRIUS 4, see below. We will also compare identification rates between SIRIUS 4 and GENFORM on the Agilent 2.0 dataset.

In silico fragmenters like CFM-ID and combinatorial fragmenters like MetFrag can be used to determine the molecular formula, but only if the corresponding molecular structure is contained in a structural database. Note that structural databases are very incomplete. We cannot assume that every molecular formula that exists in nature is contained in structural databases. But such an approach would require not only the molecular formula but also the correct molecular structure to be contained in a database. Another problem

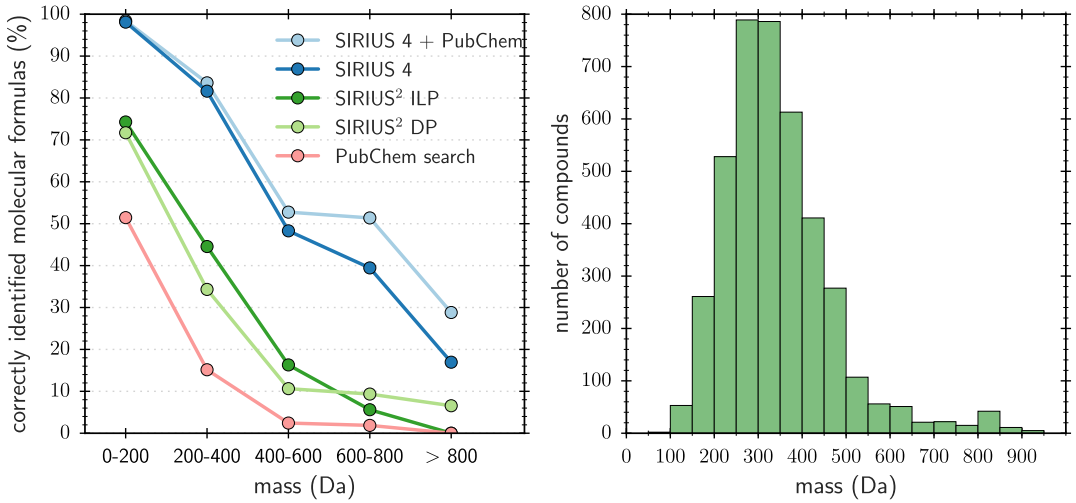


Figure 6.10: Left: Identification rates of all methods in dependence on the mass of the compound, compare to Fig. 6.9. Restricting SIRIUS 4 to molecular formulas from PubChem is included for comparison. Right: Histogram for masses of all compounds in the two datasets, bin width 50 Da.

is that molecular databases are highly biased: PubChem (status as of August 13, 2017) contains 2 585 376 different molecular formulas and 74 132 874 molecular structures. So in average we would expect around 29 structures per molecular formula. However, there are 167 different structures for the molecular formula $C_6H_{12}O_6$, but only 5 different structures for the molecular formula $C_5H_{15}N_3S_2$ which has a similar mass. The prior probability (that is, the probability when completely ignoring the data) for ranking a structure with $C_6H_{12}O_6$ on top position is therefore much larger than for $C_5H_{15}N_3S_2$. This result into a confirmation bias and an overestimation of the identification performance. But despite this advantage, SIRIUS 4 shows a better performance for molecular formula identification than these tools, see below.

Evaluation on Independent Data To show that we have not "overtrained" SIRIUS, we evaluate its performance on four independent datasets. We *do not re-estimate* any hyperparameters for these evaluations but rather use those described above. Only mass accuracy and the set of elements are chosen appropriately, see below.

First, we evaluate SIRIUS 4 on independent data by Stravs *et al.* [206]. This dataset contains 60 compounds (pesticides) with masses between 191.1 Da (DEET) and 443.1 Da (Propaquizafop). Here, 28 compounds contain a halogen element. Both isotope patterns and MS/MS data are provided. We use the alphabet of elements CHNOPSClBrI for all instances, and mass accuracies 5 and 10 ppm as suggested in [206]. Results of GENFORM and SIRIUS^2-DP are taken from [206]. For evaluation purposes, we estimate the power of SIRIUS 4 using only MS/MS data, and we also combine results of isotope pattern and MS^2 analysis as described in [57]. See Table 6.2 for details. We find that combining MS/MS and isotope pattern data, SIRIUS 4 correctly identified molecular formulas for 93.3 % of the instances and for 26 of 28 halogenated compounds, clearly outperforming both GENFORM and SIRIUS^2-DP. When omitting the isotope pattern data, SIRIUS 4 still identified the correct molecular formula for 81.7 % of the instances, and for 18 out of 28 halogenated compounds.

Table 6.2: Performance comparison of SIRIUS 4 with GENFORM using 60 compounds from [206], uncalibrated spectra. All tools are run with mass accuracy parameter 5 ppm and 10 ppm. Best entries in bold. Results for GENFORM and SIRIUS^2-DP taken from [206]. In that evaluation, SIRIUS^2-DP crashed 7/5 times for 10 ppm/5 ppm mass accuracy, and did not consider the correct molecular formula of the compound for 0/6 compounds.

	GENFORM		SIRIUS^2-DP		SIRIUS 4			
	with isotopes		with isotopes		w/o isotopes		with isotopes	
	10 ppm	5 ppm	10 ppm	5 ppm	10 ppm	5 ppm	10 ppm	5 ppm
Top 1	36	34	34	35	49	45	**55**	**56**
Top 2	44	47	50	46	51	51	**58**	**58**
Top 5	54	55	52	48	58	**60**	**60**	**60**
Average rank	2.55	2.30	1.57	1.63	1.58	1.5	**1.17**	**1.15**
Worst rank	23	20	11	15	10	**5**	**5**	**5**

Second, a preliminary version of SIRIUS 3 was used in the CASMI contest 2013 to determine the molecular formula of 12 unknown compounds. Using only the fragmentation tree analysis described here and ignoring the isotope pattern data, we correctly identified 8 molecular formulas, and placed an additional 3 in the top 2 [57]. In conjunction with isotope pattern analysis, we identified 10 out of 12 molecular formulas, and SIRIUS was selected "best automated tool" of the molecular formula challenge [148]. However, the number of challenges is too small to do any useful statistics. The CASMI contest 2016, however, consists of 208 challenges, with 127 compounds in positive and 81 negative ion mode, respectively [191].

Although, the CASMI 2016 contest is about identifying the molecular structure, we will evaluate here how often SIRIUS 4 can identify the correct molecular formula from the data: For evaluating the molecular formula identification performance, we *do not* restrict SIRIUS to use molecular formulas from some structure database; instead, we allow all possible molecular formulas that can be composed from elements CHNOPSFIBrCl and enable automated element detection. Furthermore, we do not expect (de)protonation as default, but allow any of the following adducts: [M+H]+, [M+Na]+, [M+K]+ for positive mode and [M-H]-, [M+Cl]- and [M+Br]- for negative mode. We find that SIRIUS 4 identified the correct molecular formula for 190 of 208 challenges (91.3 %); for all but ten challenges, the correct answer is in the top 4. For the ten challenges where SIRIUS could not find the correct formula, we identified three types of problems: For challenges 22, 64, 71, 113, and 202 the isotope pattern is heavily disturbed. This might be due to coeluting compounds with similar masses that overlap in the isotope patterns. Such problems can potentially be resolved by deconvolution of isotope peaks based on their elution profiles (extracted ion chromatograms). In challenges 75 and 193, both compounds contained six chlorine atoms. Unfortunately, the automated element detection estimated a maximum of four chlorine atoms, as well as the presence of bromine. This can be attributed to the fact that isotope patterns of Cl_6 and Cl_2Br look rather similar. It indicates that there is still room for improving the automated element detection. Finally, for challenge 78, no MS1 spectrum was provided and the MS/MS spectrum contains only two peaks: The precursor peak and a single FH loss. Even within 1 ppm, there are 230 fluorine containing molecular formulas which explain the precursor peak. To identify the correct molecular formula, SIRIUS would need more MS/MS peaks or isotope pattern data.

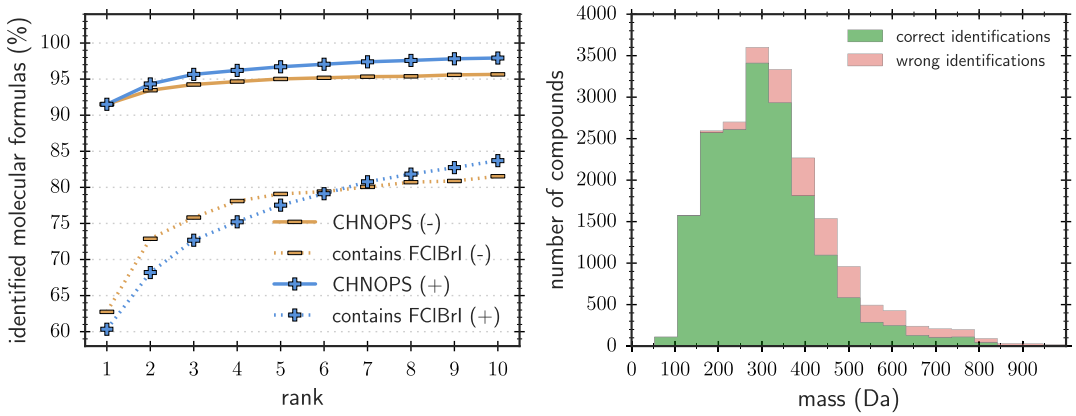


Figure 6.11: Left: Identification rates of SIRIUS 4 on 20 462 compounds from NIST. Identification rates for batch "CHNOPS" are drawn in solid lines and for batch "contains FClBrI" are drawn in dashed lines. Identification rates are coloured blue for compounds measured in positive mode and orange for compounds measured in negative mode. Right: Histogram of the mass distribution of NIST compounds. The histogram bars are further divided into compounds whose molecular formula could be identified on top rank by SIRIUS 4 (green) and instances which are identified on a rank above 1 (red).

We compare the SIRIUS 4 results with results from the combinatorial/in- silico fragmenters. Note that these tools identify molecular structures, not molecular formulas. But obviously, we get the molecular formula for free after identifying the structure. Note also, that these tools search in a structure database and, therefore, have to consider several magnitudes less candidates than SIRIUS that considers all possible molecular formulas. Furthermore, they benefit from the confirmation bias we wrote about above: just because a molecular formula is present much more frequently than other formulas makes it more likely that this formula is ranked on top position: for example, in six challenges only a single molecular formulas is present in the candidate list and for 48 challenges, 90 % of the structures have the same molecular formula as the correct compound. Despite this advantage, no combinatorial fragmenter could identify more molecular formulas in CASMI 2016 than SIRIUS: From a total of 208 challenges, MAGMa identified 179, CFM-ID 160, and MS-FINDER 167 molecular formulas correctly [191]. SIRIUS identified molecular formulas for 190 of 208 challenges correctly.

Finally, we use MS/MS spectra of 20 462 compounds from NIST as independent data. Again, we will split the dataset into two batches: "CHNOPS" with 16 948 and "contains FClBrI" with 3 514 compounds. Identification rates are reported in Fig. 6.7.2. SIRIUS 4 is able to correctly identify 91.6 % of the CHNOPS compounds and 60.8 % of the "contains FClBrI" compounds. There is no clear difference between positive and negative ionised spectra: 85.7 % of the positive and 88.0 % of the negative ionised spectra are identified correctly on top rank. Again, SIRIUS performs much better on low mass compounds due to smaller candidate lists. For compounds below 300 Da SIRIUS has an identification rate of 97.6 % on the NIST dataset. Identification rates on NIST are very close to the results for the Agilent dataset. This is due to the fact that both datasets are high quality reference spectra libraries. Remember that we trained our method solely on positive spectra from the Agilent and GNPS dataset but not on negative spectra. The excellent performance on NIST is another evidence for the robustness of our statistical model.

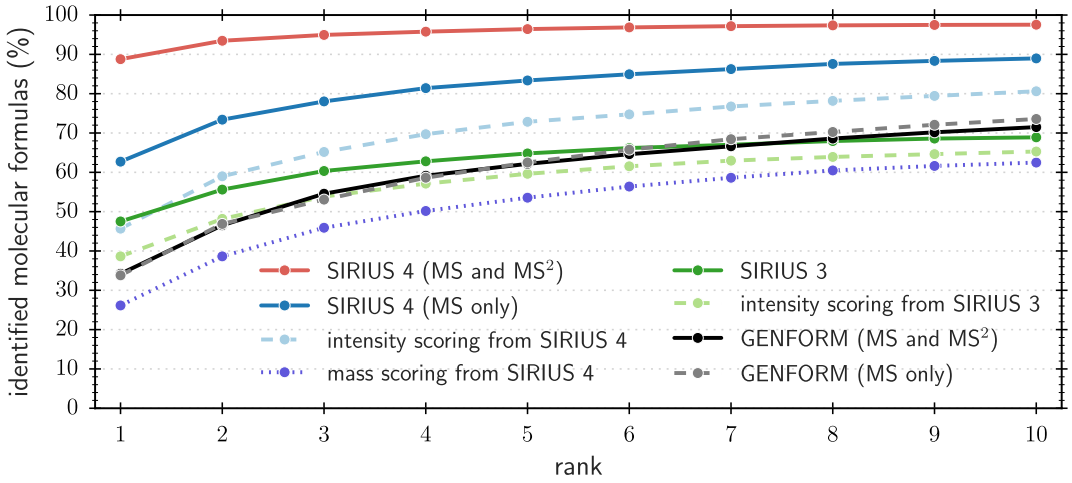


Figure 6.12: Identification rates for molecular formula identification of different methods: SIRIUS 4 using MS and MS^2 in red. SIRUS 4 and SIRIUS 3 using only isotope patterns from MS are drawn in blue and green. Methods that only score isotope peak intensities are drawn in dashed lines, namely: SIRIUS 4 (blue), SIRIUS 3 (green), and GENFORM (gray). GENFORM using MS and MS^2 is drawn in solid black line. The SIRIUS 4 scoring that only scores isotope peak masses is drawn in dotted lines (violet).

Evaluation of Isotope Pattern Analysis We have shown that the new maximum a posteriori probability estimate scoring greatly improves the identification rates. But does the new scoring for isotope pattern have a similar impact? We use the Agilent 2.0 dataset to evaluate the isotope pattern analysis. We omit 466 instances from the dataset because their MS spectra contain less than three peaks. Note, that we still could identify most of these spectra solely by fragmentation pattern analysis. Of the remaining 3 502 instances, 2 340 compounds are composed of the elements C, H, N, O, P, F, and I. 620 compounds contain sulphur and 670 compounds contain chlorine or bromine. We first evaluate how often the deep neural network can detect the correct elemental composition. We find that for 3 492 of the 3 502 instances (99.7 %) the correct elemental composition is predicted. For 9 spectra, the deep neural network could not detect the presence of sulphur. For one spectrum it misses the presence of a chlorine. Furthermore, the deep neural network made 390 false positive predictions for sulphur and 36 false positive predictions for halogens. For sulphur the detector has a recall of 98.4 % and a precision of 61 %. For halogens the recall is 99.9 % and the precision is 94.9 %.

Next, we evaluate how often the maximum likelihood scoring is able to identify the correct molecular formula. We compare these results with the old scoring of SIRIUS 3 [28, 57] and with GENFORM. We ensure that SIRIUS and GENFORM use the same candidate lists and, therefore, the same chemical alphabet, the same mass deviation tolerance and the same filtering step. The chemical alphabet is predicted using the deep neural network. The mass deviation in GENFORM can only be specified as relative measure in ppm. Therefore, for compounds with mass below 250 we specified a larger mass deviation such that the absolute mass deviation is always 2.5 mDa. For all other compounds we set the option `ppm=10`. SIRIUS is only considering molecular formulas with positive RDBE. In GENFORM we enabled this filtering with the option `vsm2ap2=0`. We tried different parameters for GENFORM and found that `wi=lin acc=2 rej=10` give best results.

We report the results in Fig. 6.12. The isotope pattern scoring in SIRIUS 4 and SIRIUS 3 is based on Bayesian statistics. Both assume normal distributed mass deviations on the monoisotopic peak and on the mass differences between the isotope peaks. However, SIRIUS 4 models the intensity error as a mixture of two Gaussians and uses maximum likelihood, while SIRIUS 3 is only considering relative intensity errors using a log normal distribution. SIRIUS 4 identifies the correct molecular formula for 62.7 % of the instances. SIRIUS 3 identifies 47.5 % of the instances correctly. This is an improvement of 30 % due to the new maximum likelihood scoring. Note that we did not optimise the hyperparameters on the Agilent 2.0 dataset. We evaluate the different scorings for isotope peak intensities separately: The maximum likelihood method in SIRIUS 4 achieves an identification rate of 45.7 %. The method from SIRIUS 3 identifies 38.63 % of the instances. The isotope pattern analysis in GENFORM searches for the molecular formula explanation with minimum absolute intensity error [107, 131, 158]. This method identifies the correct molecular formula for 34.3 % of the instances. Finally, we compare against a scoring that ignores intensities and is using only mass deviations and mass difference deviations. Such a scoring can only identify 26.2 % of the instances. Note that, while the isotope pattern analysis itself only achieves identification rates up to 62.7 %, together with the fragmentation pattern analysis SIRIUS can identify the correct molecular formula for 88.8 % of the instances. Such an improvement due to the combination of MS and MS^2 information cannot be observed for GENFORM. For GENFORM, the identification rates increases by 0.4 percentage points to 34.2 % when using both, MS and MS^2. When using solely MS^2 spectra, the identification rate drops to 1.69 %.

7 Structural Elucidation

In this chapter we introduce CSI:FingerID, a method for structural elucidation of small molecules from MS^2 that combines techniques from combinatorics and machine learning. Different from other methods for this task, it can extract valuable information about the molecular structure without having to search in structure databases. This makes it suitable for analysing *unknown unknowns*, compounds which are not contained in any database. But as we will demonstrate in Section 7.9, CSI:FingerID is also the best performing method for searching structure databases with MS^2 spectra.

CSI:FingerID is the successor of FingerID, which was shortly introduced in Section 4.4. Both methods predict a molecular fingerprint from a tandem mass spectrum and search this fingerprint in a structure database to identify the compound. In contrast to FingerID, which uses a probability product kernel on mass spectra, CSI:FingerID is using multiple kernel learning to integrate fragmentation tree information. Furthermore, it predicts more molecular properties than FingerID, in particular a comprehensive list of autogenerated molecular properties. Instead of binary predictions, CSI:FingerID predicts posterior probabilities, and uses a scoring method based on posterior probabilities for searching in structure databases.

We will first outline the original FingerID method and how to integrate fragmentation trees via multiple kernel learning. We will explain the fragmentation tree kernels and the additional molecular fingerprints. Next, we will describe the scoring. We will evaluate the individual kernels and the multiple kernel learning approach. Finally, we will evaluate the performance of CSI:FingerID on multiple datasets and compare this method against a comprehensive list of other tools for searching MS^2 spectra in structure databases.

The use of multiple kernel learning and fragmentation tree kernels was presented at the annual international conference "Intelligent Systems for Molecular Biology" (ISMB) in Boston, 2014 [196] . The improved scoring and additional fingerprints were published in [58]. Note that we renamed some kernels to make the names and abbreviations more consistent. See Table A.6 in the Appendix for a listing of renamed kernels.

7.1 Molecular Fingerprint Prediction From Tandem Mass Spectra

We will first formulate the problem of structure elucidation from MS^2 data: Let $\mathcal{M}$ be the set of all molecules and $\mathcal{S}$ be the set of all MS^2 measurements from these molecules. We want to find a mapping $\mathcal{F} : \mathcal{S} \to \mathcal{M}$ that predicts the molecule that belongs to a given MS^2 measurement. Of course, we only have a limited number of training spectra where we know the correct mapping between spectrum and molecule. We want to find a function $\mathcal{F}$ that generalises well and is able to predict the correct molecular structure for spectra of unknowns, too. To tackle this problem with machine learning methods, we first have to find a transformation that maps our molecules and spectra to numerical vectors. Alternatively,

we can use kernel methods and define a positive definite function $K : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ that computes the similarity between two spectra.

Heinonen *et al.* [79] encode molecules as molecular fingerprints and use SVMs to predict each of the molecular properties from MS^2 data. Note that the mapping from molecule to fingerprint is not bijective; it is impossible to derive the molecular structure given the fingerprint. Heinonen *et al.* [79] suggest to search the predicted molecular fingerprint in a structure database to obtain the molecular structure.

But how can we encode a mass spectrum as numerical vector? Heinonen *et al.* [79] suggest three transformations for mass spectra, using peaks, losses, and peak differences. The first transformation defines the feature vector as one/zero vector s, where s_i is one if there is a peak with rounded mass i in the spectrum and zero otherwise. The loss transformation is obtained in the same way, but it computes the feature vector of the inversed spectrum. Remember, that the inversed spectrum is obtained by subtracting the precursor mass from all peak masses. Finally, the pairwise peak difference transformation is collecting all rounded mass differences between all pairs of peaks in the spectrum.

The disadvantage of these transformations is that they omit all information coming from high resolution and high mass accuracy. One could multiply all peak masses with some constant before rounding, but this would lead to two other problems: first, the feature space would get very large (e.g. multiplying with 1 000 results in 1 000 000 possible masses for small compounds up to 1 000 Da). But this is not a big problem, because most SVM implementations can deal with sparse feature representations. The more serious problem is that misalignments can occur due to measurement errors; that means that the same ion species is represented at different positions in the feature vector between two measurements. Therefore, Heinonen *et al.* [79] represent a spectrum as mixture of Gaussians and use the probability product kernel [96] to compute the expected number of common peaks between two spectra, assuming normally distributed measurement errors. They also integrate peak intensities by using two dimensional Gaussians, one dimension for mass and one for intensity. Given a spectrum $s \in \mathcal{S}$ with s_i is the i-th peak in s with mass $\mu(s_i)$ and intensity $\iota(s_i)$. The PPK (probability product kernel) represents each peak s_i in s as Gaussian $p_{s_i} = \mathcal{N}((\mu(s_i), \iota(s_i), \Sigma)$. Here, Σ is the covariance matrix

$$\Sigma = \begin{pmatrix} \sigma_\mu^2 & 0 \\ 0 & \sigma_\iota^2 \end{pmatrix}$$

which is shared by all peaks. The spectrum s is then a mixture of Gaussians: $p_s = \frac{1}{|s|} \sum_{i=1}^{|s|} p_{s_i}$. The probability product (PP) of two spectra is defined as:

$$\begin{aligned} PP(s, s') &= \int p_s(x) \cdot p_{s'}(x) dx \\ &= \frac{1}{|s| \cdot |s'|} \sum_{\substack{i=1,\ldots,|s| \\ j=1,\ldots,|s'|}} \frac{1}{4\pi\sigma_\mu\sigma_\iota} \cdot \exp\left(-\frac{(\mu(s_i) - \mu(s_j))^2}{4\sigma_\mu^2} - \frac{(\iota(s_i) - \iota(s_j))^2}{4\sigma_\iota^2} \right) . \end{aligned} \quad (7.1)$$

These kernels perform significantly better on high resolution mass spectra than their integral counterparts. We call the probability product kernel on peaks PPK_{peak}, the probability product kernel on the inverse spectrum PPK_{diff}, and the probability product kernel on pairwise peak differences $\text{PPK}_{\text{pairs}}$. Heinonen *et al.* [79] tried linear combinations of these kernels as well as quadratic combinations and found that a uniformly weighted

linear combination of PPK_{peak} and PPK_{diff} result in the best identification performance. We will refer to the uniform combination of these two probability product kernels as *PPK*.

7.2 Kernel Methods for Fragmentation Trees

It must be understood that the feature representation from [79] provides only limited information for machine learning methods while using higher order kernels like radial basis functions might lead to overfitting. A good kernel should yield high similarities for two related objects. Two spectra are similar if their peaks are similar. But when are two peaks similar? The peak kernel effectively decides for two peaks to be similar, if they have the same mass (considering some measurement error). But it cannot say anything about peaks from similar fragments with different masses, for example two peaks that differ by an H_2O loss. Fragmentation trees, however, assign to each peak a molecular formula, which itself is a set of atoms and, therefore, contains much more information than just a scalar mass value. Furthermore, fragmentation trees contain structural information, namely relationships between peaks as well as losses. $\text{PPK}_{\text{pairs}}$ has a low prediction performance, because the number of peak differences is increasing quadratically with the number of peaks, while only a small minority of differences represent real fragmentation reactions. In fragmentation trees, however, the number of losses is equal to the number of fragment peaks. Although fragmentation trees are computed from mass spectral data, it must be understood that this computation is a highly non-linear and non-trivial transformation of the spectrum that cannot be learned by a linear machine learning method. We propose the following strategy for learning from mass spectra: first compute a fragmentation tree with SIRIUS; afterwards compare the trees using a fragmentation tree kernel instead of just comparing the spectra.

Let $\mathcal{F}$ be the set of all fragmentation trees. A fragmentation tree kernel is a positive definite function $F : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$. For a tree $x \in \mathcal{F}$ we will denote its root as $r(x)$, the set of all nodes as $N(x)$ and the set of all edges as $E(x)$. As each node in the tree corresponds to a peak in the mass spectrum, we will denote the mass of a node u as $\mu(u)$ and its intensity as $\iota(u)$. The molecular formula of the path from u to v is $\lambda(u, v) = u - v$.

In the following sections we will introduce a set of kernels on fragmentation trees. All of them fit into the framework of decomposition kernels from [77] and have the general form

$$K(x, x') = \sum_{p \in d(x)} \sum_{p' \in d(x')} w(p) \cdot w(p') \cdot K_d(p, p') \,. \tag{7.2}$$

Here, d is the decomposition function that maps the input to a multiset of features. For example, d can map a fragmentation tree to a set of nodes or a multiset of paths. And d can map a molecular formula to a multiset of atoms. w is a weighting function that gives some features larger weights than others. Often, we will downweight very frequent features by dividing them by their average frequency. Finally, K_d is a kernel that compares features. If this kernel is checking for equality (thus, $K_d(p, p') = 1$ iff $p = p'$ and $K_d(p, p') = 0$ iff $p \neq p'$), then K is a counting kernel that counts common features in the decomposition of both inputs. A special case is the binary kernel, where we assume that the decomposition is always a set and not a multiset. The underlying feature space of counting kernels is a vector v with v_i is the frequency of i in the decomposition. For binary kernels, these vectors are binary.

Proving for all of these kernel that they are positive definite is out of scope for this thesis. See [77] for the proof that eq. (7.2) is positive definite if K_d is positive definite. It is easy to see that all kernels described in the following are kinds of decomposition kernels and, therefore, positive definite.

7.2.1 Comparing Fragments and Losses

As the PPK kernel operates on peaks and root losses, the counterpart of this kernel on FT side is to compare nodes and root losses (paths from the root to any node in the tree). We define the *NB* (node binary) and *RLB* (root loss binary) kernel as:

$$NB(x, x') = \sum_{\substack{u \in N(x), v \in N(x'), \\ u=v}} 1 \tag{7.3}$$

$$RLB(x, x') = \sum_{\substack{u \in N(x), v \in N(x'), \\ r(x)-u=r(x')-v}} 1 \tag{7.4}$$

The PPK kernel also compares intensities between two peaks and root losses. In general there are two ways to compare intensities: we can use a radial basis function that yields high similarities for similar intensities and lower similarities with increasing absolute difference between intensities. Alternatively, we can weight peaks by intensity and let the SVM priorities peaks with larger intensity. We argue that intensities are not very reliable and robust for predicting molecular properties. Two measurements of the same compound can have very different peak intensities; this depends on the used collision energy and stereoisomerism. Although we could train the SVMs on spectra with same collision energies, as it is done in [4, 5], we would then loose most of our training data and the resulting model would be applicable only on data measured with the same collision energies. As we have decided for a collision-energy agnostic setup, we try to avoid features that depend too much on collision energies. On the other side, even in our fragmentation tree model we assume that high intensive peaks are more likely signals than low intensive peaks. It makes sense to use intensities for weighting; that is to put more priority into high intensive peaks than into low intensive peaks. We define two kernels using a radial basis function, namely the *NPP* (node probability product) and RLPP (root loss probability product). And two kernels using the intensity as weighting, namely the *NI* (node intensity) and *RLI* (root loss intensity). These kernels are defined as:

$$NPP(x, x') = \sum_{\substack{u \in N(x), v \in N(x'), \\ u=v}} \frac{1}{4\sigma_\rho^2 \pi} \exp\left(\frac{-(\rho(\iota(u)) - \rho(\iota(v)))^2}{4\sigma_\rho^2}\right) \tag{7.5}$$

$$RLPP(x, x') = \sum_{\substack{u \in N(x), v \in N(x'), \\ r(x)-u=r(x')-v}} \frac{1}{4\sigma_\rho^2 \pi} \exp\left(\frac{-(\rho(\iota(u)) - \rho(\iota(v)))^2}{4\sigma_\rho^2}\right) \tag{7.6}$$

$$NI_\rho(x, x') = \sum_{\substack{u \in N(x), v \in N(x'), \\ u=v}} \rho(\iota(u)) \cdot \rho(\iota(v)) \tag{7.7}$$

$$RLI_\rho(x, x') = \sum_{\substack{u \in N(x), v \in N(x'), \\ r(x)-u=r(x')-v}} = \rho(\iota(u)) \cdot \rho(\iota(v)) \tag{7.8}$$

Here, σ_ρ is a hyperparameter for the standard deviation of intensities and ρ is a transformation we apply on the intensities before multiplying them. In the most simple case this is the identity function. However, this puts too much weight into some very high intensive peaks. Using a square root ($\rho(x) = \sqrt{(x)}$) or logarithm ($\rho(x) = \frac{\log(x)}{\log(0.005)}$) on intensities can be understood as compromise between the binary and the intensity weighted kernels.

The NB, NI, RLB, and RLI kernels were first presented in [196] using the identity function as intensity transformation. NPP was introduced in [58] under the name FIPP (fragment intensity probability product).

We can distinguish three different kind of *losses* in fragmentation trees: joined losses, root losses, and direct losses (which we simply call losses). For each tree t we define the two sets

$$\mathcal{J}_t = \{\, (u, v) \mid u \in N(t),\, v \in N(T[u]),\, u \neq v \,\} \tag{7.9}$$

$$\mathcal{J}_t^\lambda = \{\, u - v \mid (u, v) \in \mathcal{J}_t \,\} \tag{7.10}$$

and call $\mathcal{J}_t$ the set of *joined losses* in t. Remember that $T[u]$ is the induced subtree rooted in node u and $N(T[u])$ is the set of nodes that are descendant to u (including u). A joined loss is the non-empty molecular formula difference between a node and a descendant node in the tree. *Root losses* are a subset of joined losses, because they always start in the root. Direct losses, or simply *losses*, are edges in the tree. So far, we only defined a kernel for root losses. $\text{PPK}_{\text{pairs}}$ was already some kind of loss kernel, but it operates on all peak differences. Here we define the LB (loss binary) and JLB (joined loss binary):

$$LB(x, x') = |\{\lambda(uv) | uv \in E(x)\} \cap \{\lambda(uv) | uv \in E(x')\}| \tag{7.11}$$

$$JLB(x, x') = |\big(\mathcal{J}_x^\lambda \cap \mathcal{J}_{x'}^\lambda| \tag{7.12}$$

With joined losses, root losses, losses, and fragments we have covered all basic elements of a fragmentation tree. The root loss and loss kernels are somewhat redundant, as root losses and losses are subsets of the joined losses. However, it is better to define a kernel for each of them and let the multiple kernel learning procedure decide how to weight each of these features.

In contrast to RLI and NI, it is not clear which peaks should be used to weight the kernels for losses. We can arbitrary decide for the intensity of the first peak of the path, or for the terminal peak. Alternatively, we can use the fold change of the intensities of the first and terminal peak and use a radial basis function to compare these fold changes. This captures information like "a H_2O loss going from a high to a low intensive peak". We define

the LC (loss counter), LI (loss intensity), and $LIPP$ (loss intensity probability product) kernels as:

$$LC(x, x') = \sum_{\substack{uv \in E(x), u'v' \in E(x'), \\ \lambda(u,v) = \lambda(u',v')}} 1 \tag{7.13}$$

$$LI(x, x') = \sum_{\substack{uv \in E(x), u'v' \in E(x'), \\ \lambda(u,v) = \lambda(u',v')}} \rho(\iota(v)) \cdot \rho(\iota(v')) \tag{7.14}$$

$$LIPP(x, x') = \sum_{\substack{uv \in E(x), u'v' \in E(x'), \\ \lambda(u,v) = \lambda(u',v')}} \frac{1}{4\sigma_\rho^2 \pi} \exp\left(\frac{-(log(\iota(u)/\iota(v)) - log(\iota(u')/\iota(v')))^2}{4\sigma_\rho^2}\right) \tag{7.15}$$

7.2.2 Comparing Molecular Formulas

So far, the fragmentation tree kernels are designed very similar to the PPK kernels in [79], although the joined losses and losses could not be modelled without the tree structure. But the real advantage of molecular formulas in contrast to masses is that we can define smoother kernel functions. For masses we can check only for equality. Beside that, masses behave very nonlinear: a mass which is slightly lower or higher than another mass belongs most likely to a completely different molecular formula and molecular structure. But a mass with a specific difference (for example the difference of an H_2O) might belong to a very similar structure. For molecular formulas, however, we can directly compare formulas and quite often a similar formula belongs to a similar structure. An obvious method is to compute the dot product between two molecular formulas. Such a kernel was introduced in [58] under the name *CEC* (chemical element counter). However, this will emphasise frequent elements stronger than rare ones. To correct for this we need additional hyperparameters that downweight elements that occur very frequently (for example: hydrogen). We propose an alternative hyperparameter-free representation for molecular formulas, which is inspired by the way molecular formulas are encoded in PubChem fingerprints: we define a binary molecular fingerprint that encodes for each element if it is contained at least once, two times, four times and so on in the molecular formula. Then we can compute the Tanimoto to compare two fingerprints, or use a radial basis function on top of the Tanimoto. Let f be a molecular formula with $\#(f, X)$ is the amount of atoms of element X in f. Σ is the set of all elements. We then define the molecular formula kernels for two molecular formulas x and x' as:

$$\omega_{dot}(x, x') = \sum_{e \in \Sigma} w(e) \cdot \#(x, e) \cdot \#(x', e) \tag{7.16}$$

$$\omega_{log}(x, x') = \frac{\sum_{e \in \Sigma} log(1 + min(\#(x, e), \#(x', e)))}{\sum_{e \in \Sigma} log(1 + max(\#(x, e), \#(x', e)))} \tag{7.17}$$

$$\omega_{rbf}(x, x') = e^{-\gamma_\omega (2 - \omega_{log}(x, x'))} \tag{7.18}$$

The ω_{dot} kernel is identical to the CEC kernel proposed in [58]. w is the weighting factor for the chemical element e. In [58] we used $w(C) = 1$, $w(H) = 0$, $w(N) = 100$, and

$w(x) = 1\,000$ for any other element x. For the ω_{rbf} kernel, γ_ω is the hyperparameter for the radial basis function.

All three kernels allow for comparing molecular formulas beyond checking for equality. We can now define a kernel that compares pairwise all fragments and root losses in an inexact way:

$$NRBF(x, x') = \sum_{u \in N(x), v \in N(x')} \omega_{rbf}(u, v) \tag{7.19}$$

$$RLRBF(x, x') = \sum_{u \in N(x), v \in N(x')} \omega_{rbf}(r(x) - u, r(x') - v) \tag{7.20}$$

NRBF (node radial basis function) and RLRBF (root loss radial basis function) are smoother variants of the NB and RLB kernels: They also yield high similarity if two nodes and root losses are identical; but for similar molecular formulas and inexact matches they give non-zero similarities. Computing these kernels is computational much more expensive: it is quadratic in the number of nodes in the tree. The NB and RLB kernels, however, can be computed in linear time by using hash maps. The radial basis functions defined above have another disadvantage: they cannot address subformula relationships. According to these kernels, the molecular formulas $C_6H_{12}O_6$ and $C_6H_{13}O_5$ are very similar, as they differ only in two atoms. However, we propose that $C_6H_{10}O_5$ is more similar to $C_6H_{12}O_6$, although it differs by three atoms. But it is a subset of the latter, and the difference even corresponds to a common loss and common chemical reaction. We cannot directly use subset relationship as kernel, because it is obviously not symmetric. But given a set of molecular formulas $\mathcal{M}$, we can define the subset relationship kernel over $\mathcal{M}$ as:

$$\omega_{sub}(x, x') = \sum_{\substack{m \in \mathcal{M} \\ m \leq x \wedge m \leq x'}} w^2(m) \tag{7.21}$$

Again, w is a weighting such that molecular formulas like CH which are contained in almost every molecular formula get a lower weight than molecular formulas like PO_4. We compute $w(f)$ by selecting all trees from the training data that contain f as subset of the root. We now count how many nodes of fragmentation trees from this set are a superset of f. We then set $w(f)$ to the number of trees divided by the average frequency of f as subset in these trees.

But how do we define $\mathcal{M}$? To calculate ω_{sub} in an efficient way we have to choose $\mathcal{M}$ to be a rather small set of molecular formulas; choosing all molecular formulas that occur in our training data (as fragment or loss) is not an option. We therefore suggest the following filtering: Add all molecular formulas that occur as fragment in at least twenty fragmentation trees in increasing order of their mass; remove molecular formulas which differ only in up to four carbon and/or hydrogen atoms from smaller molecular formulas that are already in the set. Remove all molecular formulas f with $w(f) < 0.1$.

We also suggest to add molecular formulas of substructure molecular properties to this set. We can assign molecular formulas to many substructure-based molecular properties. For example, the molecular property '[CX3]=C=O' (ketene) corresponds to the molecular formula C_2O. For other molecular properties like "$*1{\sim}*{\sim}*{\sim}*{\sim}*{\sim}1$" (a ring of five atoms) there is no such unambiguous assignment. For all molecular properties we want to predict and for which we can assign a molecular formula, we add this molecular formula to the set $\mathcal{M}$. Now, $\mathcal{M}$ has an additional interpretation: it represents for a given molecular formula

(e.g. from a measured ion) which molecular properties could potentially match to the ion. For example, an ion with molecular formula $C_6H_{12}O_6$ can never contain the phosphoric acid substructure with molecular formula PO_4.

The resulting kernel is now implicitly transforming any molecular formula f to a binary feature vector v. v_i is one if the ith molecular formula in $\mathcal{M}$ is a subset of f. This binary vector is then weighted according to the weight function w.

We again define kernels that compare fragments and root losses in an inexact way, this time using the subset relationships

$$NSF(x, x') = \sum_{u \in N(x), v \in N(x')} \omega_{sub}(u, v) \tag{7.22}$$

$$RLSF(x, x') = \sum_{u \in N(x), v \in N(x')} \omega_{sub}(r(x) - u, r(x') - v) \tag{7.23}$$

$$JLSF(x, x') = \sum_{(u,v) \in \mathcal{J}_x, (u',v') \in \mathcal{J}_{x'}} \omega_{sub}(u - v, u' - v') \,. \tag{7.24}$$

The NSF (node subformula), RLSF (root loss subformula), and JLSF (joined loss subformula) kernels count how often any molecular formula from $\mathcal{M}$ occur as subset of nodes, root losses, and joined losses in both trees.

We can generalise the JLSF even further and consider not only paths in the tree but arbitrary pairs of nodes u, v with $u \leq v$. We reach:

$$GJLSF(x, x') = \sum_{\substack{u \in N(x), v \in N(x), u' \in N(x'), v' \in N(x') \\ v \leq u \wedge v' \leq u'}} \omega_{sub}(u - v, u' - v') \tag{7.25}$$

The resulting GJLSF (generalised joined loss subformula) kernel will also consider paths which were contained in the original fragmentation graph but were omitted in the fragmentation tree computation. Different to $\text{PPK}_{\text{pairs}}$, it will not consider all pairs of fragments, but just fragments where one fragment is a subset of the other one.

Instead of comparing the elemental composition, we can also compare the ring double-bond equivalent (RDBE) between two formulas (see (2.1)). We calculate the doubled (integral) RDBE value for all nodes, losses and joined losses and calculate a histogram for each of them. For example, the histogram over all node RDBE values contains the distribution of RDBE values for molecular formulas of fragments. Furthermore, we define additional two histograms each for nodes, losses, and joined losses: one calculates the ratio of odd versus even values (remember that odd values denote radicals) and the other calculates the ratio between positive and negative RDBE values (negative RDBE values denote disconnected molecular structures). We end up with 9 histograms. The *RDBE* kernel is now normalising and concatenating these histograms into a vector and computes the dot product. Effectively, the RDBE kernel compares the distribution of RDBE values between two trees. We omit the tedious mathematical formula for this kernel.

7.2.3 Kernels for Paths and Subtrees

Kernel methods allow us to transform the input into a high dimensional (or even infinite dimensional) feature space. Thus, we can define kernels that count every path or subtree, while representing them in a feature vector would be in-feasible due to the exponential

number of possible subtrees. The CPC (common path counter) kernel counts the number of paths that are shared between two trees. We define

$$CPC(x, x') = \sum_{u \in N(x), v \in N(x')} C[u, v]\,, \tag{7.26}$$

where $C[u, v]$ is the number of common paths that occur in both x and x' and start in u and v. We compute it with dynamic programming using the recurrence

$$C[u, v] = \sum_{\substack{w \in C(u), w' \in C(v) \\ \lambda(u,w) = \lambda(v,w')}} C[w, w'] + 1\,. \tag{7.27}$$

Here, we implicitly assume that the empty sum is zero. The reasoning behind the recurrence is simple: we can extend each path that starts in a child node w and w' by adding the edges uw and vw', as long as the label of both edges is equal. Furthermore, we can start a new path uw and vw'; therefore, we have to increment the count.

Similarly, we can count the number of common subtrees. We define the CSC (common subtree counter) kernel as:

$$CSC(x, x') = \sum_{u \in N(x), v \in N(x')} C'[u, v] \tag{7.28}$$

Here, C' is the number of shared subtrees that start in u and v. We compute it with dynamic programming using the recurrence

$$C'(u, v) = \prod_{\substack{w \in C(u), w' \in C(v) \\ \lambda(u,w) = \lambda(v,w')}} (C'[u, v] + 2) - 1\,. \tag{7.29}$$

We implicitly assume that the empty product is one. For every outgoing edge of u and v with equal labelling, we can extend the subtrees by connecting the outgoing edge to the root of the subtree. Alternatively, we can use the edge itself as subtree. These subtrees starting from different outgoing edges can now be freely combined with each other; we can also decide to not use a certain outgoing edge and just combine the trees from other outgoing edges. Finally, we have to remove the empty tree from the resulting set of subtrees (because the empty tree is not starting in u and v); thus we subtract one.

The CSC kernel has the disadvantage of being *spiked*; this means that entries on the main diagonal are much larger than other entries in the kernel matrix. A tree has an exponential number of subtrees, but even two similar trees usually have only a few small subtrees in common. Another example for a spiked kernel is the Gaussian radial basis function with large gamma. Prediction methods on spiked kernels behave like a nearest-neighbour classifier: The prediction is mainly influenced by the closest training examples and the method tend to overfit on the training data [40]. To counter this problem, we can introduce a weight decay $\gamma \in (0, 1]$ that exponentially downweights large subtrees. Using such a weight decay on subtree kernels was suggested by [40]; however, they used

a different definition for subtrees and, hence, a slightly different recurrence. We define a modified variant of CSC with weight decay as:

$$CSC_\gamma(x, x') = \sum_{u \in N(x), v \in N(x')} C'_\gamma[u, v] - LC(x, x') \tag{7.30}$$

$$C'_\gamma[u, v] = \prod_{\substack{w \in C(u), w' \in C(v) \\ \lambda(u,w)=\lambda(v,w')}} (\gamma \cdot C'_\gamma[u, v] + 2) - 1 \tag{7.31}$$

Clearly, the recurrence is almost the same; but when extending subtrees from the child, their count is multiplied with γ. Thus, large subtrees, which are unlikely to be shared between different trees, get smaller weight. Furthermore, we removed the number of common edges (and, thus, subtrees of length 1); otherwise, these common edges would dominate the similarity for small γ.

We also define some specialised path kernels: *CP2* (common path of length two) is counting only paths of length two. *LPC* (loss pair counter) counts pairs of losses which are in a path from root to any leaf. It is similar to a quadratic loss counter kernel, but it counts loss interactions where the latter loss starts in a node that is descendant to the starting node of the first loss. Finally, *MLIP* (maximum loss in path) counts for each molecular formula the maximum frequency of this formula in any path of the tree. We omit the trivial mathematical descriptions of these kernels.

Algorithm 2 Recursive algorithm for calculating the number of paths from root to all leafs that either contain a certain molecular formula as subset or loose this molecular formula within a single loss.

```
1: function COUNTPATHS(formula, node)
2:     if node is a leaf then
3:         if node contains formula as subset then
4:             Return 1
5:         else
6:             Return 0
7:         end if
8:     else
9:         count ← 0
10:        for each child of node do
11:            if loss between node and child contains formula as subset then
12:                count ← count+ COUNTPATHS(∅, child)
13:            else
14:                count ← count+ COUNTPATHS(formula, child)
15:            end if
16:        end for
17:        Return count
18:    end if
19: end function
```

7.2.4 Fragment-Loss Interaction Kernels

All kernels defined above either compare losses or nodes between two trees. Intuitively, kernels that compare losses *and* nodes should have an improved performance. This could already been shown in Shen *et al.* [196], where the product of node intensity and root loss intensity got the largest weight in their multiple kernel learning setup. Multiplying existing kernels is a simple approach, but we argue that it is more effective to count local interactions between losses and nodes: For example, a loss that ends in a certain node. We have to use inexact matching of either nodes or losses here, otherwise, we would just count pairs of nodes. We define the NLI (node loss interaction) kernel as:

$$NLI(x, x') = \sum_{\substack{(u,v)\in\mathcal{J}_x \\ (u',v')\in\mathcal{J}_{x'} \\ \lambda(u,v)=\lambda(u',v')}} \omega_{sub}(v, v') \tag{7.32}$$

For every joined loss in x ending in a node v and every equal joined loss in x' ending in a node v' we compare both nodes using the subset relationship ω_{sub}. The corresponding feature space consists of pairs (f, l) where f is a molecular formula from $\mathcal{M}$ and l is a joined loss. The feature vector counts how often we observe a joined loss $l = u - v$ ending in a fragment v that is a superset of f. We could also define a kernel that matches joined losses inexactly and nodes exactly; but this would result in a very sparse kernel which is probably too similar to the node binary kernel.

Another possibility to connect node and loss information is to search for molecular structures that do not fragment. For example, the bond between a phosphor and a sulphur atom seems to be very stable, because it does not fragment in any of our training examples. In the fragmentation tree, we will see that the molecular formula PS is never divided. We see nodes and losses containing PS, but none of the nodes or losses contains only one of both atoms. We define a kernel that counts for each molecular formula in $\mathcal{M}$, in how many paths this formula either cleaves off in a loss or is contained in the leaf node. We can do this using the recursive algorithm in 7.2.4. The SLL (substructure in losses and leafs) kernel is then calculated as

$$SLL(x, x') = \sum_{f\in\mathcal{M}} \text{COUNTPATHS}(f, r(x)) \cdot \text{COUNTPATHS}(f, r(x')) \tag{7.33}$$

with COUNTPATHS is the function defined in algorithm 7.2.4.

7.2.5 Tree Alignment Kernel

Fragmentation tree alignments were introduced for comparing and clustering mass spectra [165]. They correlate better with chemical similarity (Tanimoto similarity) than various spectral alignment algorithms [165]. Therefore, fragmentation tree alignment are an interesting candidate for a kernel method.

Although the underlying problem is shown to be NP–hard, there is a fixed parameter tractable algorithm that solves the problem fast in practice for trees with limited out-degree [90]. However, the tree alignment is not a positive definite function. Rasche *et al.* [165] have proposed to not directly compare alignment scores but instead build feature vectors $\mathbf{a}$ with $\mathbf{a}_i$ is the normalised alignment score to the i-th fragmentation tree in a reference library of fragmentation trees. The Pearson correlation between such feature

vectors correlates better with the Tanimoto similarity of the measured molecules than the raw alignment score.

In contrast to the tree alignment score, the Pearson correlation is a positive definite function. The underlying feature space consists of the normalised, centred vectors of alignment scores to the fragmentation trees in the training set. As we can transform every scoring scheme into a kernel by computing its covariance matrix, such kernels are also called *pseudo kernels*. We name the resulting kernel *TREEALIGN*. Note that this kernel is computational very expensive: First, we have to solve an NP–hard problem. This is fast in practice for most trees, but might be extremely slow for some trees with high out-degree. For predicting a molecular fingerprint from a fragmentation tree, we have to compute the alignment between this tree and all trees in the training set. Afterwards, we have to compute the Pearson correlation between the resulting vector of alignment scores against all vectors in the training set. Computing the Pearson correlation between two vectors (without considering the time for aligning the trees) can be done in $O(n)$ with n is the number of training points. For large training sets, this is more expensive than all other kernels defined above, which can be computed in $O(m^2)$ or $O(dm)$ where m is the number of nodes in the tree (and, thus, $m << n$) and d is the depth of the tree. A possible solution for this problem is to not align against all trees in the training set but to choose a small subset of interesting trees. However, it is still an open issue how to select this subset and how many trees should be chosen.

7.3 Multiple Kernel Learning

Instead of choosing only one good-performing kernel, we use a linear combination of the above defined kernels, using weights obtained by centred kernel alignment [42]. This method requires that kernels are centred. We observe that normalising each kernel before centring improves performance. Given any kernel matrix $\mathbf{K}$, the normalised and centred variants of this kernel matrix are computed as:

$$[\mathbf{K}^n]_{i,j} = \frac{\mathbf{K}_{i,j}}{\sqrt{\mathbf{K}_{i,i} \cdot \mathbf{K}_{j,j}}} \tag{7.34}$$

$$[\mathbf{K}^c]_{i,j} = \mathbf{K}^n_{i,j} - \frac{2}{m}\sum_{i=1}^{m}\mathbf{K}^n_{i,j} + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\mathbf{K}^n_{i,j} \tag{7.35}$$

The normalisation of the kernel matrix (eq. (7.34)) implicitly normalises the feature space such that the inner product between a data point to itself is 1. The centring of the kernel matrix (eq. (7.35)) subtracts for each matrix entry the row and column mean and then adds the matrix mean.

Note that normalising the kernel matrix is very different from normalising features in the feature space. For example, if we normalise the LC (loss counter) kernel, we end up with a histogram-like vector that can be interpreted as a discrete probability distribution of losses (in contrast to a real probability distribution, the vector does not sum up to 1 but to another constant). The inner product between two such vectors calculates the similarity of both distributions. Normalising features, however, will downweight very frequent losses like H_2O or C_2H_2 such that they do not dominate over rare (and maybe more important) losses like C_6H_6. Thus, for some kernels it might be useful to first normalise the feature

space and afterwards normalise the kernel matrix. Obviously, normalisation and centring has to be done in the same manner on test data, too.

The basic idea behind centred alignment is to find a linear combination of kernels such that the resulting kernel is similar to the ideal kernel [43]. But what is the ideal kernel? We can think of the best possible kernel as a kernel over the output we want to predict. In our case this is YY^T, where Y is the fingerprint matrix. Instead of computing kernel weights for each molecular property separately, we optimise weights over all molecular properties. This works like a regulariser and prevents overfitting [197].

The similarity between two kernels can be measured with the Frobenius norm $\langle \cdot, \cdot \rangle_F$; this norm is obtained by first building a vector by stacking all columns of the matrix and then computing the dot product between these vectors. We enforce that linear combinations of kernels are strictly positive, because such a kernel combination can be explained as concatenating the weighted transformed feature vectors of each kernel. The underlying optimisation problem is

$$\begin{aligned} \underset{\mu}{\text{maximise}} \quad & \frac{\langle \mathbf{K}_\mu, \mathbf{Y}\mathbf{Y}^T \rangle_F}{\sqrt{\langle \mathbf{K}_\mu, \mathbf{K}_\mu \rangle_F}} \\ \text{subject to} \quad & \mu_q \geq 0 \quad \text{for } q = 1, \ldots, l \\ & ||\mu||^2 = 1 \,, \end{aligned} \tag{7.36}$$

where $\mathbf{K}_\mu$ is the linear combination of l different, centred kernels $\mathbf{K}_1^c, \ldots, \mathbf{K}_l^c$, and μ is a vector of kernel weights. The linear combination of the kernel matrices is defined as $\mathbf{K}_\mu = \sum_{q=1}^{l} \mu_q \mathbf{K}_q^c$. Analogously, we will write K_μ for the kernel function we obtain by summing up all normalized and centred kernels with the corresponding weights in μ. We solve this quadratic problem using the quadratic solver in the Gurobi mathematical programming framework.

Shen *et al.* [197] show that the centred kernel alignment can be made more robust against noise in the target kernel by adding upperbounds for the kernel weights. This method is called *ALIGNF+*. This results into less sparse kernel weights. If the upperbounds are small enough, the resulting kernel weights become uniform [197]. We use ALIGNF+ whenever we obtain too sparse kernel weights from ALIGNF (as it is the case for negative ion mode spectra).

7.4 Extending the Set of Molecular Properties

Although we would like to predict molecular structures, this problem is incredible hard. Thus, we solve a similar but much easier problem: Instead of predicting the structure, we predict molecular fingerprints. The molecular fingerprint is a binary encoding of a molecule. Thus, we can use classification methods like support vector machines for predicting the fingerprint.

In [79, 196] the molecular fingerprints were computed with OpenBabel [150]. However, we noticed that the SMARTS query tool in OpenBabel uses heuristics and do not find the exact solution in some cases [58]. Therefore, we switched to the chemistry development kit (CDK) version 2.0 to compute the molecular fingerprints [238]. Additionally to the FP2 (55 bits), FP3 (307 bits), and MACCS fingerprint (166) from OpenBabel (these are also implemented in CDK), we add PubChem fingerprint (881 bits), Klekota-Roth

fingerprint [109] (4860 bits), and extended connectivity fingerprints (ECFP) by Rogers and Hahn [174].

The extended connectivity fingerprints are computed using the Morgan algorithm [137] and have variable length. Each molecular property of the ECFP fingerprint is represented by a 32 bit integer hash value and, thus, 2^{32} different properties can be described. However, the ECFP fingerprint is usually stored in a fixed length binary format. For example, to map the fingerprint to a 1 024 bit vector, a second hash function is applied on the molecular properties that maps each integer to a number between 1 and 1024. Although this is repeatedly done, even in machine learning applications, we argue that this will result in significantly decreased performance: One position in a fingerprint might belong to very different molecular properties; this is called *collision*. Hash functions are designed in a way that such collisions occur very randomly (in the sense that very different substructures map to the same hash value). However, such an arbitrary mapping is exactly the opposite of what we want in machine learning applications. Instead of using hashing we propose a different approach: We first compute the full extended connectivity fingerprint for all compounds in our training set. We count for each molecular property how frequently it occurs in the set of training compounds. We only keep molecular properties that occur in at least 50 different structures. 1 324 molecular properties satisfy this condition. We define a fixed length fingerprint containing only those molecular properties. Note that each molecular property is still a 32 bit hash value and, thus, collisions are still possible but very unlikely.

The same approach could also be used to integrate path fingerprints or shortest path fingerprints. However, we decided for ECFP fingerprints because they show excellent performance in many machine learning applications that use them as input features.

In total, 7 593 molecular properties are available for learning. However, many molecular properties are defined multiple times in different fingerprints. Other molecular properties do not occur in our training data. Thus, we remove all molecular properties which occur less than 25 times in our training data and all redundant molecular properties. The resulting set of molecular properties is dependent on our training dataset.

7.5 Predicting Probabilistic Fingerprints

We have defined the kernel function of CSI:FingerID as a linear combination of fragmentation tree and spectrum kernels. The output labels are the binary fingerprint of a molecular structure. We can now train kernel support vector machines using a training set $\mathcal{T}$ of n reference spectra. We use the Java implementation of LibSVM for training [38]. We choose for each predictor the hyperparameter c from a range of values (2^{-2}, 2^{-1},$2^{0}, \ldots, 2^{7}$) via cross-validation such that the F1 score is maximised in the test batch.

The outcome of the training is a set of predictors, one for each molecular property. A predictor consists of a bias b and a vector of coefficients $\alpha = (\alpha_1, \ldots, \alpha_n)$, one for each training point. This vector should be relatively sparse; most of its coefficients are zero. The non-zero entries are the support vectors. To predict the molecular fingerprint of an unknown spectrum and fragmentation tree x', we have to compute for each spectrum and fragmentation tree kernel K the kernel vector $\mathbf{k} = (K(x', x_1), \ldots, K(x', x_n))$ by comparing the spectrum and its fragmentation tree against all spectra and fragmentation trees in the training set. The kernel vectors are then centred and normalised in the same manner as the kernel matrix we used for training. We obtain the final kernel vector $\mathbf{k}^{\mu}$ by multiplying

each kernel vector with the kernel weight and sum them up. For each molecular property we evaluate the predictor function f as

$$f(\mathbf{x}') = \sum_{i=1}^{n} \alpha_i y_i \mathbf{k}_i^{\mu} - b\,. \tag{7.37}$$

The sign of the decision function (7.37) is the binary prediction of the molecular property predictor. The molecular property is predicted as present if $f(\mathbf{x}') \geq 0$.

By defining the problem as binary classification, we enforce the predictor to decide between two possibilities: a molecular property is either present or absent. However, from the hinge loss (the loss function used in support vector machines) it is apparent that decision values between -1 and 1 are within the margin and, therefore, are not as reliable as values outside the margin. In general we can say: decision values close to zero are less reliable than decision values far away.

Platt scaling allows us to transform the output of a predictor function into a probability distribution over classes: We ask for a classification that not only gives an answer, but also a degree of certainty about the answer. Formally, we want to estimate the posterior probability $\mathbb{P}(y' = 1 \mid \mathbf{x}')$ that our unknown molecule has a particular property.

Platt [155] proposed to use a sigmoid function as an approximation of posterior probabilities:

$$\mathbb{P}(y' = 1 \mid \mathbf{x}') \approx P_{A,B}(f) \equiv \frac{1}{1 + \exp(Af + B)}\,, \tag{7.38}$$

where $f = f(\mathbf{x}')$ is the decision value. Given training examples $(x_i, y_i) \in \mathcal{T}$ with binary labels, we search for parameters A^*, B^* that maximise the following likelihood [155]:

$$\begin{aligned} &\underset{A,B}{\text{minimise}} \quad F(z) = -\sum_{i=1}^{n} \big(t_i \log p_i + (1 - t_i) \log(1 - p_i)\big), \\ &\text{where} \quad p_i := \mathbb{P}_{A,B}(f_i) \quad \text{and} \quad t_i := \begin{cases} \frac{N_+ + 1}{N_+ + 2} & \text{if} \quad y_i = +1 \\ \frac{1}{N_- + 2} & \text{if} \quad y_i = -1 \end{cases} \quad \text{for } i = 1, \ldots, m. \end{aligned} \tag{7.39}$$

Here, N_+ is the number of examples with positive labels and N_- the number of examples with negative labels. To solve this optimisation problem, Lin *et al.* [120] proposed to use a Newton method with backtracking line search. Again, we used the implementation in LibSVM for fitting the sigmoid function [38].

See Fig. 7.1 for an example of the empirical distribution of conditional probabilities $\mathbb{P}(y' = 1 \mid \mathbf{x}')$ and the fitted sigmoid function for the predictor of the molecular property "phenol group".

7.6 Maximum Likelihood Estimator for Probabilistic Fingerprints

After predicting the probabilistic molecular fingerprint for an unknown molecule from MS data, we can use it to search in a structure database: We first select all molecular structures with the correct molecular formula; remember, that at this point we already identified the molecular formula using SIRIUS. We now compute molecular fingerprints for each selected candidate structure. We then compare the predicted fingerprint with each

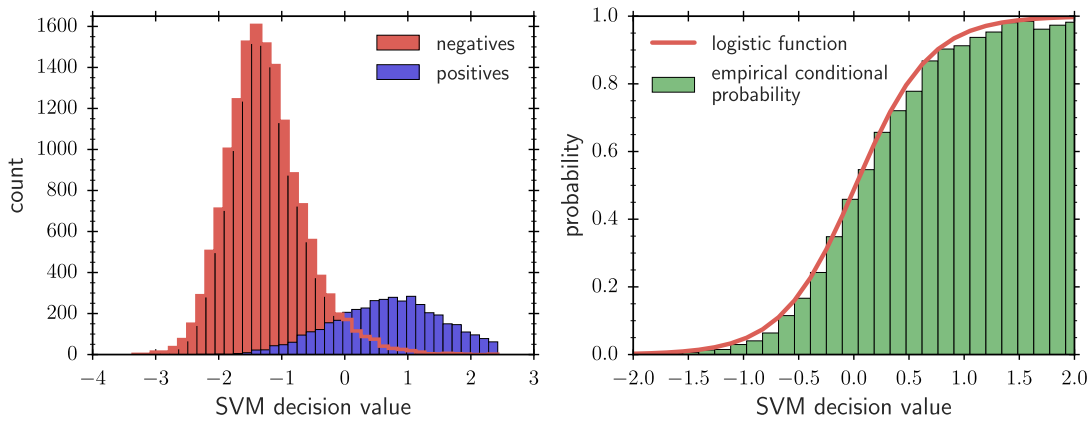


Figure 7.1: The SVM prediction for phenol groups ("c[OH]") on the large CSI:FingerID 1.1 training dataset on positive ion mode spectra. Left: The histogram shows the SVM predictions obtained in cross-validation for positive examples (compounds containing a phenol group) in blue and negative examples (compounds without phenol group) in red. Right: For each histogram bin we have drawn the conditional probability for observing a positive example (green). The logistic function estimated as proposed by Platt [155] is drawn in red.

candidate fingerprint and rank the molecular structures according to their match with the prediction.

But how to compare two fingerprints? Probably the most obvious approach for comparing two binary strings is the Hamming similarity that counts how many positions in both strings are identical. We will call this naïve approach the *unit score*. However, some molecular properties can be predicted with high accuracy while other predictions are error-prone. Heinonen *et al.* [79] suggest to weight each position by the accuracy the predictor obtained in cross validation. In the following we describe four additional scoring methods for comparing fingerprints which we have presented in [58].

In the following we describe a binary fingerprint as set of molecular properties: Let $1, \ldots, m$ be the molecular property indices. A *fingerprint* is a subset $F \subseteq \{1, \ldots, m\}$. A fingerprint scoring function is a function $F \times F \to \mathbb{R}$ which assigns high scores to similar and low scores to dissimilar fingerprints. We will denote the predicted fingerprint as F' and the database fingerprint as F. Note that the prediction might contain errors and, thus, even for the correct structure F and F' will not be identical.

Tanimoto Coefficient Using the Tanimoto coefficient for scoring is well established in Cheminformatics. This scoring is also used in [114]. The scoring function is defined as:

$$\sigma_{Tanimoto}(F', F) = \frac{|F' \cap F|}{|F' \cup F|} \tag{7.40}$$

Maximum Likelihood Scoring The scoring by Heinonen *et al.* [79] uses the predictor accuracy for weighting molecular properties. However, because the distribution of molecular properties is very imbalanced, the accuracy of a predictor on positive examples might be extremely different from its accuracy on negative examples. In other words, predicting the presence of a certain substructure might be much easier or harder than

predicting its absence. Thus, we suggest a maximum likelihood scoring that differentiates these cases. The scoring function is defined as:

$$\sigma_{ml}(F', F) = \underbrace{\sum_{i \in F' \cap F} \log sens_i}_{\text{true positives}} + \underbrace{\sum_{i \in F' \setminus F} \log(1 - spec_i)}_{\text{false positives}} + \underbrace{\sum_{i \in F \setminus F'} \log(1 - sens_i)}_{\text{false negatives}} + \underbrace{\sum_{i \notin F' \cup F} \log spec_i}_{\text{true negatives}} \tag{7.41}$$

There are four possible outcomes in prediction tasks: true positives, false positives, false negatives and true negatives. The maximum likelihood scoring effectively counts how often each of these cases happens in the cross-validation and uses this for weighting. $sens_i$ stands for the sensitivity (true positive rate) of the predictor for molecular property i. Note that $1 - sens_i$ is equivalent to the false negative rate. Analogously, $spec_i$ is the specificity (true negative rate) and $1 - spec_i$ the false-positive rate. As usual, we take the logarithm of the probabilities such that we can work with sums instead of products. Note, that we implicitly assume here that these probabilities are independent. This is obviously wrong. For example, a molecular property that encodes for a phenol ring ("c[OH]") and a molecular property that encodes for an aromatic ring ("c") are obviously correlated to each other. When the aromatic ring predictor makes a false negative prediction, also the phenol ring predictor will be false negative. Therefore, Ludwig *et al.* [123] developed a new maximum likelihood scoring to get rid of the independence assumption.

Posterior Probability Scoring All scorings defined above use binary fingerprints. Incorporating posterior probabilities is straightforward as long as we assume independence. Let p_i be the estimated posterior probability for the molecular property i. Then we define the scoring function

$$\sigma_{posterior}(F', F) = \sum_{i \in F} \log p_i + \sum_{i \notin F} \log(1 - p_i)\,. \tag{7.42}$$

Modified Posterior Probability Finally, we try to define a scoring that combines both, the maximum likelihood score and the posterior probability score. The obvious way for doing so is to add both scorings, using some weighting. This work was mainly done by my colleague Marvin Meusel. While experimenting with several weightings, he came up with a scoring which beats all other scorings in a wide range of evaluations. Unfortunately, we still do not have any statistical explanation for this scoring. The *modified Platt* scoring is defined as:

$$\sigma_{mp}(F', F) = \underbrace{\sum_{i \in F' \cap F} \left(\tfrac{3}{4} \log p_i + \tfrac{1}{4} \log(1 - sens_i)\right)}_{\text{true positives}} + \underbrace{\sum_{i \in F' \setminus F} \tfrac{3}{4} \log(1 - p_i)}_{\text{false positives}} + \underbrace{\sum_{i \in F \setminus F'} \tfrac{3}{4} \log p_i}_{\text{false negatives}} + \underbrace{\sum_{i \notin F' \cup F} \left(\tfrac{3}{4} \log(1 - p_i) + \tfrac{1}{4} \log(1 - spec_i)\right)}_{\text{true negatives}} \tag{7.43}$$

Smoothing and Filtering To avoid numerical instabilities, we add 0.5 pseudocounts to true positives, false positives, true negatives, and false negatives before estimating the above measures for any predictor of a molecular property. Similarly, we apply Laplace smoothing (additive smoothing) to Platt probabilities, and replace probability p_i by $\frac{p_i+\alpha}{1+2\alpha}$ for some small α; here, we choose $\alpha = 1/n$ where n is the size of the training dataset.

The weighting in the maximum likelihood scoring as well as the posterior probability both downweight molecular properties which are hard or impossible to predict. However, the statistics for maximum likelihood and posterior probability come from the training dataset (although in a cross-validation setup). When the test data is distributed very different from our training data, the estimated sensitivity, specificity, and posterior probabilities might be no longer accurate. This makes the resulting scoring noisy. To make the scoring more robust, we suggest to omit molecular properties with very low F1 score. Heinonen *et al.* [79] found that such a filter does not improve the scoring. However, they did not evaluate on any independent dataset. We define a cutoff of 0.25 for the F1 and omit all molecular properties with an F1 below 0.25 in cross-validation.

7.7 CSI:FingerID - Searching Spectra in Structure Databases

Before evaluating the method, we will give a short summarise of the workflow. See Fig. 7.2 for a graphical description. We divide the workflow in three phases: learning, prediction, and scoring.

Learning Phase In the learning phase we train the prediction model using a training dataset, which is a library of reference spectra with known structures. We compute fragmentation trees for each spectrum using SIRIUS. For each kernel we compute a kernel matrix with $\mathbf{K}_{i,j}$ is the similarity of the i-th training example to the j-th training example. All these matrices are normalised and centred. We compute the molecular fingerprint for each structure. From the matrix of fingerprints $\mathbf{Y}$ we compute the Gram matrix $\mathbf{Y}\mathbf{Y}^T$ which we will use to learn a linear combination of kernels using the centred kernel alignment method [43]. The output is a list of weights for each kernel. We weight each kernel matrix according to this weight and sum them up. The resulting kernel matrix is then used to train kernel support vector machines; one machine for every molecular property. Every kernel support vector machine is a bias value and a list of support vectors. We optimise the hyperparameter c for the support vector machine as well as the hyperparameter A and B for the logistic function via ten-fold cross-validation.

Note, that training has to be done only once. Afterwards, we store the support vector machines and the parameters of the logistic function for every molecular property in a text file. This is our predictor model.

Prediction Phase In the prediction phase we predict the molecular fingerprint of an unknown spectrum. We start again with computing fragmentation trees using SIRIUS. At this point we have the problem that SIRIUS will sometimes rank the wrong molecular formula on top. If there are several top ranking fragmentation trees with similar score, we repeat the prediction phase and scoring phase for each of them.

For each kernel we now compute a kernel vector with $\mathbf{k}_i$ is the similarity of the test spectrum/FT against the i-th training spectrum/FT. We normalise and centre the vector, weight them according to the kernel weights, and sum them up. For each

molecular property we multiply the corresponding support vector coefficients with the kernel similarity and sum them up. We add the bias and feed the value into the logistic function to obtain a probability value for the molecular property. We end up with a vector of probabilities, the probabilistic molecular fingerprint.

Scoring Phase In the scoring phase we search a molecular structure database for candidate structures with the same molecular formula as the test fragmentation tree. We then retrieve the molecular fingerprints for each of these candidate structures and calculate a similarity score between the predicted probabilistic fingerprint and the candidate fingerprint, using one of the scorings defined in Section 7.6. We order the candidate structures according to the score and report the ranked list to the user.

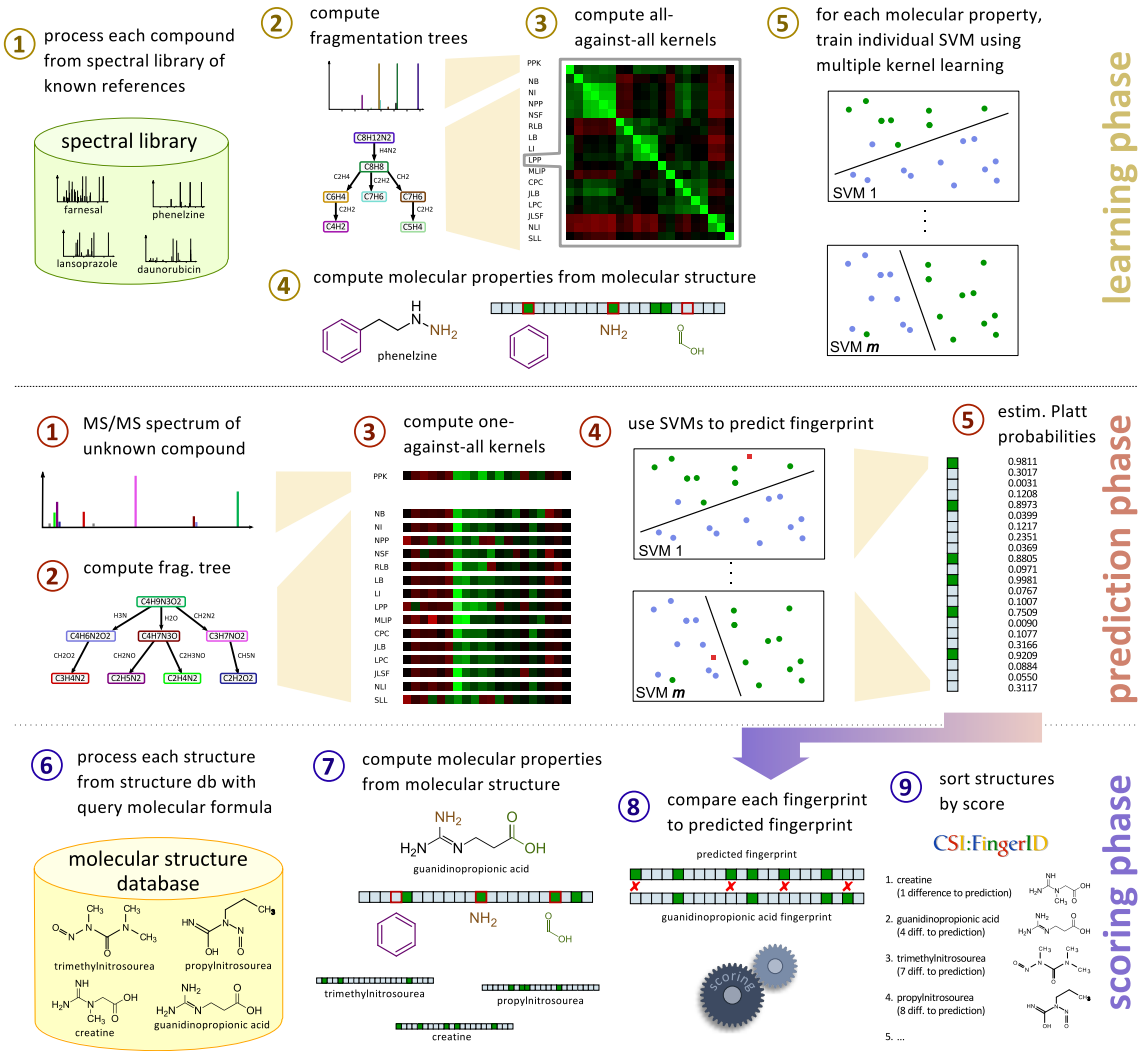


Figure 7.2: Workflow of CSI:FingerID. During the *learning phase*, we use MS^2 reference data to train a set of predictors for molecular properties (the fingerprint). In the *prediction phase* we use MS^2 data of an unknown compound to find a fragmentation tree, and to predict the fingerprint of the unknown. In the *scoring phase* we compare the predicted fingerprint of the unknown to fingerprints of molecular structures in a structure database, searching for a best match.

7.8 Evaluation of Kernels and Multiple Kernel Learning

In this section we will evaluate the kernels defined in Section 7.2. We are interested in how good each kernel performs separately, and if multiple kernel learning improves generalisation and prediction quality. Because learning support vector machines for each kernel is very time consuming, we will do the evaluation on a smaller training dataset: We choose 8 027 unique structures from GNPS and MassBank for training the method. For evaluation, we will use a set of similar size: 8 477 unique structures from Agilent 2.0 and NIST. Both, training and evaluation set are structural disjoint. Instead of predicting all fingerprints, we restrict the training to a subset of 580 molecular properties. We only include molecular properties that occur at least 50 times in the training set and the evaluation set; furthermore, we excluded ECFP fingerprints. Choosing molecular properties that occur frequently in training and evaluation set allows us to do reasonable statistics for every predicted molecular property. To avoid that most of the molecular properties are frequent properties, we exclude all properties that occur more than 1 000 times in the training set. Such properties often do not discriminate molecules very well, especially within one candidate list with the same molecular formula. Furthermore, it is difficult to obtain one statistical measure for frequent and rare molecular properties: the accuracy is a good measure for frequent properties, while F1 score is better suited for evaluating rare properties.

For each kernel we now train 580 support vector machines; one for every molecular property. We estimate the hyperparameter C by 5-fold cross-validation, allowing values from $2^0, 2^1, \ldots, 2^6$. We then do statistics on the cross-validation results: note, that we do one cross-validation for statistics and choosing the hyperparameter. This speeds up the evaluation but might result in some overoptimistic evaluation. However, we noticed that the parameter C has only small impact on the results. In general: less regularisation seems to improve the results. This indicates some underfitting behaviour. We are primarily interested in the F1 score of each SVM.

We then predict molecular properties on the independent set and compare the F1 scores. We split the independent data into two subsets: the "independent < 0.6" dataset contains all instances where the maximum Tanimoto to a compound in our training set is below 0.6 (using the 580 molecular properties to compute the Tanimoto). These instances should be very hard to predict, because they belong to structures very dissimilar to the training data. All instances that do not belong to the "independent < 0.6" set belong to the "independent ≥ 0.6" set.

Finally, we use the centred kernel alignment to find a linear combination of kernels with optimal alignment. We use all molecular properties (not only 580) to build the target alignment matrix. For comparison, we also check if the kernel weights we learned for CSI:FingerID 1.1 on the larger dataset with 19 118 structures improve the results. We name the centred kernel alignment trained on the small dataset ALIGNF^1 and the alignment on the large dataset ALIGNF^2. The TREEALIGN and NRBF kernels are omitted in ALIGNF^2 because they are also omitted in CSI:FingerID 1.1 due to their high computing time. Furthermore, PPK_{peak} and PPK_{diff} are uniformly combined (such that both get the same weight). Thus, ALIGNF^2 is using exactly the same kernels with the same weights as they are used by CSI:FingerID 1.1. We also evaluate the regularised variant ALIGNF+ [197] using an upperbound such that no kernel gets larger weight than 0.0822.

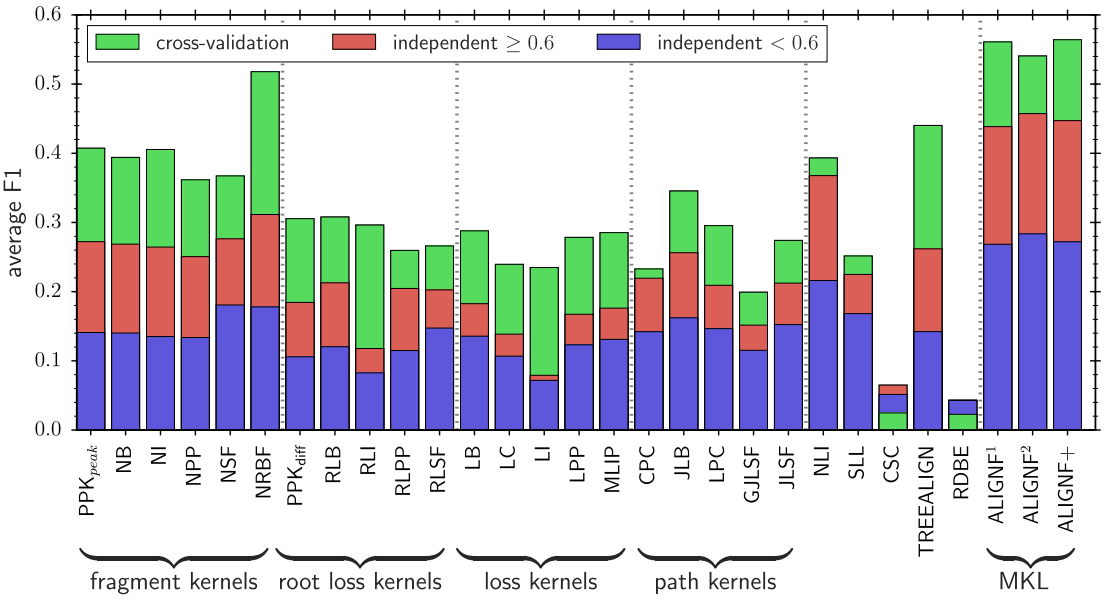


Figure 7.3: Mean F1 score of each individual kernel evaluated on training data via cross-validation (green), on "independent ≥ 0.6" (red), and on "independent < 0.6" (blue). Mean F1 of the centred kernel alignments is also given. ALIGNF^1 is trained on the small dataset with 8 027 structures and ALIGNF^2 is trained on a large dataset with 19 118 structures. Unlike all other kernels, for CSC and RDBE the F1 of the cross-validation data is smaller than the F1 of independent data

Some kernels have hyperparameters which we do not want to evaluate in detail. We use the same hyperparameters for PPK kernel that were used in [79]; namely $\sigma_\mu^2 = 10^{-5}$ and $\sigma_\iota^2 = 10^5$. For the LI, NI, RLI, and LPP kernels we use logarithmised intensities. For NPP and RLPP we used the square root over intensities. We found that these variants perform slightly better than other intensity transformations. The CSC kernel here is not using weight decay. However, we also evaluated the variant with weight decay and found that it works better than CSC but still worse than any other kernel (except RDBE).

Results To simplify evaluation, we calculate for every kernel the average F1 over all 580 molecular properties. If a kernel has no F1 for a molecular property, (for example, because recall and precision are zero) it is counted as zero. We then compare the mean F1 between cross-validation results as well as for the "independent ≥ 0.6" and "independent < 0.6" datasets. See Fig. 7.3 for the mean F1 of all kernels on all three datasets.

We grouped the kernels into the following categories: Fragment kernels, which are all kernels that compare peaks or fragments, including $\mathrm{PPK}_{\mathrm{peak}}$, NB, NI, NPP, and NSF. Root loss kernels refer to kernels that compare root losses. These are $\mathrm{PPK}_{\mathrm{diff}}$, RLB, RLI, RLNSF and RLPP. Loss kernels compare losses (or edges) in the trees. LB, LC, LI, LPP and MLIP are loss kernels. We omitted $\mathrm{PPK}_{\mathrm{peakdiff}}$ because it has shown bad performance in [79]. Path kernels refer to kernels that compare paths or joined losses, namely: CPC, JLB, LPC, GJLB, and JLNSF. The kernels SLL, CSC, TREEALIGN, and RDBE are not put into any category. Finally, MKL refers to the linear combinations of kernels.

All three linear combinations of kernels show similar performance; ALIGNF^1 and ALIGNF are better on cross-validation data with an F1 of 0.56, while ALIGNF^2 is

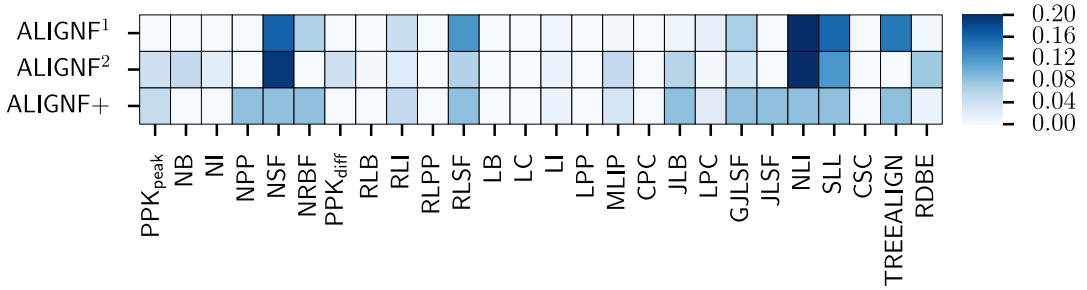


Figure 7.4: Kernel weights learned by centred kernel alignment. ALIGNF^1 is the centred kernel alignment on the training dataset with 4 477 structures. ALIGNF^2 is trained on 19 118 structures. For ALIGNF^2, $\mathrm{PPK}_{\mathrm{peak}}$ and $\mathrm{PPK}_{\mathrm{diff}}$ are uniformly combined with same weight and TREEALIGN and NRBF are omitted. ALIGNF+ is the regularised centred kernel alignment with upperbound of 0.0822 % on kernel weights.

slightly better on independent data with F1 is 0.46 on "independent $\geq$ 0.6" and 0.28 on "independent $<$ 0.6".

For the cross-validation set, the best performing individual kernel is NRBF with an F1 of 0.52. Runner-up is the TREEALIGN kernel with an F1 of 0.44 and the $\mathrm{PPK}_{\mathrm{peak}}$ with an F1 of 0.41. For the independent data, however, the NLI outperforms all other individual kernels with an F1 of 0.37 for "independent $\geq$ 0.6" and 0.22 for "independent $<$ 0.6". The NSF and RLSF kernels outperform their binary variants (NB, RLB) on independent data, but show worse performance in cross-validation. The uniform combination of $\mathrm{PPK}_{\mathrm{peak}}$ and $\mathrm{PPK}_{\mathrm{diff}}$ has an F1 of 0.42 cross validation, 0.36 in "independent $\geq$ 0.6" and 0.19 in "independent $<$ 0.6". The CSC and RDBE kernels perform almost random on all three datasets.

We count for each kernel how often it has best F1 among all kernels for a single molecular property in cross-validation. When including the ALIGNF^1, 474 molecular properties are predicted with best F1 by ALIGNF^1. For 79 molecular properties, the NRBF kernel has better F1. When we exclude the linear kernel combinations, NRBF is best for 361 molecular properties, followed by the NSF with 45 properties and SLL with 33 properties. Six kernels, namely CSC,CPC, GJLSF, RDBE, RLPP, and LIPP are for none of the molecular properties the best predicting kernel.

We compare the weights of the kernels in the three MKL approaches: ALIGNF^1, ALIGNF^2 and ALIGNF+ (Fig.7.4). The three kernels with largest weights in all MKL methods are NLI, NSF, and SLL. In ALIGNF+, ten kernels get the maximum weight, nine kernels get zero weights, and only six kernels get a weight in between. In ALIGNF^1, 12 kernels get zero weight, including the two PPK kernels.

Discussion We demonstrated that the centred kernel alignment ALIGNF clearly outperforms all individual kernels. Only the NRBF kernel has similar prediction quality on cross-validation data. However, ALIGNF has also by far the best generalisation property; it shows good performance in cross-validation and on independent data. ALIGNF picks at least one kernel from every category with large weight. The two best individual kernels are NRBF and TREEALIGN. Unfortunately, these are also the computational most expensive kernels. ALIGNF^2, without both kernels, still shows excellent performance. This indicates

that a set of weaker kernels can compensate for more involved kernels. Furthermore, ALIGNF^2 has better performance on independent data. Note, that most spectra from the independent set were used to train hyperparameters of ALIGNF^2. Nevertheless, this indicates that the multiple kernel learning still benefit from more data. The difference between ALIGNF^1 and ALIGNF+ is almost negligible. Some kernels, like RDBE, have very low performance but are still picked with large weight in the multiple kernel learning. These kernels introduce features that cannot separate the data themselves but seem to improve separation in combination with other features. This would be not possible by ensemble learning strategies, where we train predictors for each kernel separately and average over their predictions.

Fragment based kernels perform better than other kernel categories; followed by path kernels. Interestingly, the PPK_{peak} kernel performs slightly better than the NB and NI kernels. This indicates that there are some peaks in the spectrum which are important for classification but are not annotated by SIRIUS. The uniform combination of PPK_{peak} and PPK_{diff} which is used in FingerID [79] has only slightly better identification rates in cross-validation but generalises much better on the test dataset. However, the difference in mean F1 between the FingerID kernel and the ALIGNF kernel of CSI:FingerID is around 10 percentage points for cross-validation, "independent ≥ 0.6", and "independent < 0.6".

ALIGNF seems to pick at least one kernel from every category. The three largest weighted kernels are the same for all three MKL approaches: NSF, NLI, SSL. All these kernels are using molecular formula subset relationships. Although the NRBF kernel has the best performance, it gets only moderate weight. The kernel weights in ALIGNF^2 are very sparse. In ALIGNF+ most kernels get either maximum weight or zero. It is therefore very similar to UNIMKL, but seems to result in some kind of feature selection by removing kernels that do not contribute to the alignment.

7.9 Evaluation of Structure Database Search

In this section we evaluate the capability of CSI:FingerID for searching in structure databases. We first compare CSI:FingerID against its predecessor FingerID [79]. We evaluate how each of the methodical enhancements in CSI:FingerID contributes to the identification rates when searching structure databases. In the next part we evaluate against several in silico and combinatorial fragmenters.

7.9.1 Training Data and Evaluation Set

We carry out most evaluations on the dataset from [58]. We use 4 138 small compounds from GNPS [227] and 2 120 compounds from the Agilent library for training. The corresponding model is called "CSI:FingerID (small)". For evaluation, we use 3 868 compounds from GNPS. Thus, for evaluation, we discarded some compounds from the GNPS dataset: namely, we discarded compounds containing metal complexes and other structures not connected by covalent bonds, non-protonated ions, and compounds that carry other charges.

The up-to-date version of CSI:FingerID 1.1 is trained on a larger dataset with 15 235 structures, with 19 118 independent MS/MS measurements in positive mode. For negative ion mode spectra, the model is trained on 5 321 structures with 10 823 independent measurements. See Chapter 5 for more details. We call both the positive and negative

Table 7.1: The kernel weights used in the small CSI:FingerID model trained on 6 258 compounds, the large CSI:FingerID model trained on 19 118 compounds measured in positive ion mode, and the large CSI:FingerID model trained on 12 548 compounds measured in negative ion mode. The first two models were trained using ALIGNF. For the negative ion mode model we used ALIGNF+. (*) The model for negative ion mode spectra is using the NRBF kernel. This kernel was omitted for the other two models, because it is computational very expensive.

Abbrev.	Kernel name	Kernel weights		
		CSI:FingerID (small)	CSI:FingerID (large) pos.	CSI:FingerID (large) neg.
NI	node intensity	0.194	0.021	0.000
NLI	node-loss interaction	0.191	0.215	0.032
NSF	node subformula	0.183	0.193	0.277
SLL	subformula in leafs and losses	0.114	0.119	0.053
JLSF	joined loss subformula	0.085	0.062	0.032
JLB	joined loss binary	0.079	0.058	0.032
LPC	loss pair counter	0.046	0.004	0.000
RLI	root loss intensity	0.022	0.023	0.032
LI	loss intensity	0.021	0.011	0.000
PPK	probability product	0.018	0.082	0.032
RDBE	ring double bond equivalent	0.017	0.074	0.061
MLIP	maximum loss in path	0.011	0.052	0.032
GJLSF	generalised joined loss subformula	0.009	0.028	0.000
NB	node binary	0.009	0.053	0.032
RLB	root loss binary	0.000	0.004	0.032
NRBF*	node radial basis function	–	–	0.351

model "CSI:FingerID (large)". The negative model is used solely for the negative ion mode spectra in the CASMI 2016 evaluation in section 7.9.4; all other evaluations are carried out on positive ion mode spectra.

We have a second evaluation set consisting of compounds from NIST, GNPS, Agilent 2.0, and MassBank. We restrict the evaluation on the 30 105 compounds in this large evaluation set for which we have a structure recorded in PubChem. Note, that evaluations on this large dataset are computational very expensive. Therefore, we use the small GNPS dataset for most evaluations.

We report the kernel weights used in all three models in Table 7.1. Why using different models of CSI:FingerID for evaluation? First, this allows us to evaluate how additional training data improves identification performance. Second, the larger models are computational more expensive and, thus, we have to perform evaluations of hyperparameters (e.g. different scorings) on the smaller model. The large models are used in the up-to-date version of CSI:FingerID, while the small model is trained on the same set of spectra as the initial version of CSI:FingerID 1.0 from [58].

For each of both training sets, we train 10 different CSI:FingerID models, each model leaves out one cross-validation fold. Kernel weights are optimised on the complete training set. We predict fingerprints for each spectrum in the evaluation set using the model that is not trained on a spectrum with same structure. Thus, we ensure that the predictor never predicts on a structure it has already seen in its training data.

We report identification rates for rank k as the frequency of observing the correct structure in the top k positions of the ranking list. Sometimes, several candidate structures receive the same score. In particular, this happens when two structures have an identical molecular fingerprint. We break ties by adding random noise onto the score and report the expected mean identification rate. This means that if the correct compound is tied with l other compounds on rank i to $i + l$, we will add $\frac{1}{l+1}$ to the frequency of observing the correct compound on rank i, $i + 1$,. . . ,$i + l$.

We search the predicted fingerprints in the PubChem database. We assume that the molecular formula was correctly identified beforehand. Thus, we only score candidate structures with the correct molecular formula. We also exclude structures which are unconnected or carry additional charges. Such structures cannot be measured in this

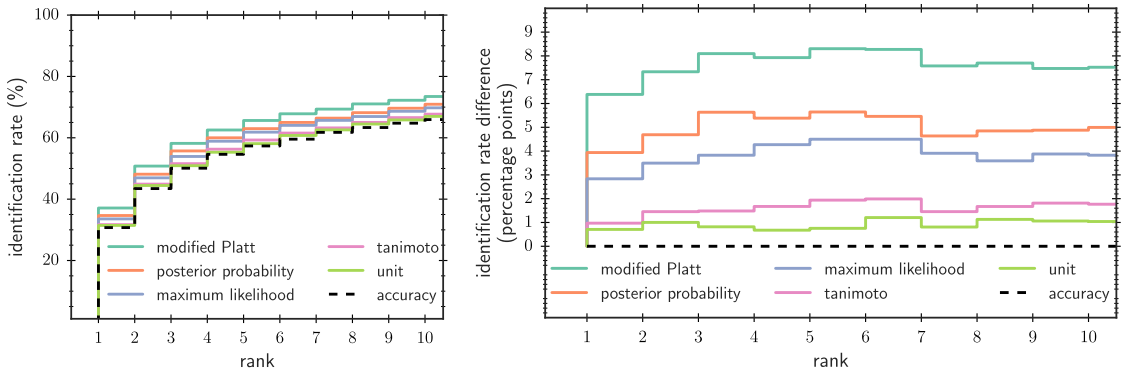


Figure 7.5: Left: Evaluation of different scoring functions for our method. Searching with $N = 3\,868$ compounds from GNPS dataset. See Section 7.6 for a description of the different scoring functions. Right: Difference in percentage points to the "accuracy" scoring function from [79, 196].

form in a mass spectrometry instrument. We will also evaluate search performance on the biological database, consisting of molecular structures from several biological databases (see Table 5.2 in Chapter 5).

7.9.2 Evaluation against FingerID

CSI:FingerID is the successor of FingerID from [79]. It adds fragmentation tree kernels, a new scoring and new molecular fingerprints. In this section we evaluate how each of these enhancements contributes to the performance of CSI:FingerID. Unless otherwise stated, all evaluations are done on the small GNPS evaluation set with the "CSI:FingerID (small)" model.

Scoring Methods In Fig. 7.5 we report identification rates for the different scoring functions described in Section 7.6. We find that all new scorings perform better than the function proposed in [79, 196]. Among these, "modified Platt" achieves the best identification rates and was therefore selected as the default scoring function for CSI:FingerID. Compared to the original scoring function based on predictor accuracy [79, 196], we reach 20.77 % (6.4 percentage points) more correct identifications.

Additional Fingerprints Do additional molecular properties improve the identification rates? We perform database search using only the FP2 fingerprints (55 bits). Then we successively add additional fingerprints and compare identification rates. We repeat the same evaluation, but randomly select a varying number of molecular properties from the set of all molecular properties. See Fig. 7.6. The first three fingerprints (FP2, FP3, and MACCS) with 528 bits in total are used by FingerID [79, 196]. PubChem and Klekota-Roth fingerprints were added in CSI:FingerID 1.0 [58]. ECFP fingerprints were added in CSI:FingerID 1.1. We see that from around 400 molecular properties, adding additional fingerprints results in a linear increase in identification rates. However, when randomly selecting molecular properties, it seems that this increase starts to saturate. This indicates that molecular properties are more similar to other molecular properties of the same fingerprint than to molecular properties of other fingerprint types. We assume that not the

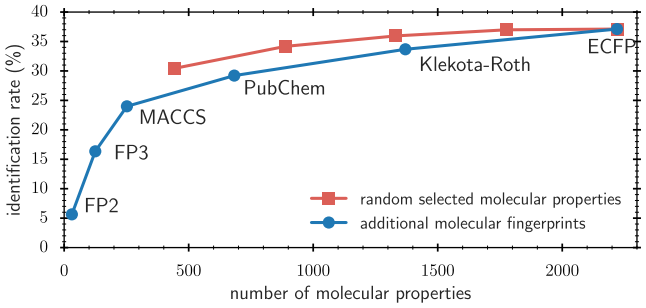


Figure 7.6: Identification rate of CSI:FingerID using modified Platt scoring with different sets of fingerprints. The blue line (circles) shows the identification rate when successively adding additional molecular fingerprints. The red line (squares) shows the identification rate when randomly selecting 444, 888, 1332, 1 776 and all 2 220 molecular properties. We only count molecular properties which are used in the scoring; these are molecular properties with more than 25 positive samples and an F1 score above 0.25.

number of molecular properties, but their diversity is important for molecular structure identification. Therefore, it is still worth to search for new kind of fingerprints.

New Kernels We have already shown how the new fragmentation tree kernels and the multiple kernel learning improves the prediction accuracy (Section 7.8). How does this contributes to the scoring? In Fig. 7.7 we compare the identification rates of CSI:FingerID when just using the PPK kernel against the identification rate when using multiple kernel learning on fragmentation tree kernels. We find that the difference is 4.2 percentage points on the small dataset.

So far, we trained on two quite homogeneous datasets: GNPS and Agilent, with most spectra in there are measured on Q-ToF instruments. We repeat this evaluation on the larger evaluation set (30 105 compounds) with the large CSI:FingerID model and find that fragmentation tree kernels identify 36.15 % of the compounds on top rank while the PPK kernel only reaches an identification rate of 28.14 %. Thus, on a less homogeneous dataset the difference between PPK and fragmentation tree kernels almost doubles to a difference of 8 percentage points.

Note that the identification rate of CSI:FingerID in [58] on the GNPS data is 31.8 %. The small model here, when trained on the same fingerprints and using the same scoring function, reaches an identification rate of 33.7 %. Thus, just by improving and adding new fragmentation kernels we could improve identification rates by 1.9 percentage points.

More Training Data As our method employs machine learning to predict molecular properties of the query compound, additional data can improve the performance of the predictors. FingerID was initially trained on 293 high resolution mass spectra from MassBank. The first version of CSI:FingerID used 6 258 training compounds. The up-to-date version of CSI:FingerID is using 19 118 training compounds. In [58] we trained several models of CSI:FingerID with different number of training compounds. For the resulting identification rates, we observed an almost linear increase when varying the relative training data size between 40 % and 90 %, whereby 400 additional compounds in the training data result in an increase of roughly one percentage point in the identification rate. We do not

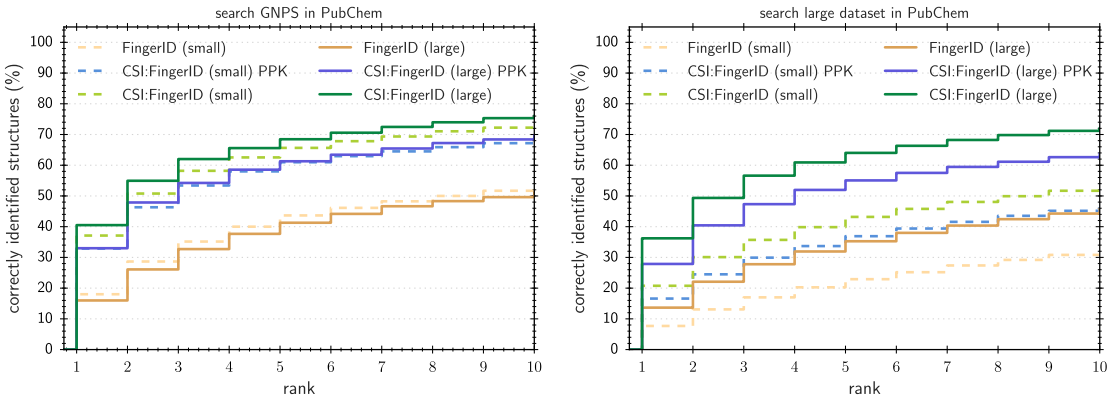


Figure 7.7: Identification rates of CSI:FingerID and FingerID on the small dataset with 3 868 compounds from GNPS (left) and on the large dataset with 30 105 compounds (right). FingerID is the method from [79]. CSI:FingerID PPK is using the PPK kernel like FingerID but the larger fingerprints and the modified Platt scoring function of CSI:FingerID. CSI:FingerID (small) PPK and CSI:FingerID (small) are trained on the small GNPS and Agilent data. CSI:FingerID (large) PPK and CSI:FingerID (large) are trained on the large dataset.

repeat this evaluation, but compare the identification rates between the small model and the large model, see Fig. 7.7. We find that the small model identifies 37.11 % of the GNPS compounds while the large model identifies 40.51 %. Thus, although the number of training data was increased by more than three times, identification rates have increased only by 3.4 percentage points. However, on the large evaluation set the difference between small and large model increases significantly: The small model identifies 21.9 % of the compounds while the large model identifies 36.15 %. The largest difference can be observed for NIST compounds, where the small model can only identify 17.7 % of the compounds and the large model identifies 35.5 % of the compounds. The reason is most likely that NIST spectra are measured on Orbitrap instruments, but most training spectra for the small model are measured on Q-ToF. Another reason might be that the distribution of compounds and compound classes in NIST differs a lot from the compounds in GNPS. However, these big differences in identification rates cannot be addressed solely on the heterogeneity of the larger evaluation set: Also the identification rates on GNPS compounds increase from 23.14 % to 34.16 % when using more training data.

Interestingly, when using only the PPK kernel, identification rates on GNPS do not significantly increase when using more training data. This indicates that the fragmentation tree kernels can extract more patterns from the training data than the spectral kernels.

We conclude that more training data make CSI:FingerID "more robust", in the sense that it will get to learn fragmentation patterns previously not observed in the training data and will be able to generalise across different instruments and experimental setups; but that further improvements in identification rate on a dataset which has already a good coverage in the training data will be mostly due to methodological advances.

7.9.3 GNPS Cross-validation

We have shown that CSI:FingerID significantly improves on the prediction of molecular fingerprints as well as on searching in structure databases. But we only evaluated against its predecessor FingerID. In this section we want to evaluate CSI:FingerID against a wide range

of methods for searching in structure databases. So far, this is the most comprehensive evaluation of different methods for searching in structure databases. We use the same data for training and evaluation as in the previous section. The methods we compare against are MetFrag [241], MIDAS [229], MAGMa [170], CFM-ID 1.5 [5], and FingerID [79]. The evaluation was initially done in [58]. Since then, CFM-ID 2.0 was released [6]. On the CASMI 2016 contest, CFM-ID 2.0 performed almost on par with its predecessor, with only 0.4 % more correct identifications. Simulating spectra with CFM-ID is computational very expensive. Simulating whole PubChem with CFM-ID would require several weeks on our compute cluster. Therefore, we do not repeat the evaluation for CFM-ID 2.0. However, we will evaluate against CFM-ID in Section 7.9.4. Two important new tools for structure database search were released since our evaluation in [58]: MS-FINDER [212] and ChemDistiller [114]. However, both tools are not trivial to evaluate. MS-FINDER runs only on Windows. We observed very low identification rates with MS-FINDER as well as different scores when using the commandline tool and the user interface. Because we cannot be absolutely sure that this is not due to a wrong usage of the software, we exclude it from analysis. ChemDistiller is a machine learning tool like FingerID and, thus, we would have to retrain it on our data. But the source code for training is not publicly available. In their own evaluation, ChemDistiller performed on par with MetFrag. Furthermore, ChemDistiller uses an integral mass kernel which was reported to perform significantly worse than the PPK kernel in FingerID [79].

Methods and Parameters FingerID was used as described in the publication [79], and predicts 528 molecular properties. Competitive Fragmentation Modelling ID (CFM-ID) [5] was downloaded from `http://sourceforge.net/projects/cfm-id/` (version 1.5). As suggested in [5] we used the Combined Energy model (CE-CFM). From the ten cross-validation models provided on the CFM website, we selected the first one. The GNPS dataset was processed using the "medium energy" model. We used the mass accuracy (maximum of 10 ppm and 0.01 Da) and probability threshold (0.001) suggested in [5]. MetFrag [241] was downloaded from `http://c-ruttkies.github.io/MetFrag/projects/commandline/` on June 18, 2014 (MetFrag does not provide version numbers). We used a 10 ppm relative mass accuracy, a 0.005 Da absolute mass accuracy, and the Java Virtual Machine option `-Djava.util.Arrays.useLegacyMergeSort=true`. In contrast to other approaches, MetFrag uses the sum (instead of the maximum) of relative and absolute mass accuracy. MIDAS [229] was downloaded from `http://midas.omicsbio.org` (version 1.1). MIDAS does not allow us to choose a relative mass accuracy; we chose an absolute mass accuracy of 0.01 Da. We modified MIDAS to output all candidates, instead of the top 5 only. In view of exceedingly long running times, we stopped MIDAS for those instances where computation was not completed after 24 h; the resulting instances were counted as random. MAGMa [170] (version 1.0) was provided as a standalone program by Lars Ridder (Wageningen University), and was run with options `-f -b 4 -c 0 -d 0` and mass accuracies of 0.001 Da and 10 ppm.

For CSI:FingerID we will restrict evaluation on the large model. Note, that identification rates between small and large model are quite similar, anyways (see Fig 7.7).

In our evaluation, we performed a certain amount of parameter optimisation for all methods, but found that the above choices resulted in maximum identification rates for the combined dataset. We retrained the original FingerID on the same training data with 19 118 compounds as the method proposed here, using cross validation; this is done to

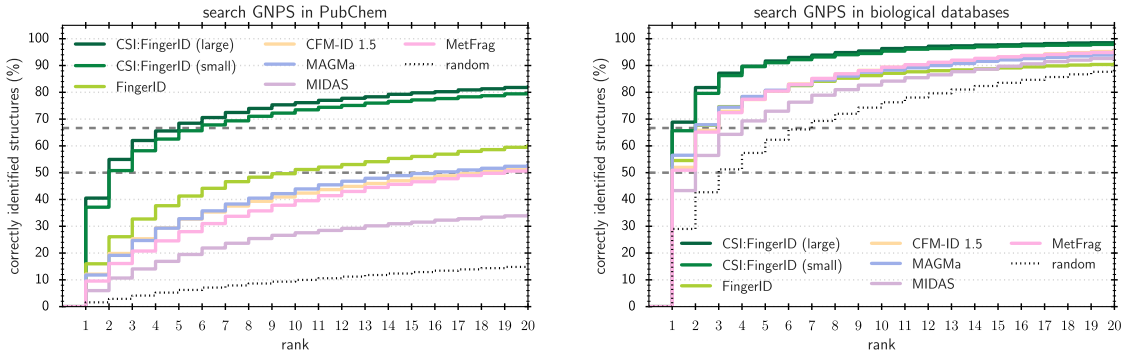


Figure 7.8: Methods evaluation: Percentage of correctly identified structures found in the top k output of the different methods, for maximum rank $k = 1, \ldots, 20$. Searching $N = 3\,868$ compounds from GNPS dataset in PubChem (left), and the $N = 2\,858$ bio-compounds from GNPS dataset in the biological database (right). Identification rates 50 % and 66.7 % marked by dashed lines.

allow a sensible evaluation of the impact of the methodological advances. For CFM-ID, we also evaluated the "low energy" and "high energy" model for the GNPS dataset as well as an absolute mass accuracy of 0.002 Da and an intensity baseline of 0.02 %. For MAGMa and MetFrag, we merge spectra at different fragmentation energies by choosing mass and intensity of the largest peak within the given mass accuracy. For MetFrag, this resulted in significantly better identification rates than the method proposed in the original MetFrag publication [241]. For MetFrag, we also tested absolute mass accuracies of 0.001 Da, 0.002 Da, and 0.01 Da. For MAGMa, we also tested absolute mass accuracies of 0.002 Da, 0.005 Da, and 0.01 Da, and option `-b 3`.

CFM-ID, MAGMa, MIDAS, and MetFrag cannot process all instances, sometimes crashing and sometimes omitting the correct answer from the output list. Some problems are seemingly caused by atoms that have uncommon valence states, such as chlorine with valence five in 5-chlorylsulfanyl-1-methylindazole (PubChem CID 57613613). Using SMILES from PubChem for CFM-ID, and SDF files from PubChem for MAGMa resulted in fewer crashes, and was used here. In case one tool cannot process an instance, we give all structure candidates of this instance the same score.

Results and Discussion We evaluate each method using compounds from the GNPS dataset with 3 868 compounds. CSI:FingerID strongly outperforms all other available tools for searching MS/MS data in a molecular structure database (Fig. 7.8). Compared to the runner-up, FingerID, the number of correct identifications is 2.5-fold higher (40.5 % vs. 15.99 %) when searching PubChem. CFM-ID reaches third place with 12.1 % identification. We achieve 67.8 % correct identifications in the top 5 output; next come CFM-ID and MAGMa both with 32.7 %.

Searching biocompounds in the biological database, we achieve 68.9 % correct identifications, compared to 56.5 % and 54.5 % for the two next best methods, MAGMa and FingerID, respectively. For 91.3 % of the query compounds, the correct answer is contained in the top 5 for our method, compared to runners-up FingerID (81.2 %) and MAGMa (80.8 %).

We also evaluate against the baseline method of randomly ordering candidates with the correct molecular formula. Random ordering performs well for searching bio-compounds in the bio-database, with 29.0 % correct identifications and 62.3 % in the top 5 for the combined dataset. This demonstrates the power of knowing the correct molecular formula for structure elucidation when searching in a restricted structure database. However, 1 016 compounds (26.3 % of the GNPS compounds) are not contained in the biological database and, thus, cannot be identified when restricting the search to known biomolecules.

7.9.4 Re-evaluating CASMI 2016

Using a preliminary version of SIRIUS and CSI:FingerID, we participated in the CASMI 2016 contest (category 2, automated methods) [191]. The contest provided MS/MS data (but no isotope patterns) for 208 compounds, with 127 and 81 in positive and negative ion mode, respectively. Candidate compound structures were provided as part of each challenge, and were retrieved from ChemSpider based on precursor mass. At that time, little reference data were publicly available for negative ion mode, and CSI:FingerID was trained exclusively on positive ion mode; therefore, we participated solely for the 127 challenges measured in positive ion mode. Missing isotope pattern data made it considerably harder to identify the correct molecular formula, so this search was restricted to molecular formulas in the candidate structure list. In the contest, we were able to unambiguously identify the correct structure for 70 of the 127 challenges. In addition, the correct structure tied for first place for 7 challenges; this is possible if the molecular fingerprints of two or more structure candidates are identical. In total, for 60.6 % of the challenges, the correct structure was on or tied for first place.

Runner-up with regards to correct identifications for positive ion mode data was the Input Output Kernel Regression (IOKR) version of CSI:FingerID [32], which correctly identified 53 structures (not counting ties). In this contest, there was significant overlap between *structures* in training and challenge data; in a subsequent evaluation, we found that CSI:FingerID was able to unambiguously identify 35 challenges (40 with ties) if we ensure structure-disjoint training data. See Schymanski *et al.* [191] for details.

Here, we reevaluate on the CASMI 2016 data using the up-to-date version of CSI:FingerID trained on the large dataset. As we now have sufficient negative ion mode training data available, we also evaluate CSI:FingerID on these challenges.

Methods and Parameters The CASMI contests do not provide a statistically sound evaluation where score ties are randomly broken and empty output lists (if a program crashes on an instance) are assessed as random ordered candidate list. This has weird effects: for example, CSI:FingerID would get a better CASMI score if we would have added small random noise on the score and randomly ordered negative challenges instead of omitting them. Here, to keep the evaluations consistent, we will use the same statistics for assessing the identification rates of the methods as we did in the previous section. Therefore, the results presented here will be slightly different than in [191]. However, this does not change the general ranking of the methods. We will call identifications without ties "unambiguous".

For this evaluation, we again predict fingerprints using a cross-validation fold that was not trained on the correct structure. Thus, CSI:FingerID was never trained on a compound it has to predict. Note the the original submission to the CASMI contest was trained on all training instances and, thus, will have better identification rates.

Table 7.2: Identification rates of various methods for the CASMI 2016 challenges. Identification rates are given as the ratio of challenges in which the correct answer is ranked within the top 1, 5, and 10 ranks. Ties are broken with random noise. (*) The submissions "CSI:FingerID (o.s.)[1]" and "IOKR[1]" are trained on compounds which also occur (as independent measurements) in the CASMI challenge data. Therefore, both methods have better identification rates than "CSI:FingerID (o.s.)" and "CSI:FingerID (large)", which ensure no overlap between training and evaluation data.

	Positive ion mode			Negative ion mode		
	Top 1	Top 5	Top 10	Top 1	Top 5	Top 10
CSI:FingerID (large)	39.0 %	67.8 %	74.1 %	29.9 %	55.6 %	61.7 %
CSI:FingerID (o.s.)	29.8 %	62.2 %	69.3 %	N/A	N/A	N/A
CSI:FingerID (o.s.)[1]	57.7 %	73.2 %	78.7 %	N/A	N/A	N/A
IOKR*	56.6 %	70.9 %	81.1 %	11.1 %	25.9 %	34.6 %
MetFrag+CFM-ID	17.5 %	45.4 %	57.9 %	27.5 %	54.4 %	67.5 %
MetFrag	14.6 %	37.1 %	48.3 %	17.8 %	48.2 %	58.1 %
MAGMa	13.2 %	35.2 %	45.0 %	16.6 %	44.6 %	60.1 %
MAGMa+	16.0 %	34.7 %	46.6 %	17.7 %	45.1 %	56.7 %
MS-FINDER	27.8 %	40.7 %	46.0 %	21.7 %	46.8 %	54.8 %
CFM-ID 1.5	21.9 %	48.5 %	61.7 %	16.7 %	44.1 %	56.9 %
CFM-ID 2.0	22.0 %	51.2 %	55.9 %	16.7 %	44.1 %	56.9 %

In contrast to the GNPS cross-validation, we will not assume that we know the correct molecular formula beforehand. Instead, we compute fragmentation trees with SIRIUS using the default settings from SIRIUS 4 with an allowed mass deviation of 10 ppm. We set the adduct type to "unknown" (protonation, sodium adduct, and potassium adduct for positive ion mode; deprotonation and chlorine adduct for negative ion mode). We *do not* restrict SIRIUS to use molecular formulas from some of the provided structure list; instead, we allow all possible molecular formulas that can be composed from elements CHNOPSFIBrCl and enable automated element detection. We use a soft threshold for retrieving the structure candidates: We select the best fragmentation tree as well as all trees with a score larger than 75 % of the optimal score. For each of these trees we predict a molecular fingerprint and search in the provided structure candidate list. We then merge the scored candidate lists for each tree.

We call the original submission of CSI:FingerID "CSI:FingerID (o.s.)".

Results and Discussion See Fig. 7.9 and Table 7.2 for the identification rates of all participating methods in CASMI 2016 as well as for "CSI:FingerID (large)" as presented here. In six cases, the correct molecular formula (and, hence, the correct candidate structure) was discarded by soft thresholding and, thus, CSI:FingerID could not find the correct structure. In two additional cases, the automatic element detection failed. In positive ion mode, "CSI:FingerID (large)" unambiguously identifies the correct structure for 48 of 127 challenges. In three cases, the correct structure is tied for first place with another structure. Thus, the identification rate of "CSI:FingerID (large)" is 39.0 % for top rank and 67.8 % for top five. IOKR and the original CSI:FingerID submission which are not structure disjoint have for this reason a better identification rate with 56.6 % for IOKR and 57.7 % for CSI:FingerID (o.s.)*.

For negative ion mode challenges, "CSI:FingerID (large)" unambiguously identifies the correct structure for 23 of 81 challenges; in addition, for two challenges the correct

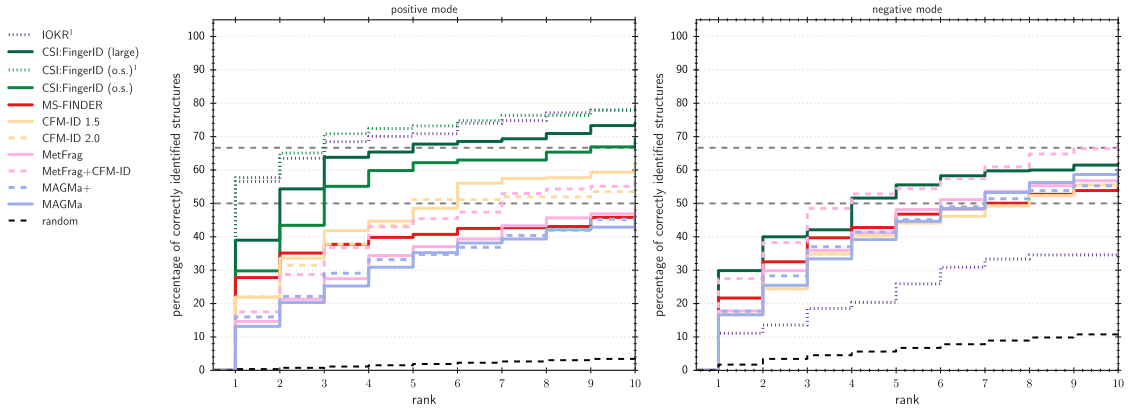


Figure 7.9: Identification rates of several methods on the CASMI 2016 challenges. We separately evaluate positive and negative mode challenges. "CSI:FingerID (o.s.)" is the original submission of CSI:FingerID to the CASMI 2016 contest. "CSI:FingerID (large)" denotes the method presented here. The methods IOKR* and CSI:FingerID (o.s.)* are evaluated without cross-validation. Therefore, they sometimes have to rank a structure which was already part of the training data as independent measurement.

structure is tied for first place with another structure with identical fingerprint. The identification rate is 29.9 % for top rank and 55.6 % for top five. In comparison, MS-FINDER [212], CFM-ID [5, 6] and MAGMa+ [222] correctly and unambiguously identified 14, 12 and 8 structures in negative ion mode, respectively, as part of the CASMI 2016 contest [191]. IOKR [32] which was trained on a rather small set of negative ion mode data, unambiguously identified 9 structures for the negative ion mode challenges [191].

In practice, it is often reasonable to search in a more restricted database focusing on biomolecule structures. When we restrict the search to the biomolecule structure database (Table 5.2 in Chapter. 5) which contains about half a million structure candidates, identification rates improve significantly: For positive ion mode we identify the correct structure for 91 of 127 challenges. Identification rate is 72.1 for top rank and 88.2 % for top five. For negative ion mode we correctly identify 46 of 81 challenges. Identification rate is 59.1 % (85.2 % for top five).

We note that some of the CASMI 2016 challenges do not contain enough information to be structurally elucidated by any computational method or even manual interpretation: For 19 challenges, the fragmentation trees of the correct molecular formula explain less than three peaks. For nine challenges, even the MS/MS spectrum contains only two peaks. We argue that structural elucidation using a spectrum with two peaks or less is rather arbitrary.

7.10 Evaluation of Fingerprint Prediction

In this section we want to take a closer look at the molecular properties which can be and cannot be predicted with CSI:FingerID. We use the "CSI:FingerID (large)" model for this evaluation and the statistics obtained via cross-validation.

First, we want to compare the different fingerprints we predict: FP2, FP3, MACCS, PubChem, Klekota-Roth, and ECFP. See Fig. 7.10 (left) for the F1 scores of molecular properties for each of these fingerprints. The Klekota-Roth and ECFP fingerprints contain

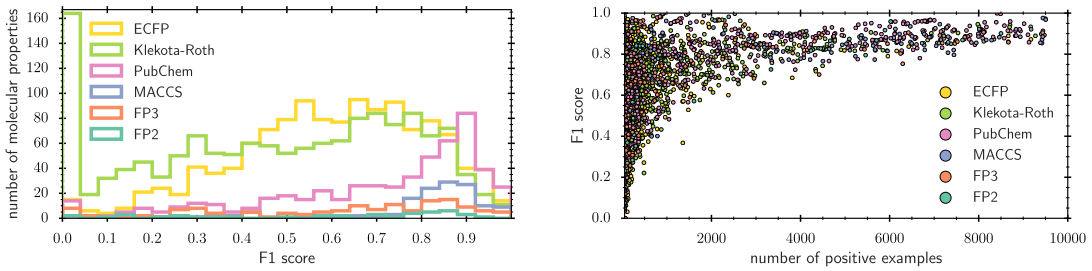


Figure 7.10: Left: Histogram of F1 scores of molecular properties of different fingerprint types. The F1 scores are obtained via cross-validation. Right: A scatter plot of F1 score and the frequency of a molecular property in the training data.

the most molecular properties. However, most molecular properties with high F1 stem from PubChem fingerprints: There are 863 molecular properties with F1 above 0.8. 259 belong to PubChem fingerprint, 238 to ECFP, and 203 to Klekota-Roth. The 139 molecular properties of the MACCS fingerprint have in average the highest F1 over all fingerprint types with 99 molecular properties having F1 above 0.8 and only 14 molecular properties having an F1 below 0.5. The average F1 is 0.79. Klekota-Roth has the lowest average F1, but this is due to the fact that the Klekota-Roth fingerprint is very sparse, with 592 molecular properties occuring less than 50 times in the dataset.

Is there a relation between the frequency of a molecular property and its F1 score? See Fig. 7.10 (right) for a plot of molecular property frequency against F1 score. Although, there is a relation between both properties, this relation is weaker than we would have expected. The Pearson correlation between frequency and F1 is 0.48. However, this might be an artefact of the statistics: For frequent molecular properties even the random predictor has a larger F1. If we exclude molecular properties with a frequency below 25 and above 1 000, the correlation drops to 0.36. Still, there are a lot of molecular properties that can be predicted with very high F1 but occur very infrequently in the data.

Similarly, we can ask if there is a relation between the size of the molecular property and its F1. We measure the size of a property in terms of number of atoms. Naturally, this is only possible for molecular properties that match to a specific molecular substructure. Furthermore, this measure is sometimes not clearly defined. For example, the SMARTS "[cX3]" matches for an aromatic carbon atom with three neighbours. We decided that the number of atoms for this property is 1. See Fig. 7.11 for a plot of molecular property size against F1. There is no clear pattern. The Pearson correlation is -0.28, but this is most likely an artefact of the correlation between molecular property size and frequency with Pearson correlation of -0.25.

In general, we do not see a common pattern among molecular properties with high F1 and molecular properties with low F1. Maybe it is the distribution of the training data that is important for the prediction quality? We use the "CSI:FingerID (large)" model to predict fingerprints for all instances in the large dataset. We also use these fingerprints to search in PubChem and report the worst-case rank using the modified Platt scoring. We now search for each predicted fingerprint the five most similar compounds in the training data to this test compound. For similarity, we use the Tanimoto similarity on all molecular properties we can predict. We report the median of these Tanimoto values (that is: the Tanimoto of the third closest training point). For prediction accuracy, we have used the

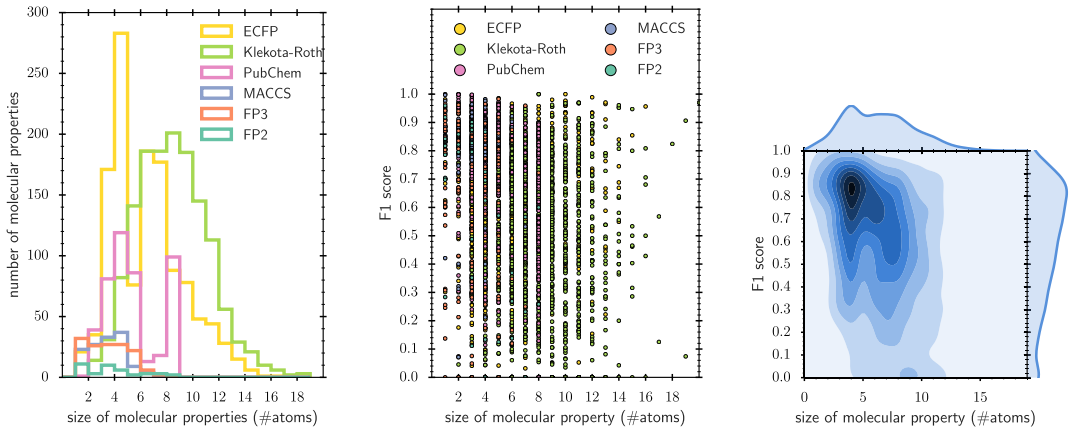


Figure 7.11: Left: Histogram of molecular properties sizes. The size of a molecular property is measured as the number of atoms of the corresponding molecular substructure. Middle: A scatter plot of size and F1 score of molecular property. F1 scores are obtained via cross-validation. Right: The same data visualised in a kernel density plot.

Tanimoto of the predicted fingerprint against the real fingerprint. See Fig. 7.12 for density plots of Tanimoto similarity to training data against prediction accuracy and ranking. We find that the quality of the predicted fingerprint depends strongly on the similarity of compounds in the training data. Pearson correlation between both variables, Tanimoto of third closest training compound and Tanimoto between predicted and true fingerprint, is 0.69.

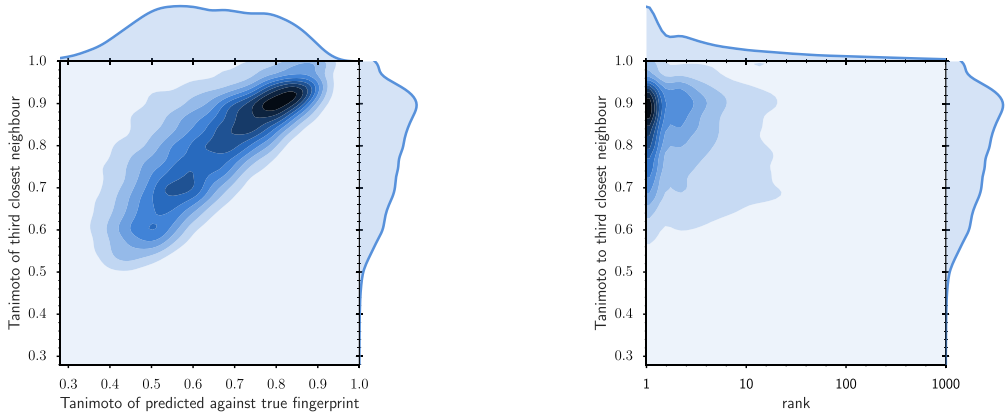


Figure 7.12: Kernel density plots of prediction results on the large dataset with the "CSI:FingerID (large)" model. As usual, we ensure that the model is not trained on the same structure it has to predict. Left: Kernel density of Tanimoto of the third most similar training compound to the test compound and the Tanimoto of the predicted fingerprint against the true fingerprint. Right: Kernel density of Tanimoto of the third most similar training compound to the test compound and the rank of the compound when searching the predicted fingerprint in PubChem. In case of ties, we arbitrarily decided for the worst case rank. Note that the ranks are logarithmised.

8 Conclusion

We presented two methods for the analysis of mass spectrometry data: SIRIUS computes fragmentation trees for identifying the molecular formula of the ion and annotating the fragment peaks and fragmentation reactions in the mass spectrum. In Chapter 6 we developed a maximum a posteriori probability estimation to improve the fragmentation tree computation. In Section 6.4 we introduced a maximum likelihood scoring for isotope pattern and described how to integrate isotope peak detection and scoring into the fragmentation tree optimisation. In Chapter 7 we presented CSI:FingerID, a tool that predicts a molecular fingerprint from the annotated mass spectrum. With this fingerprint, it searches in structure databases to identify the molecular structure.

Both methods are state-of-the-art: We attended several times to the "Critical Assessment of Small Molecule Identification" (CASMI) contest, a blind contest for molecular formula and structure identification from MS^2 data. In CASMI 2013, SIRIUS was selected as the best automated tool for molecular formula annotation. In CASMI 2017, CSI:FingerID won the contest, with almost twice as many correct identifications as IOKR (another method based on CSI:FingerID) and 6.5 times more correct identification than the next competitor.

We have demonstrated the performance of both methods on several datasets. SIRIUS reaches identification rates between 73.8 % on the Agilent and GNPS data and 86.4 % on the independent NIST dataset; see Section 6.7.2. When isotope pattern data are available, identification rates are even higher with 88.8 % on the Agilent 2.0 data and 93.3 % on the pesticide dataset from [206].

CSI:FingerID identifies 40.5 % of the GNPS spectra and 39.0 % of the CASMI 2016 spectra correctly; see Section 7.9. On a large dataset with 30 105 compounds from NIST, GNPS, Agilent, and MassBank, CSI:FingerID is able to identify 36.5 % of the spectra correctly. We compared CSI:FingerID against other methods for searching in structure databases and found that CSI:FingerID outperforms its predecessor FingerID with 2.5 fold more correct identifications, and the runner-up CFM-ID with 3.4 times more correct identifications.

Note, that the identification rates of SIRIUS refer to de novo molecular formula identification. Thus, we considered all possible molecular formulas and not only those contained in molecular databases. Similarly, we searched with CSI:FingerID in PubChem, a database which is several magnitudes larger than any biological molecule database. For most applications, people would rather search in specialised databases, for example plant libraries or microbial databases. When searching with CSI:FingerID in our already very large biological database, identification rates on GNPS increase to 68.9 %.

We have shown in Section 6.7 that the statistical model of SIRIUS generalises well across different datasets and even across different instruments. For example, we analysed the NIST dataset with 20 462 compounds measured on Orbitrap instruments, using the statistical model trained on 4 051 Q-ToF spectra from GNPS and Agilent. We find that the fitted parameters and hyperparameters work well and the list of common losses is comprehensive.

In contrast, for CSI:FingerID we found in Section 7.10 that the methods prediction performance for an unknown compound depends on the availability of structural similar compounds in the training data. However, the fragmentation tree kernels developed in this thesis are much more robust and generalise better on compounds which are structurally very dissimilar to training compounds; see Section 7.8.

All methods presented here are released under the name "SIRIUS" and written in the Java programming language. We offer a commandline tool as well as an interactive user interface that allows for batch processing large datasets and visualisation of the results. Structure elucidation with CSI:FingerID is implemented via a representational state transfer (REST) webservice: user can upload computed fragmentation trees and download the predicted molecular fingerprint directly from the SIRIUS user interface or commandline tool. This enables high performance, reliability, and scalability, and the integration of methodical upgrades without having the user to install new releases. No spectral libraries or compound structure databases have to be installed or updated on the user side. Thus, we can train the model on commercial libraries like NIST. Since its release, the CSI:FingerID web service was requested more than 4 million times. Currently, we count one million requests per month.

Ongoing Research and Future Work

In Section 6.7.2 we found that the molecular formula identification rates of SIRIUS drop for metabolites with large mass. Currently, Markus Ludwig is developing a network approach that uses Gibbs sampling to boost molecular formula annotations with SIRIUS. Preliminary results show that this results in identification rates above 90 % for large mass molecules on several biological datasets.

Such a network approach is also possible for CSI:FingerID. Da Silva et al. [45] used an annotation propagation method to improve the search in structure databases. However, such a method is highly biased, because it favours annotations with many biotransformations. To avoid this problem, we could operate on the fingerprint level and, thus, avoid database search at all. However, it is an ongoing research how to propagate fingerprint predictions in a network and if we should do this in an one-step approach (integrating the network into a machine learning model) or in a two-step approach (correcting the already predicted fingerprints).

Although we implemented several heuristics and reduction techniques to speed up SIRIUS, computing fragmentation trees might take hours for very large datasets. Currently, Oliver Alka et al. are working on a cloud server application that can compute fragmentation trees on a large compute cluster. We will integrate this cloud solution in the SIRIUS commandline tool and user interface, such that the users can smoothly switch between local and cloud computation.

We have shown in Section 7.9 that more molecular properties, better kernels, better scoring functions, and more training data are crucial for improving the search performance of CSI:FingerID. A promising candidate for more molecular properties are shortest path fingerprints. First tests reveal that shortest path fingerprints are quite different from the other fingerprints and can be predicted with high F1. We already demonstrated that tree alignment kernel shows good performance. But tree alignments are very computational expensive. Currently, my co-worker Thomas Köhler is researching in multiple tree alignments and profiled alignments which can then be used for developing novel tree

alignment kernels. For the scoring we found that a maximum likelihood scoring which take into account dependencies between molecular properties can improve upon the modified Platt scoring [123]. Finally, we can retrain CSI:FingerID on new training data whenever a new versions of GNPS (or other mass spectral library) is released. However, another way to increase the amount of available training data is reinforcement learning: We can use CSI:FingerID to identify molecular structures in biological samples, searching in highly limited databases (e.g. biological database). We can then use these identified compounds for retraining the method. However, to make this work we need a method that estimates a confidence or significance for the results of structure database search. Currently, my colleague Martin Hoffmann is working on a method for estimating a confidence score. Preliminary results on real biological samples are very promising.

Instead of aiming at structure identification, we can use predicted molecular fingerprints to search for structural similar molecules. This offers applications in natural product research, to distinguish potential novel molecules from analogues of already known metabolites. Preliminary research indicates that similarity between predicted fingerprints correlates better with structural similarity than the commonly used cosine similarity between mass spectra. Thus, predicted molecular fingerprints can be used for building molecular networks. In contrast to spectral similarity, such a fingerprint similarity measure can also be applied between a measured compound and a database structure.

So far, we used CSI:FingerID for the prediction of molecular substructures. But the kernel framework defined in Section 7.2 can be used for other tasks and methods, too. For example, we could use kernel regression methods for predicting numerical properties (e.g. polarity or hydrophobicity). It is also conceivable to replace kernel support vector machines by another machine learning method. Brouard *et al.* [32] uses input output kernel regression (IOKR) to improve searching in structure databases. We are also experimenting with logistic regression as an alternative to support vector machines. Beside supervised learning, we can also use unsupervised learning methods in combination with our kernel framework to cluster spectra or build molecular networks.

Tox21 is a reference dataset of hazardous compounds [211]. Mayr *et al.* [125] have shown that they could predict toxicity from molecular fingerprints using deep neural networks. So far, there is no such reference dataset for tandem mass spectrometry data. But we can retrain the deep neural network model in [125] on simulated probabilistic fingerprints. We can then feed the predicted fingerprints from CSI:FingerID directly into a deep neural network trained on *molecular structures* to predict toxicity. This is a general approach that can also be applied on other chemical properties for which we lack reference spectra. For example, we developed *CANOPUS*, a deep neural network for predicting compound categories as they are defined by Classyfire [51] directly from tandem mass spectra. This tool is able to predict compound categories, even if no reference spectra for a category are available for training.

SIRIUS and CSI:FingerID will be integrated into other software frameworks like OpenMS, MZmine, and GNPS. SIRIUS, CSI:FingerID, and CANOPUS are already intensively used in combination with molecular networking to identify families of metabolites in biological samples. So far, we run our methods mostly on reference spectra or the CASMI challenges. We expect exciting new findings when analysing whole biological datasets in large scale. There are already interesting preliminary results, on a large variety of data from the microbiome in human gut, fecal samples in immunosuppressed patients, microbiomes on human skin, and from marine microalgae.

Bibliography

[1] B. L. Ackermann, J. E. Hale and K. L. Duffin. The role of mass spectrometry in biomarker discovery and measurement. *Curr Drug Metab*, 7:525–539, 2006.

[2] A. A. Aksenov, R. da Silva, R. Knight, N. P. Lopes and P. C. Dorrestein. Global chemical analysis of biology by mass spectrometry. *Nature Reviews Chemistry*, 1(7): 0054, 2017.

[3] T. Alexandrov, S. Böcker, P. Dorrestein and E. Schymanski. Computational metabolomics: Identification, interpretation, imaging (Dagstuhl Seminar 17491). *Dagstuhl Reports*, 7(12):1–17, 2018.

[4] F. Allen, M. Wilson, A. Pon, R. Greiner and D. Wishart. CFM-ID: a web server for annotation, spectrum prediction and metabolite identification from tandem mass spectra. *Nucleic Acids Res*, 42(W1):W94–W99, 2014.

[5] F. Allen, R. Greiner and D. Wishart. Competitive fragmentation modeling of ESI-MS/MS spectra for putative metabolite identification. *Metabolomics*, 11(1):98–110, 2015.

[6] F. Allen, A. Pon, R. Greiner and D. Wishart. Computational prediction of electron ionization mass spectra to assist in GC/MS compound identification. *Anal Chem*, 88(15):7689–7697, 2016.

[7] T. Alon and A. Amirav. Isotope abundance analysis methods and software for improved sample identification with supersonic gas chromatography/mass spectrometry. *Rapid Commun Mass Spectrom*, 20(17):2579–2588, 2006.

[8] R. Aminov. A brief history of the antibiotic era: lessons learned and challenges for the future. *Front Microbiol*, 1:134, 2010.

[9] E. Bach, S. Szedmak, C. Brouard, S. Böcker and J. Rousu. Liquid-chromatography retention order prediction for metabolite identification. *Bioinformatics*, 34(17):i875–i883, 2018. Proc. of *European Conference on Computational Biology* (ECCB 2018).

[10] R. Bade, L. Bijlsma, T. H. Miller, L. P. Barron, J. V. Sancho and F. Hernández. Suspect screening of large numbers of emerging contaminants in environmental waters using artificial neural networks for chromatographic retention time prediction and high resolution mass spectrometry data analysis. *Sci Total Environ*, 538:934–941, 2015.

[11] D. Bajusz, A. Rácz and K. Héberger. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *J Cheminf*, 7(1):20, 2015.

[12] A. Barla, F. Odone and A. Verri. Histogram intersection kernel for image classification. In *Proc. of International Conference on Image Processing (ICIP 2003)*, volume 3, pages III–513. IEEE, 2003.

[13] L. P. Barron and G. L. McEneff. Gradient liquid chromatographic retention time prediction for suspect screening applications: a critical assessment of a generalised artificial neural network-based approach across 10 multi-residue reversed-phase analytical methods. *Talanta*, 147:261–270, 2016.

[14] C. A. Bauer and S. Grimme. How to compute electron ionization mass spectra from first principles. *J Phys Chem A*, 120(21):3755–3766, 2016.

[15] A. Bender, J. L. Jenkins, J. Scheiber, S. C. K. Sukuru, M. Glick and J. W. Davies. How similar are similarity searching methods? a principal component analysis of molecular descriptor space. *J Chem Inf Model*, 49(1):108–119, 2009. PMID: 19123924.

[16] Y. Bengio, N. L. Roux, P. Vincent, O. Delalleau and P. Marcotte. Convex neural networks. In *Proc. of Advances in neural information processing systems (NIPS 2006)*, pages 123–130, 2006.

[17] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle. Greedy layer-wise training of deep networks. In *Proc. of Advances in neural information processing systems (NIPS 2007)*, pages 153–160, 2007.

[18] H. P. Benton, D. M. Wong, S. A. Trauger and G. Siuzdak. XCMS2: Processing tandem mass spectrometry data for metabolite identification and structural characterization. *Anal Chem*, 80(16):6382–6389, 2008.

[19] I. Blaženović, T. Kind, J. Ji and O. Fiehn. Software tools and approaches for compound identification of lc-ms/ms data in metabolomics. *Metabolites*, 8(2), 2018.

[20] I. Blaženović, T. Kind, H. Torbašinović, S. Obrenović, S. S. Mehta, H. Tsugawa, T. Wermuth, N. Schauer, M. Jahn, R. Biedendieck, *et al.* Comprehensive comparison of in silico MS/MS fragmentation tools of the CASMI contest: database boosting is needed to achieve 93% accuracy. *J Cheminf*, 9(1):32, 2017.

[21] S. Böcker. Searching molecular structure databases using tandem MS data: are we there yet? *Curr Opin Chem Biol*, 36:1–6, 2017.

[22] S. Böcker and K. Dührkop. Fragmentation trees reloaded. *J Cheminform*, 8:5, 2016.

[23] S. Böcker and Zs. Lipták. A fast and simple algorithm for the Money Changing Problem. *Algorithmica*, 48(4):413–432, 2007.

[24] S. Böcker and V. Mäkinen. Combinatorial approaches for mass spectra recalibration. *IEEE/ACM Trans Comput Biology Bioinform*, 5(1):91–100, 2008.

[25] S. Böcker and F. Rasche. Towards de novo identification of metabolites by analyzing tandem mass spectra. *Bioinformatics*, 24:I49–I55, 2008. Proc. of *European Conference on Computational Biology* (ECCB 2008).

[26] S. Böcker, M. Letzel, Zs. Lipták and A. Pervukhin. Decomposing metabolomic isotope patterns. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2006)*, volume 4175 of *Lect Notes Comput Sci*, pages 12–23. Springer, Berlin, 2006.

[27] S. Böcker, Zs. Lipták, M. Martin, A. Pervukhin and H. Sudek. DECOMP–from interpreting mass spectrometry peaks to solving the Money Changing Problem. *Bioinformatics*, 24(4):591–593, 2008.

[28] S. Böcker, M. Letzel, Zs. Lipták and A. Pervukhin. SIRIUS: Decomposing isotope patterns for metabolite identification. *Bioinformatics*, 25(2):218–224, 2009.

[29] S. Böcker, J. Rousu and E. Schymanski. Computational Metabolomics (Dagstuhl Seminar 15492). *Dagstuhl Reports*, 5(11):180–192, 2016.

[30] L. Bottou and C.-J. Lin. Support vector machine solvers. In *Large scale kernel machines*, volume 3, pages 301–320. MIT Press Cambridge, MA, 2007.

[31] C. D. Broeckling, F. A. Afsar, S. Neumann, A. Ben-Hur and J. E. Prenni. RAMClust: a novel feature clustering method enables spectral-matching-based annotation for metabolomics data. *Anal Chem*, 86(14):6812–6817, 2014.

[32] C. Brouard, H. Shen, K. Dührkop, F. d'Alché-Buc, S. Böcker and J. Rousu. Fast metabolite identification with input output kernel regression. *Bioinformatics*, 32(12): i28–i36, 2016. Proc. of *Intelligent Systems for Molecular Biology* (ISMB 2016).

[33] M. L. Brownawell and J. S. Filippo Jr. A program for the synthesis of mass spectral isotopic abundances. *Journal of Chemical Education*, 59(8):663–65, 1982.

[34] B. Buchanan, G. Sutherland and E. A. Feigenbaum. *Heuristic DENDRAL: a program for generating explanatory hypotheses in organic chemistry.* Stanford University, 1968.

[35] R. Caspi, T. Altman, R. Billington, K. Dreher, H. Foerster, C. A. Fulcher, T. A. Holland, I. M. Keseler, A. Kothari, A. Kubo, M. Krummenacker, M. Latendresse, L. A. Mueller, Q. Ong, S. Paley, P. Subhraveti, D. S. Weaver, D. Weerasinghe, P. Zhang, and P. D. Karp. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Research*, 42(D1):D459–D471, 2014.

[36] J. Cautereels, M. Claeys, D. Geldof and F. Blockhuys. Quantum chemical mass spectrometry: ab initio prediction of electron ionization mass spectra and identification of new fragmentation pathways. *J Mass Spectrom*, 51(8):602–614, 2016.

[37] A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé and G. Pujadas. Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58 – 63, 2015. Virtual Screening.

[38] C.-C. Chang and C.-J. Lin. LIBSVM: A library for Support Vector Machines. *ACM Trans Intell Syst Technol*, 2(3):27:1–27:27, 2011.

[39] J. M. Chick, D. Kolippakkam, D. P. Nusinow, B. Zhai, R. Rad, E. L. Huttlin and S. P. Gygi. A mass-tolerant database search identifies a large proportion of unassigned spectra in shotgun proteomics as modified peptides. *Nat Biotechnol*, 33(7):743, 2015.

[40] M. Collins and N. Duffy. Convolution kernels for natural language. In *Proc. of Advances in neural information processing systems (NIPS 2002)*, pages 625–632, 2002.

[41] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[42] C. Cortes, M. Mohri and A. Rostamizadeh. Algorithms for learning kernels based on centered alignment. *J Mach Learn Res*, 13(1):795–828, 2012.

[43] N. Cristianini, J. Shawe-Taylor, A. Elisseeff and J. S. Kandola. On kernel-target alignment. In *Proc. of Advances in neural information processing systems (NIPS 2002)*, pages 367–373, 2002.

[44] R. R. da Silva, P. C. Dorrestein and R. A. Quinn. Illuminating the dark matter in metabolomics. *Proc Natl Acad Sci U S A*, 112(41):12549–12550, 2015.

[45] R. R. da Silva, M. Wang, L.-F. Nothias, J. J. J. van der Hooft, A. M. Caraballo-Rodríguez, E. Fox, M. J. Balunas, J. L. Klassen, N. P. Lopes, and P. C. Dorrestein. Propagating annotations of molecular networks using in silico fragmentation. *PLoS Comput Biol*, 14(4):e1006089, 2018.

[46] D. C. Dallas, W. F. Martin, S. Hua and J. B. German. Automated glycopeptide analysis – review of current state and future directions. *Briefings Bioinf*, 14(3): 361–374, 2012.

[47] W. Demuth, M. Karlovits and K. Varmuza. Spectral similarity versus structural similarity: Mass spectrometry. *Anal Chim Acta*, 516(1-2):75–85, 2004.

[48] T. Depke, R. Franke and M. Brönstrup. Clustering of MS2 spectra using unsupervised methods to aid the identification of secondary metabolites from pseudomonas aeruginosa. *J Chromatogr B*, 1071:19–28, 2017.

[49] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, third edition, 2005.

[50] S. L. Dixon and R. T. Koehler. The hidden component of size in two-dimensional fragment descriptors: side effects on sampling in bioactive libraries. *J Med Chem*, 42 (15):2887–2900, 1999. PMID: 10425098.

[51] Y. Djoumbou Feunang, R. Eisner, C. Knox, L. Chepelev, J. Hastings, G. Owen, E. Fahy, C. Steinbeck, S. Subramanian, E. Bolton, R. Greiner, and D. S. Wishart. Classyfire: automated chemical classification with a comprehensive, computable taxonomy. *J Cheminf*, 8(1):61, 2016.

[52] J. Duan, S. L. Dixon, J. F. Lowrie and W. Sherman. Analysis and comparison of 2d fingerprints: Insights into database screening performance using eight fingerprint methods. *J Mol Graphics Modell*, 29(2):157 – 170, 2010.

[53] A. Z. Dudek, T. Arodz and J. Gálvez. Computational methods in developing quantitative structure-activity relationships (QSAR): a review. *Comb Chem High T Scr*, 9(3):213–228, 2006.

[54] K. Dührkop and S. Böcker. Fragmentation trees reloaded. In *Proc. of Research in Computational Molecular Biology (RECOMB 2015)*, volume 9029 of *Lect Notes Comput Sci*, pages 65–79. Springer, Berlin, 2015.

[55] K. Dührkop, M. Ludwig, M. Meusel and S. Böcker. Faster mass decomposition. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2013)*, volume 8126 of *Lect Notes Comput Sci*, pages 45–58. Springer, Berlin, 2013.

[56] K. Dührkop, K. Scheubert and S. Böcker. Molecular formula identification with SIRIUS. *Metabolites*, 3:506–516, 2013.

[57] K. Dührkop, F. Hufsky and S. Böcker. Molecular formula identification using isotope pattern analysis and calculation of fragmentation trees. *Mass Spectrom*, 3(special issue 2):S0037, 2014.

[58] K. Dührkop, H. Shen, M. Meusel, J. Rousu and S. Böcker. Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proc Natl Acad Sci U S A*, 112(41):12580–12585, 2015.

[59] K. Dührkop, M. A. Lataretu, W. T. J. White and S. Böcker. Heuristic algorithms for the maximum colorful subtree problem. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2018)*, volume 113 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[60] B. Efron, R. Tibshirani, J. D. Storey and V. Tusher. Empirical Bayes analysis of a microarray experiment. *J Am Stat Assoc*, 96(456):1151–1160, 2001.

[61] J. E. Elias and S. P. Gygi. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nat Methods*, 4(3):207–214, 2007.

[62] E. Fahy, S. Subramaniam, H. A. Brown, C. K. Glass, A. H. Merrill, R. C. Murphy, C. R. H. Raetz, D. W. Russell, Y. Seyama, W. Shaw, *et al.* A comprehensive classification system for lipids. *J Lipid Res*, 46(5):839–862, 2005.

[63] E. A. Feighenbaum, B. G. Buchanan and J. Lederberg. On generality and problem solving: a case study using the DENDRAL program. *Machine Intelligence*, 6:165–190, 1971.

[64] A. R. Fernie, R. N. Trethewey, A. J. Krotzky and L. Willmitzer. Metabolite profiling: From diagnostics to systems biology. *Nat Rev Mol Cell Biol*, 5(9):763–769, 2004.

[65] G. C. Fonger, P. Hakkinen, S. Jordan and S. Publicker. The National Library of Medicine's (NLM) Hazardous Substances Data Bank (HSDB): background, recent enhancements and future plans. *Toxicology*, 325:209–216, 2014.

[66] J. Friedman, T. Hastie and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[67] Y. Fu, C. Zhao, X. Lu and G. Xu. Nontargeted screening of chemical contaminants and illegal additives in food based on liquid chromatography–high resolution mass spectrometry. *Trends Anal Chem*, 2017.

[68] H. Geppert, M. Vogt and J. Bajorath. Current trends in ligand-based virtual screening: molecular representations, data mining methods, new application areas, and performance evaluation. *J Chem Inf Model*, 50(2):205–216, 2010.

[69] X. Glorot, A. Bordes and Y. Bengio. Deep sparse rectifier neural networks. In *Proc. of the International conference on Artificial Intelligence and Statistics (AISTATS 2011)*, pages 315–323, 2011.

[70] M. Gönen and E. Alpaydın. Multiple kernel learning algorithms. *J Mach Learn Res*, 12:2211–2268, 2011.

[71] S. Grimme. Towards first principles calculation of electron impact mass spectra of molecules. *Angew Chem Int Ed Engl*, 52:6306–6312, 2013.

[72] J. Gu, Y. Gui, L. Chen, G. Yuan, H.-Z. Lu and X. Xu. Use of natural products as chemical library for drug discovery and network pharmacology. *PLoS One*, 8(4): 1–10, 2013.

[73] R. Gugisch, A. Kerber, A. Kohnert, R. Laue, M. Meringer, C. Rücker and A. Wassermann. MOLGEN 5.0, a molecular structure generator. In *Advances in mathematical chemistry and applications*, pages 113–138. Elsevier, 2015.

[74] R. Guha, M. T. Howard, G. R. Hutchison, P. Murray-Rust, H. Rzepa, C. Steinbeck, J. Wegner, and E. L. Willighagen. The Blue Obelisk: Interoperability in chemical informatics. *J Chem Inf Model*, 46(3):991–998, 2006.

[75] C. Guijas, J. R. Montenegro-Burke, X. Domingo-Almenara, A. Palermo, B. Warth, G. Hermann, G. Koellensperger, T. Huan, W. Uritboonthai, A. E. Aisporna, *et al.* METLIN: A technology platform for identifying knowns and unknowns. *Analytical chemistry*, 90(5):3156–3164, 2018.

[76] J. Hastings, P. de Matos, A. Dekker, M. Ennis, B. Harsha, N. Kale, V. Muthukrishnan, G. Owen, S. Turner, M. Williams, and C. Steinbeck. The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Res*, 41(Database issue):D456–D463, 2013.

[77] D. Haussler. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.

[78] M. Heinonen, A. Rantanen, T. Mielikäinen, J. Kokkonen, J. Kiuru, R. A. Ketola and J. Rousu. FiD: A software for ab initio structural identification of product ions from tandem mass spectrometric data. *Rapid Commun Mass Spectrom*, 22(19):3043–3052, 2008.

[79] M. Heinonen, H. Shen, N. Zamboni and J. Rousu. Metabolite identification and molecular fingerprint prediction via machine learning. *Bioinformatics*, 28(18):2333–2341, 2012.

[80] S. R. Heller, A. McNaught, I. Pletnev, S. Stein and D. Tchekhovskoi. Inchi, the iupac international chemical identifier. *J Cheminform*, 7:23, 2015.

[81] P. Hering, A. Maaswinkel and K. Kompa. Photo-ionization mass spectrometry with psec UV-lasers. *Int J Mass Spectrom Ion Phys*, 46:273–276, 1983.

[82] S. Heuerding and J. T. Clerc. Simple tools for the computer-aided interpretation of mass spectra. *Chemom Intell Lab Syst*, 20(1):57–69, 1993.

[83] A. W. Hill and R. J. Mortishire-Smith. Automated assignment of high-resolution collisionally activated dissociation mass spectra using a systematic bond disconnection approach. *Rapid Commun Mass Spectrom*, 19(21):3111–3118, 2005.

[84] D. W. Hill, T. M. Kertesz, D. Fontaine, R. Friedman and D. F. Grant. Mass spectral metabonomics beyond elemental formula: Chemical database querying by matching experimental with computational fragmentation spectra. *Anal Chem*, 80(14):5574–5582, 2008.

[85] G. E. Hinton, S. Osindero and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput*, 18(7):1527–1554, 2006.

[86] T. Hofmann, B. Schölkopf and A. J. Smola. Kernel methods in machine learning. *Ann Stat*, pages 1171–1220, 2008.

[87] A. C. Hogenboom, J. A. Van Leerdam and P. De Voogt. Accurate mass screening and identification of emerging contaminants in environmental samples by liquid chromatography–hybrid linear ion trap orbitrap mass spectrometry. *J Chromatogr A*, 1216(3):510–519, 2009.

[88] J. Hollender, E. L. Schymanski, H. P. Singer and P. L. Ferguson. Nontarget screening with high resolution mass spectrometry in the environment: ready to go? *Environ Sci Technol*, 51(20):11505–11512, 2017. PMID: 28877430.

[89] H. Horai, M. Arita, S. Kanaya, Y. Nihei, T. Ikeda, K. Suwa, Y. Ojima, K. Tanaka, S. Tanaka, K. Aoshima, Y. Oda, Y. Kakazu, M. Kusano, T. Tohge, F. Matsuda, Y. Sawada, M. Y. Hirai, H. Nakanishi, K. Ikeda, N. Akimoto, T. Maoka, H. Takahashi, T. Ara, N. Sakurai, H. Suzuki, D. Shibata, S. Neumann, T. Iida, K. Tanaka, K. Funatsu, F. Matsuura, T. Soga, R. Taguchi, K. Saito, and T. Nishioka. MassBank: A public repository for sharing mass spectral data for life sciences. *J Mass Spectrom*, 45(7):703–714, 2010.

[90] F. Hufsky, K. Dührkop, F. Rasche, M. Chimani and S. Böcker. Fast alignment of fragmentation trees. *Bioinformatics*, 28(12):i265–i273, 2012. Proc. of *Intelligent Systems for Molecular Biology* (ISMB 2012).

[91] F. Hufsky, K. Scheubert and S. Böcker. New kids on the block: Novel informatics methods for natural product discovery. *Nat Prod Rep*, 31(6):807–817, 2014.

[92] C. C. Hughes and W. Fenical. Antibacterials from the sea. *Chem Eur J*, 16(42): 12512–12525, 2010.

[93] J. R. Idle and F. J. Gonzalez. Metabolomics. *Cell Metab*, 6(5):348–351, 2007.

[94] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad and R. G. Coleman. ZINC: a free tool to discover chemistry for biology. *J Chem Inf Model*, 52(7):1757–1768, 2012.

[95] N. Jaitly, M. E. Monroe, V. A. Petyuk, T. R. W. Clauss, J. N. Adkins and R. D. Smith. Robust algorithm for alignment of liquid chromatography-mass spectrometry analyses in an accurate mass and time tag data analysis pipeline. *Anal Chem*, 78 (21):7397–7409, 2006.

[96] T. Jebara, R. Kondor and A. Howard. Probability product kernels. *J Mach Learn Res*, 5:819–844, 2004.

[97] J. G. Jeffryes, R. L. Colastani, M. Elbadawi-Sidhu, T. Kind, T. D. Niehaus, L. J. Broadbelt, A. D. Hanson, O. Fiehn, K. E. J. Tyo, and C. S. Henry. MINEs: open access databases of computationally predicted enzyme promiscuity products for untargeted metabolomics. *J Cheminform*, 7:44, 2015.

[98] L. Käll, J. D. Storey, M. J. MacCoss and W. S. Noble. Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. *J Proteome Res*, 7(1):29–34, 2008.

[99] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: New developments in KEGG. *Nucleic Acids Res*, 34:D354–D357, 2006.

[100] M. Kanehisa, Y. Sato, M. Kawashima, M. Furumichi and M. Tanabe. KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Res*, 44(D1): D457–D462, 2016.

[101] L. J. Kangas, T. O. Metz, G. Isaac, B. T. Schrom, B. Ginovska-Pangovska, L. Wang, L. Tan, R. R. Lewis, and J. H. Miller. In silico identification software (ISIS): A machine learning approach to tandem mass spectral identification of lipids. *Bioinformatics*, 28(13):1705–1713, 2012.

[102] M. Katajamaa and M. Oresic. Processing methods for differential analysis of LC/MS profile data. *BMC Bioinf*, 6(1):179, 2005.

[103] E. Kenar, H. Franken, S. Forcisi, K. Wörmann, H.-U. Häring, R. Lehmann, P. Schmitt-Kopplin, A. Zell, and O. Kohlbacher. Automated label-free quantification of metabolites from liquid chromatography-mass spectrometry data. *Mol Cell Proteomics*, 13(1):348–359, 2014.

[104] S. Kern, K. Fenner, H. P. Singer, R. P. Schwarzenbach and J. Hollender. Identification of transformation products of organic contaminants in natural waters by computer-aided prediction and high-resolution mass spectrometry. *Environ Sci Technol*, 43 (18):7039–7046, 2009.

[105] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang, and S. H. Bryant. PubChem substance and compound databases. *Nucleic Acids Res*, 44:D1202–D1213, 2016.

[106] T. Kind and O. Fiehn. Metabolomic database annotations via query of elemental compositions: Mass accuracy is insufficient even at less than 1 ppm. *BMC Bioinf*, 7 (1):234, 2006.

[107] T. Kind and O. Fiehn. Seven golden rules for heuristic filtering of molecular formulas obtained by accurate mass spectrometry. *BMC Bioinf*, 8:105, 2007.

[108] T. Kind, K.-H. Liu, D. Y. Lee, B. DeFelice, J. K. Meissen and O. Fiehn. LipidBlast in silico tandem mass spectrometry database for lipid identification. *Nat Methods*, 10(8):755–758, 2013.

[109] J. Klekota and F. P. Roth. Chemical substructures that enrich for biological activity. *Bioinformatics*, 24(21):2518–2525, 2008.

[110] H. Kubinyi. Calculation of isotope distributions in mass spectrometry: A trivial solution for a non-trivial problem. *Anal Chim Acta*, 247:107–119, 1991.

[111] C. Kuhl, R. Tautenhahn, C. Böttcher, T. R. Larson and S. Neumann. CAMERA: An integrated strategy for compound spectra extraction and annotation of liquid chromatography/mass spectrometry data sets. *Anal Chem*, 84(1):283–289, 2012.

[112] T. Kwon, H. Choi, C. Vogel, A. I. Nesvizhskii and E. M. Marcotte. MSblender: A probabilistic approach for integrating peptide identifications from multiple database search engines. *J Proteome Res*, 10(7):2949–2958, 2011.

[113] E. Lange, R. Tautenhahn, S. Neumann and C. Gröpl. Critical assessment of alignment procedures for LC-MS proteomics and metabolomics measurements. *BMC Bioinf*, 9:375, 2008.

[114] I. Laponogov, N. Sadawi, D. Galea, R. Mirnezami, K. A. Veselkov and J. Wren. Chemdistiller: an engine for metabolite annotation in mass spectrometry. *Bioinformatics*, 1:7, 2018.

[115] S. Lazebnik, C. Schmid and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. of Conference on Computer vision and pattern recognition (CVPR 2006)*, volume 2, pages 2169–2178. IEEE, 2006.

[116] Y. LeCun, Y. Bengio and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[117] M. Letzel, Zs. Lipták, A. Pervukhin and S. Böcker. Sum formula generation of chemical compounds using isotope distribution abundances and exact masses: A comparison of two instruments, 2008.

[118] L. I. Levitsky, M. V. Ivanov, A. A. Lobas and M. V. Gorshkov. Unbiased false discovery rate estimation for shotgun proteomics based on the target-decoy approach. *J Proteome Res*, 16(2):393–397, 2016.

[119] Y. F. Li, R. J. Arnold, P. Radivojac and H. Tang. Protein identification problem from a Bayesian point of view. *Stat Interface*, 5(1):21–37, 2012.

[120] H.-T. Lin, C.-J. Lin and R. C. Weng. A note on Platt's probabilistic outputs for Support Vector Machines. *Machine Learning*, 68(3):267–276, 2007.

[121] J. L. Little, A. J. Williams, A. Pshenichnov and V. Tkachenko. Identification of "known unknowns" utilizing accurate mass data and ChemSpider. *J Am Soc Mass Spectrom*, 23(1):179–185, 2012.

[122] M. Loos, C. Gerber, F. Corona, J. Hollender and H. Singer. Accelerated isotope fine structure calculation using pruned transition trees. *Anal Chem*, 87(11):5738–5744, 2015.

[123] M. Ludwig, K. Dührkop and S. Böcker. Bayesian networks for mass spectrometric metabolite identication via molecular fingerprints. *Bioinformatics*, 34(13):i333–i340, 2018. Proc. of *Intelligent Systems for Molecular Biology* (ISMB 2018).

[124] Y. Ma, T. Kind, D. Yang, C. Leon and O. Fiehn. MS2Analyzer: A software for small molecule substructure annotations from accurate tandem mass spectra. *Anal Chem*, 86(21):10724–10731, 2014.

[125] A. Mayr, G. Klambauer, T. Unterthiner and S. Hochreiter. DeepTox: toxicity prediction using deep learning. *Front Environ Sci*, 3:80, 2016.

[126] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *B Math Biophys*, 5(4):115–133, 1943.

[127] L. A. McDonnell and R. M. Heeren. Imaging mass spectrometry. *Mass Spectrometry Reviews*, 26(4):606–643, 2007.

[128] K. F. Medzihradszky and R. J. Chalkley. Lessons in de novo peptide sequencing by tandem mass spectrometry. *Mass Spectrom Rev*, 34(1):43–63, 2015.

[129] S. Menchetti, F. Costa and P. Frasconi. Weighted decomposition kernels. In *Proc. of International conference on Machine learning (ICML 2005)*, pages 585–592. ACM, 2005.

[130] L. C. Menikarachchi, S. Cawley, D. W. Hill, L. M. Hall, L. Hall, S. Lai, J. Wilder, and D. F. Grant. MolFind: A software package enabling HPLC/MS-based identification of unknown chemical structures. *Anal Chem*, 84(21):9388–9394, 2012.

[131] M. Meringer, S. Reinker, J. Zhang and A. Muller. MS/MS data improves automated determination of molecular formulas by mass spectrometry. *MATCH-Commun Math Co*, 65:259–290, 2011.

[132] T. O. Metz, E. S. Baker, E. L. Schymanski, R. S. Renslow, D. G. Thomas, T. J. Causon, I. K. Webb, S. Hann, R. D. Smith, and J. G. Teeguarden. Integrating ion mobility spectrometry into mass spectrometry-based exposome measurements: what can it add and how far can it go? *Bioanalysis*, 9(1):81–98, 2017.

[133] M. Meusel, F. Hufsky, F. Panter, D. Krug, R. Müller and S. Böcker. Predicting the presence of uncommon elements in unknown biomolecules from isotope patterns. *Anal Chem*, 88(15):7556–7566, 2016.

[134] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE T Pattern Anal*, 27(10):1615–1630, 2005.

[135] B. L. Milman and I. K. Zhurkovich. Mass spectral libraries: A statistical review of the visible use. *TrAC, Trends Anal Chem*, 80:636 – 640, 2016.

[136] H. Mohimani, A. Gurevich, A. Mikheenko, N. Garg, L.-F. Nothias, A. Ninomiya, K. Takada, P. C. Dorrestein, and P. A. Pevzner. Dereplication of peptidic natural products through database search of mass spectra. *Nat Chem Biol*, 13(1):30, 2017.

[137] H. L. Morgan. The generation of a unique machine description for chemical structures – a technique developed at chemical abstracts service. *J Chem Doc*, 5(2):107–113, 1965.

[138] K. Morreel, Y. Saeys, O. Dima, F. Lu, Y. Van de Peer, R. Vanholme, J. Ralph, B. Vanholme, and W. Boerjan. Systematic structural characterization of metabolites in *Arabidopsis* via candidate substrate-product pair networks. *Plant Cell*, 26(3):929–945, 2014.

[139] L. Moruz and L. Käll. Peptide retention time prediction. *Mass Spectrom Rev*, 36(5): 615–623, 2017.

[140] A. Mrzic, P. Meysman, W. Bittremieux and K. Laukens. Automated recommendation of metabolite substructures from mass spectra using frequent pattern mining. Technical report, University of Antwerp, 2017.

[141] T. Muth, B. Y. Renard and L. Martens. Metaproteomic data analysis at a glance: advances in computational microbial community proteomics. *Expert Rev Proteomics*, 13(8):757–769, 2016.

[142] S. J. Nelson, W. D. Johnston and B. L. Humphreys. Relationships in medical subject headings. In C. A. Bean and R. Green, editors, *Relationships in the organization of knowledge*, pages 171–184. Kluwer Academic Publishers, 2001.

[143] A. I. Nesvizhskii. A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. *J Proteomics*, 73(11):2092–2123, 2010.

[144] A. G. Newsome and D. Nikolic. CASMI 2013: identification of small molecules by tandem mass spectrometry combined with database and literature mining. *Mass Spectrometry*, 3(Special_Issue_2):S0034–S0034, 2014.

[145] J. Ng, N. Bandeira, W.-T. Liu, M. Ghassemian, T. L. Simmons, W. H. Gerwick, R. Linington, P. C. Dorrestein, and P. A. Pevzner. Dereplication and de novo sequencing of nonribosomal peptides. *Nat Methods*, 6(8):596, 2009.

[146] D. D. Nguyen, C.-H. Wu, W. J. Moree, A. Lamsa, M. H. Medema, X. Zhao, R. G. Gavilan, M. Aparicio, L. Atencio, C. Jackson, J. Ballesteros, J. Sanchez, J. D. Watrous, V. V. Phelan, C. van de Wiel, R. D. Kersten, S. Mehnaz, R. De Mot, E. A. Shank, P. Charusanti, H. Nagarajan, B. M. Duggan, B. S. Moore, N. Bandeira, B. Ø. Palsson, K. Pogliano, M. Gutiérrez, and P. C. Dorrestein. MS/MS networking guided analysis of molecule and gene cluster families. *Proc Natl Acad Sci U S A*, 110(28): E2611–E2620, 2013.

[147] N. Nikolova and J. Jaworska. Approaches to measure chemical similarity – a review. *QSAR Comb. Sci.*, 22(9-10):1006–1026, 2003.

[148] T. Nishioka, T. Kasama, T. Kinumi, H. Makabe, F. Matsuda, D. Miura, M. Miyashita, T. Nakamura, K. Tanaka, and A. Yamamoto. Winners of CASMI2013: Automated tools and challenge data. *Mass Spectrom*, 3(special issue 2):S0039, 2014.

[149] H. Oberacher, F. Pitterl, E. Siapi, B. R. Steele, T. Letzel, S. Grosse, B. Poschner, F. Tagliaro, R. Gottardo, S. A. Chacko, *et al.* On the inter-instrument and the inter-laboratory transferability of a tandem mass spectral reference library. 3. focus on ion trap and upfront CID. *J Mass Spectrom*, 47(2):263–270, 2012.

[150] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch and G. R. Hutchison. Open Babel: An open chemical toolbox. *J Cheminform*, 3:33, 2011.

[151] S. Ojanperä, A. Pelander, M. Pelzing, I. Krebs, E. Vuori and I. Ojanperä. Isotopic pattern and accurate mass determination in urine drug screening by liquid chromatography/time-of-flight mass spectrometry. *Rapid Commun Mass Spectrom*, 20(7):1161–1167, 2006.

[152] D. B. Parker. *Learning logic: casting the cortex of the human brain in silicon.* PhD thesis, Massachusetts Institute of Technology, 1985.

[153] J. E. Peironcely, M. Rojas-Chertó, D. Fichera, T. Reijmers, L. Coulier, J.-L. Faulon and T. Hankemeier. OMG: open molecule generator. *J Cheminform*, 4(1):21, 2012.

[154] K. Peters, A. Worrich, A. Weinhold, O. Alka, G. Balcke, C. Birkemeyer, H. Bruelheide, O. W. Calf, S. Dietz, K. Dührkop, E. Gaquerel, U. Heinig, M. Kücklich, M. Macel, C. Müller, Y. Poeschl, G. Pohnert, C. Ristok, V. M. Rodríguez, C. Ruttkies, M. Schuman, R. Schweiger, N. Shahaf, C. Steinbeck, M. Tortosa, H. Treutler, N. Ueberschaar, P. Velasco, B. M. Weiß, A. Widdig, S. Neumann, and N. M. van Dam. Current challenges in plant eco-metabolomics. *Int J Mol Sci*, 19(5), 2018.

[155] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, chapter 5. MIT Press, Cambridge, Massachusetts, 2000.

[156] I. Pletnev, A. Erin, A. McNaught, K. Blinov, D. Tchekhovskoi and S. Heller. InChIKey collision resistance: an experimental testing. *J Cheminf*, 4(1):39, 2012.

[157] T. Pluskal, S. Castillo, A. Villar-Briones and M. Oresic. MZmine 2: Modular framework for processing, visualizing, and analyzing mass spectrometry-based molecular profile data. *BMC Bioinf*, 11:395, 2010.

[158] T. Pluskal, T. Uehara and M. Yanagida. Highly accurate chemical formula prediction tool utilizing high-resolution mass spectra, MS/MS fragmentation, heuristic rules, and isotope pattern matching. *Anal Chem*, 84(10):4396–4403, 2012.

[159] C. Poultney, S. Chopra, Y. L. Cun *et al.* Efficient learning of sparse representations with an energy-based model. In *Proc. of Advances in neural information processing systems (NIPS 2007)*, pages 1137–1144, 2007.

[160] R. A. Quinn, L.-F. Nothias, O. Vining, M. Meehan, E. Esquenazi and P. C. Dorrestein. Molecular networking as a drug discovery, drug metabolism, and precision medicine strategy. *Trends Pharmacol Sci*, 38(2):143–154, 2017.

[161] A. Rakotomamonjy, F. R. Bach, S. Canu and Y. Grandvalet. SimpleMKL. *J Mach Learn Res*, 9(Nov):2491–2521, 2008.

[162] G. M. Randazzo, D. Tonoli, S. Hambye, D. Guillarme, F. Jeanneret, A. Nurisso, L. Goracci, J. Boccard, and S. Rudaz. Prediction of retention time in reversed-phase liquid chromatography as a tool for steroid identification. *Anal Chim Acta*, 916:8–16, 2016.

[163] F. Rasche. *Identification of Small Molecules using Mass Spectrometry: A fully automated pipeline proposing similar molecules and compound classes.* PhD thesis, Friedrich-Schiller-Universität Jena, Jena, Germany, 2013.

[164] F. Rasche, A. Svatoš, R. K. Maddula, C. Böttcher and S. Böcker. Computing fragmentation trees from tandem mass spectrometry data. *Anal Chem*, 83(4):1243–1251, 2011.

[165] F. Rasche, K. Scheubert, F. Hufsky, T. Zichner, M. Kai, A. Svatoš and S. Böcker. Identifying the unknowns by aligning fragmentation trees. *Anal Chem*, 84(7):3417–3426, 2012.

[166] I. Rauf, F. Rasche, F. Nicolas and S. Böcker. Finding maximum colorful subtrees in practice. *J Comput Biol*, 20(4):1–11, 2013.

[167] J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J Comput Aided Mol Des*, 16(7):521–533, 2002.

[168] F. Rebentrost and A. Ben-Shaul. On the fragmentation of benzene by multiphotoionization. *J Chem Phys*, 74(6):3255–3264, 1981.

[169] A. L. Richards, A. E. Merrill and J. J. Coon. Proteome sequencing goes deep. *Curr Opin Chem Biol*, 24:11–17, 2015.

[170] L. Ridder, J. J. J. van der Hooft, S. Verhoeven, R. C. H. de Vos, R. J. Bino and J. Vervoort. Automatic chemical structure annotation of an LC-MS(n) based metabolic profile from green tea. *Anal Chem*, 85(12):6033–6040, 2013.

[171] A. L. Rockwood. Relationship of Fourier transforms to isotope distribution calculations. *Rapid Commun Mass Spectrom*, 9:103–105, 1995.

[172] A. L. Rockwood and P. Haimi. Efficient calculation of accurate masses of isotopic peaks. *J Am Soc Mass Spectrom*, 17(3):415–419, 2006.

[173] A. L. Rockwood, S. L. Van Orden and R. D. Smith. Rapid calculation of isotope distributions. *Anal Chem*, 67:2699–2704, 1995.

[174] D. Rogers and M. Hahn. Extended-connectivity fingerprints. *J Chem Inf Model*, 50 (5):742–754, 2010.

[175] F. B. Rogers. Communications to the editor. *Bull Med Libr Assoc*, 51(1):114–116, 1963.

[176] S. Rogers, R. A. Scheltema, M. Girolami and R. Breitling. Probabilistic assignment of formulas to mass peaks in metabolomics experiments. *Bioinformatics*, 25(4):512–518, 2009.

[177] M. Rojas-Chertó, P. T. Kasper, E. L. Willighagen, R. J. Vreeken, T. Hankemeier and T. H. Reijmers. Elemental composition determination based on MS^n.. *Bioinformatics*, 27:2376–2383, 2011.

[178] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev*, 65(6):386, 1958.

[179] H. L. Röst, T. Sachsenberg, S. Aiche, C. Bielow, H. Weisser, F. Aicheler, S. Andreotti, H.-C. Ehrlich, P. Gutenbrunner, E. Kenar, X. Liang, S. Nahnsen, L. Nilse, J. Pfeuffer, G. Rosenberger, M. Rurik, U. Schmitt, J. Veit, M. Walzer, D. Wojnar, W. E. Wolski, O. Schilling, J. S. Choudhary, L. Malmström, R. Aebersold, K. Reinert, and O. Kohlbacher. OpenMS: a flexible open-source software platform for mass spectrometry data analysis. *Nat Methods*, 13(9):741–748, 2016.

[180] D. E. Rumelhart, G. E. Hinton and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[181] C. Ruttkies, E. L. Schymanski, S. Wolf, J. Hollender and S. Neumann. MetFrag relaunched: incorporating strategies beyond in silico fragmentation. *J Cheminform*, 8:3, 2016.

[182] K. Scheubert, F. Hufsky, F. Rasche and S. Böcker. Computing fragmentation trees from metabolite multiple mass spectrometry data. *J Comput Biol*, 18(11):1383–1397, 2011.

[183] K. Scheubert, F. Hufsky and S. Böcker. Multiple mass spectrometry fragmentation trees revisited: Boosting performance and quality. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2014)*, volume 8701 of *Lect Notes Comput Sci*, pages 217–231. Springer, Berlin, 2014.

[184] K. Scheubert, F. Hufsky, D. Petras, M. Wang, L.-F. Nothias, K. Dührkop, N. Bandeira, P. C. Dorrestein, and S. Böcker. Significance estimation for large scale metabolomics annotations by spectral matching. *Nat Commun*, 8:1494, 2017.

[185] J. Schmidhuber. Deep learning in neural networks: an overview. *Neural networks*, 61:85–117, 2015.

[186] B. M. Schmidt, D. M. Ribnicky, P. E. Lipsky and I. Raskin. Revisiting the ancient concept of botanical therapeutics. *Nat Chem Biol*, 3(7):360–366, 2007.

[187] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[188] J. E. Schollée, E. L. Schymanski, M. A. Stravs, R. Gulde, N. S. Thomaidis and J. Hollender. Similarity of high-resolution tandem mass spectrometry spectra of structurally related micropollutants and transformation products. *J Am Soc Mass Spectr*, 28(12):2692–2704, 2017.

[189] V. Schwämmle, T. Verano-Braga and P. Roepstorff. Computational and statistical methods for high-throughput analysis of post-translational modifications of proteins. *J Proteomics*, 129:3–15, 2015.

[190] E. L. Schymanski and S. Neumann. The critical assessment of small molecule identification (CASMI): Challenges and solutions. *Metabolites*, 3(3):517–538, 2013.

[191] E. L. Schymanski, C. Ruttkies, M. Krauss, C. Brouard, T. Kind, K. Dührkop, F. R. Allen, A. Vaniya, D. Verdegem, S. Böcker, J. Rousu, H. Shen, H. Tsugawa, T. Sajed, O. Fiehn, B. Ghesquière, and S. Neumann. Critical Assessment of Small Molecule Identification 2016: Automated methods. *J Cheminf*, 9:22, 2017.

[192] P. K. Sen. Estimates of the regression coefficient based on Kendall's tau. *J Am Stat Assoc*, 63:1379–1389, 1968.

[193] J. Senior. Partitions and their representative graphs. *Amer J Math*, 73(3):663–689, 1951.

[194] N. Shahaf, I. Rogachev, U. Heinig, S. Meir, S. Malitsky, M. Battat, H. Wyner, S. Zheng, R. Wehrens, and A. Aharoni. The WEIZMASS spectral library for high-confidence metabolite identification. *Nat Commun*, 7:12423, 2016.

[195] M. T. Sheldon, R. Mistrik and T. R. Croley. Determination of ion structures in structurally related compounds using precursor ion fingerprinting. *J Am Soc Mass Spectrom*, 20(3):370–376, 2009.

[196] H. Shen, K. Dührkop, S. Böcker and J. Rousu. Metabolite identification through multiple kernel learning on fragmentation trees. *Bioinformatics*, 30(12):i157–i164, 2014. Proc. of *Intelligent Systems for Molecular Biology* (ISMB 2014).

[197] H. Shen, S. Szedmak, C. Brouard and J. Rousu. *Soft Kernel Target Alignment for Two-Stage Multiple Kernel Learning*, pages 427–441. Springer International Publishing, Cham, 2016.

[198] Z. Shi. *Learning to predict the sites of metabolism and metabolic endpoints*. PhD thesis, University of Alberta, 2016.

[199] Y. Shinbo, Y. Nakamura, M. Altaf-Ul-Amin, H. Asahi, K. Kurokawa, M. Arita, K. Saito, D. Ohta, D. Shibata, and S. Kanaya. KNApSAcK: A comprehensive species-metabolite relationship database. In K. Saito, R. A. Dixon and L. Willmitzer, editors, *Plant Metabolomics*, volume 57 of *Biotechnology in Agriculture and Forestry*, pages 165–181. Springer-Verlag, 2006.

[200] P. R. Spackman, B. Bohman, A. Karton and D. Jayatilaka. Quantum chemical electron impact mass spectrum prediction for de novo structure elucidation: assessment against experimental reference data and comparison to competitive fragmentation modeling. *Int J Quantum Chem*, 118(2), 2018.

[201] A. Sreekumar, L. M. Poisson, T. M. Rajendiran, A. P. Khan, Q. Cao, J. Yu, B. Laxman, R. Mehra, R. J. Lonigro, Y. Li, M. K. Nyati, A. Ahsan, S. Kalyana-Sundaram, B. Han, X. Cao, J. Byun, G. S. Omenn, D. Ghosh, S. Pennathur, D. C. Alexander, A. Berger, J. R. Shuster, J. T. Wei, S. Varambally, C. Beecher, and A. M. Chinnaiyan. Metabolomic profiles delineate potential role for sarcosine in prostate cancer progression. *Nature*, 457(7231):910–914, 2009.

[202] M. Stahl, H. Mauser, M. Tsui and N. R. Taylor. A robust clustering method for chemical structures. *J Med Chem*, 48(13):4358–4366, 2005. PMID: 15974588.

[203] S. E. Stein. Mass spectral reference libraries: An ever-expanding resource for chemical identification. *Anal Chem*, 84(17):7274–7282, 2012.

[204] S. E. Stein and D. R. Scott. Optimization and testing of mass spectral library search algorithms for compound identification. *J Am Soc Mass Spectrom*, 5(9):859–866, 1994.

[205] C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann and E. Willighagen. The Chemistry Development Kit (CDK): An open-source Java library for chemo- and bioinformatics. *J Chem Inf Comput Sci*, 43:493–500, 2003.

[206] M. A. Stravs, E. L. Schymanski, H. P. Singer and J. Hollender. Automatic recalibration and processing of tandem mass spectra using formula annotation. *J Mass Spectrom*, 48(1):89–99, 2013.

[207] R. Tautenhahn, C. Böttcher and S. Neumann. Highly sensitive feature detection for high resolution lc/ms. *BMC Bioinformatics*, 9(1):504, 2008.

[208] R. Tautenhahn, G. J. Patti, D. Rinehart and G. Siuzdak. XCMS Online: A web-based platform to process untargeted metabolomic datas. *Anal Chem*, 84(11):5035–5039, 2012.

[209] M. The, A. Tasnim and L. Käll. How to talk about protein-level false discovery rates in shotgun proteomics. *Proteomics*, 16(18):2461–2469, 2016.

[210] H. Theil. A rank-invariant method of linear and polynomial regression analysis I. *Proc R Neth Acad Arts Sci*, 53:386–392, 1950.

[211] R. R. Tice, C. P. Austin, R. J. Kavlock and J. R. Bucher. Improving the human hazard characterization of chemicals: a Tox21 update. *Environ Health Perspect*, 121 (7):756–765, 2013.

[212] H. Tsugawa, T. Kind, R. Nakabayashi, D. Yukihira, W. Tanaka, T. Cajka, K. Saito, O. Fiehn, and M. Arita. Hydrogen rearrangement rules: Computational ms/ms fragmentation and structure elucidation using MS-FINDER software. *Anal Chem*, 88(16):7946–7958, 2016.

[213] K. Uppal, D. I. Walker, K. Liu, S. Li, Y.-M. Go and D. P. Jones. Computational metabolomics: a framework for the million metabolome. *Chem Res Toxicol*, 29(12): 1956–1975, 2016.

[214] D. Valkenborg, I. Mertens, F. Lemière, E. Witters and T. Burzykowski. The isotopic distribution conundrum. *Mass Spectrom Rev*, 31(1):96–109, 2012.

[215] J. J. J. van der Hooft, J. Wandy, M. P. Barrett, K. E. V. Burgess and S. Rogers. Topic modeling for untargeted substructure exploration in metabolomics. *Proc Natl Acad Sci USA*, 113(48):13738–13743, 2016.

[216] A. Vaniya and O. Fiehn. Using fragmentation trees and mass spectral trees for identifying unknown compounds in metabolomics. *Trends Anal Chem*, 69:52–61, 2015.

[217] V. Vapnik. Pattern recognition using generalized portrait method. *Autom Remote Control*, 24:774–780, 1963.

[218] V. Vapnik. A note one class of perceptrons. *Autom Remote Control*, 1964.

[219] V. N. Vapnik. An overview of statistical learning theory. *IEEE Trans Neural Netw*, 10(5):988–999, 1999.

[220] K. Varmuza, M. Karlovits and W. Demuth. Spectral similarity versus structural similarity: infrared spectroscopy. *Anal Chim Acta*, 490(1-2):313–324, 2003.

[221] C. L. Ventola. The antibiotic resistance crisis: part 1: causes and threats. *Pharmacy and Therapeutics*, 40(4):277, 2015.

[222] D. Verdegem, D. Lambrechts, P. Carmeliet and B. Ghesquière. Improved metabolite identification with MIDAS and MAGMa through MS/MS spectral dataset-driven parameter optimization. *Metabolomics*, 12(6):1–16, 2016.

[223] M. Vinaixa, E. L. Schymanski, S. Neumann, M. Navarro, R. M. Salek and O. Yanes. Mass spectral databases for LC/MS- and GC/MS-based metabolomics: State of the field and future prospects. *Trends Anal Chem*, 78:23–35, 2016.

[224] C.-W. von der Lieth, T. Lütteke and M. Frank. The role of informatics in glycobiology research with special emphasis on automatic interpretation of MS spectra. *BBA-Gen Subjects*, 1760(4):568–577, 2006.

[225] J. Wandy, Y. Zhu, J. J. J. van der Hooft, R. Daly, M. P. Barrett and S. Rogers. Ms2lda.org: web-based topic modelling for substructure discovery in mass spectrometry. *Bioinformatics*, 34(2):317–318, 2018.

[226] M. Wang, G. Audi, A. Wapstra, F. Kondev, M. MacCormick, X. Xu and B. Pfeiffer. The AME2012 atomic mass evaluation (II). tables, graphs and references. *Chinese Physics C*, 36(12):1603–2014, 2012.

[227] M. Wang, J. J. Carver, V. V. Phelan, L. M. Sanchez, N. Garg, Y. Peng, D. D. Nguyen, J. Watrous, C. A. Kapono, T. Luzzatto-Knaan, C. Porto, A. Bouslimani, A. V. Melnik, M. J. Meehan, W.-T. Liu, M. Crüsemann, P. D. Boudreau, E. Esquenazi, M. Sandoval-Calderón, R. D. Kersten, L. A. Pace, R. A. Quinn, K. R. Duncan, C.-C. Hsu, D. J. Floros, R. G. Gavilan, K. Kleigrewe, T. Northen, R. J. Dutton, D. Parrot, E. E. Carlson, B. Aigle, C. F. Michelsen, L. Jelsbak, C. Sohlenkamp, P. Pevzner, A. Edlund, J. McLean, J. Piel, B. T. Murphy, L. Gerwick, C.-C. Liaw, Y.-L. Yang, H.-U. Humpf, M. Maansson, R. A. Keyzers, A. C. Sims, A. R. Johnson, A. M. Sidebottom, B. E. Sedio, A. Klitgaard, C. B. Larson, C. A. Boya P, D. Torres-Mendoza, D. J. Gonzalez, D. B. Silva, L. M. Marques, D. P. Demarque, E. Pociute,

E. C. O'Neill, E. Briand, E. J. N. Helfrich, E. A. Granatosky, E. Glukhov, F. Ryffel, H. Houson, H. Mohimani, J. J. Kharbush, Y. Zeng, J. A. Vorholt, K. L. Kurita, P. Charusanti, K. L. McPhail, K. F. Nielsen, L. Vuong, M. Elfeki, M. F. Traxler, N. Engene, N. Koyama, O. B. Vining, R. Baric, R. R. Silva, S. J. Mascuch, S. Tomasi, S. Jenkins, V. Macherla, T. Hoffman, V. Agarwal, P. G. Williams, J. Dai, R. Neupane, J. Gurr, A. M. C. Rodríguez, A. Lamsa, C. Zhang, K. Dorrestein, B. M. Duggan, J. Almaliti, P.-M. Allard, P. Phapale, L.-F. Nothias, T. Alexandrov, M. Litaudon, J.-L. Wolfender, J. E. Kyle, T. O. Metz, T. Peryea, D.-T. Nguyen, D. VanLeer, P. Shinn, A. Jadhav, R. Müller, K. M. Waters, W. Shi, X. Liu, L. Zhang, R. Knight, P. R. Jensen, B. Ø. Palsson, K. Pogliano, R. G. Linington, M. Gutiérrez, N. P. Lopes, W. H. Gerwick, B. S. Moore, P. C. Dorrestein, and N. Bandeira. Sharing and community curation of mass spectrometry data with Global Natural Products Social molecular networking. *Nat Biotechnol*, 34(8):828–837, 2016.

[228] Y. Wang, J. Xiao, T. O. Suzek, J. Zhang, J. Wang and S. H. Bryant. PubChem: A public information system for analyzing bioactivities of small molecules. *Nucleic Acids Res*, 37(Web Server issue):W623–W633, 2009.

[229] Y. Wang, G. Kora, B. P. Bowen and C. Pan. MIDAS: A database-searching algorithm for metabolite identification in metabolomics. *Anal Chem*, 86(19):9496–9503, 2014.

[230] P. Waridel, J.-L. Wolfender, K. Ndjoko, K. R. Hobby, H. J. Major and K. Hostettmann. Evaluation of quadrupole time-of-flight tandem mass spectrometry and ion-trap multiple-stage mass spectrometry for the differentiation of c-glycosidic flavonoid isomers. *J Chromatogr A*, 926(1):29–41, 2001.

[231] J. Watrous, P. Roach, T. Alexandrov, B. S. Heath, J. Y. Yang, R. D. Kersten, M. van der Voort, K. Pogliano, H. Gross, J. M. Raaijmakers, B. S. Moore, J. Laskin, N. Bandeira, and P. C. Dorrestein. Mass spectral molecular networking of living microbial colonies. *Proc Natl Acad Sci U S A*, 109(26):E1743–E1752, 2012.

[232] R. J. M. Weber, E. Li, J. Bruty, S. He and M. R. Viant. MaConDa: A publicly accessible mass spectrometry contaminants database. *Bioinformatics*, 28(21):2856–2857, 2012.

[233] D. Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J Chem Inf Comput Sci*, 28(1):31–36, 1988.

[234] P. J. Werbos. *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.

[235] W. T. J. White, S. Beyer, K. Dührkop, M. Chimani and S. Böcker. Speedy colorful subtrees. In *Proc. of Computing and Combinatorics Conference (COCOON 2015)*, volume 9198 of *Lect Notes Comput Sci*, pages 310–322. Springer, Berlin, 2015.

[236] P. Willett. Similarity-based virtual screening using 2D fingerprints. *Drug Discov Today*, 11(23-24):1046–1053, 2006.

[237] P. Willett, J. M. Barnard and G. M. Downs. Chemical similarity searching. *J Chem Inf Comput Sci*, 38(6):983–996, 1998.

[238] E. L. Willighagen, J. W. Mayfield, J. Alvarsson, A. Berg, L. Carlsson, N. Jeliazkova, S. Kuhn, T. Pluskal, M. Rojas-Chertó, O. Spjuth, G. Torrance, C. T. Evelo, R. Guha, and C. Steinbeck. The Chemistry Development Kit (CDK) v2.0: atom typing, depiction, molecular formulas, and substructure searching. *J Cheminf*, 9(1):33, 2017.

[239] M. Wink. Evolution of secondary metabolites from an ecological and molecular phylogenetic perspective. *Phytochemistry*, 64(1):3–19, 2003.

[240] D. S. Wishart, Y. D. Feunang, A. Marcu, A. C. Guo, K. Liang, R. Vázquez-Fresno, T. Sajed, D. Johnson, C. Li, N. Karu, Z. Sayeeda, E. Lo, N. Assempour, M. Berjanskii, S. Singhal, D. Arndt, Y. Liang, H. Badran, J. Grant, A. Serra-Cayuela, Y. Liu, R. Mandal, V. Neveu, A. Pon, C. Knox, M. Wilson, C. Manach, and A. Scalbert. HMDB 4.0: the human metabolome database for 2018. *Nucleic Acids Research*, 46(D1):D608–D617, 2018.

[241] S. Wolf, S. Schmidt, M. Müller-Hannemann and S. Neumann. In silico fragmentation for computer assisted identification of metabolite mass spectra. *BMC Bioinf*, 11:148, 2010.

[242] World Health Organization *et al. Antimicrobial resistance: global report on surveillance*. World Health Organization, 2014.

[243] J. Y. Yang, L. M. Sanchez, C. M. Rath, X. Liu, P. D. Boudreau, N. Bruns, E. Glukhov, A. Wodtke, R. de Felicio, A. Fenner, W. R. Wong, R. G. Linington, L. Zhang, H. M. Debonsi, W. H. Gerwick, and P. C. Dorrestein. Molecular networking as a dereplication strategy. *J Nat Prod*, 76:1686–1699, 2013.

[244] N. Zhang, R. Aebersold and B. Schwikowski. ProbID: A probabilistic algorithm to identify peptides through sequence database searching using tandem mass spectral data. *Proteomics*, 2(10):1406–1412, 2002.

[245] R. Zubarev and M. Mann. On the proper use of mass accuracy in proteomics. *Mol Cell Proteomics*, 6(3):377–381, 2007.

A Appendix

Table A.1: Features which are used to train a linear support vector machine that discriminates between biomolecular formulas and random decompositions.

Binary features, encoded as -1 (false) and 1 (true).	
RDBE is zero	
contains oxygen and sulphur	contains oxygen and phosphor
contains phosphor and sulphur	contains bromine and chlorine
Real valued features.	
RDBE	*RDBE* / $mass^{2/3}$
mass	log *mass*
Elemental composition features.	
#C	log #C
#H	log #H
#N	log #N
#O	log #O
#P	log #P
#S	log #S
#Cl	log #Cl
#Br	log #Br
#I	log #I
#F	log #F
Element ratio features.	
Hetero-to-carbon ratio	Hetero-minus-oxygen-to-carbon ratio
(#N + #O) / (#C + 0.8)	(#F + #I + #Cl + #Br) / (#C + 0.8)
#H / (#C + 0.8)	#P / (#O + #S + 0.25)
#O / (#C + 0.8)	#N / (#C + 0.8)
#S / (#C + 0.8)	#P / (#C + 0.8)
#Cl / (#C + 0.8)	#Br / (#C + 0.8)
#I / (#C + 0.8)	#F / (#C + 0.8)
#N / (#O + 0.8)	#N / (#C + #O + 0.8)
Probability Distribution Features For each of the following properties we fit a probability distribution. We estimate parameters of the distribution using maximum likelihood on the biological database. We evaluate the probability density function of this distribution at the property value and take the logarithm of the density as feature. Furthermore, we define a mixture of an uniform distribution for 99 % of the biomolcules with smallest property value and an exponential distribution fitted to the remaining 1 % with highest property value. We define a second feature which is the logarithm of the density function of this mixed distribution evaluated at the property value.	
RDBE with log-normal distribution	
RDBE / $mass^{2/3}$ with normal distribution	
Hetero-to-carbon ratio with Pareto distribution	
Hetero-minus-oxygen-to-carbon ratio with Pareto distribution	
(#N + #O) / (#C + 0.8) with Pareto distribution	
(#F + #I + #Cl + #Br) / (#C + 0.8) with Pareto distribution	
#H / (#C + 0.8) with log-normal distribution	
RDBE / $\left(\min(\lfloor\frac{\#C}{5}\rfloor, \lfloor\frac{\#C}{4}\rfloor) + 0.8\right)$ with log-normal distribution	

Table A.2: Priors $P_{\text{loss-comm}}(l)$ for *common losses* l. Entry "mass" is the exact theoretical mass of the loss. Entry "known" indicates whether the loss was included in the expert-curated common loss lists in A [25], B [164], or C [165]. Entry "total" indicate the (rounded) frequency of the loss in the trees computed from the dataset, weighted by the maximum peak intensity of the two peaks that are responsible for this loss. Entries "expected" is the weighted frequency we would expect from the loss mass prior, and $P_{\text{loss-comm}}$ is the common loss prior after correcting for the loss mass prior. (*) Losses H and H_2 can be interpreted as artefacts of the loss mass prior, see text for details. (**) $C_{10}H_9NO_3S$, $C_{12}H_8ClNS$ and $C_{11}H_{10}Cl_2N_2O$ are artefacts, stemming from either their high mass or the small number of chlorine-containing compounds in the datasets.

				Intensity GNPS		Intensity Agilent		
Mol. formula	Mass	Loss name	Known	total	expected	total	expected	$P_{\text{loss-comm}}$
H*	1.0078	Hydrogen radical		110	0.00	77	0.00	*
H_2*	2.0157	Hydrogen	A,B	1 799	0.00	890	0.00	*
CH_2	14.0157	Methylene		33	17.47	71	37.35	1.92
CH_3	15.0235	Methyl	A	3 231	46.48	1 481	21.31	69.53
CH_4	16.0313	Methane	A,B,C	2 011	75.23	929	34.76	26.73
H_3N	17.0265	Ammonia	A,B,C	1 409	62.73	1 481	65.92	22.47
H_2O	18.0106	Water	A,B,C	5 548	85.53	4 014	61.88	64.87
HF	20.0062	Hydrogen fluoride		266	13.43	365	18.36	19.88
C_2H_2	26.0157	Ethine	B,C	2 434	133.98	2 324	127.90	18.17
CHN	27.0109	Hydrogen cyanide		1 117	139.90	1 078	134.94	7.99
CO	27.9949	Carbon monoxide	B,C	4 232	177.14	2 614	109.45	23.89
C_2H_4	28.0313	Ethene	A,B,C	483	87.19	1 108	199.82	5.55
CH_3N	29.0265	Methyleneimine	B	347	158.43	305	139.34	2.19
S	31.9721	Sulfur	B,C	79	38.60	179	87.07	2.06
CH_4O	32.0262	Methyl esters		202	127.42	341	214.18	1.59
Cl	34.9689	Chlorine		296	45.18	394	60.27	6.55
HCl	35.9767	Hydrogen chloride		462	45.88	613	60.95	10.07
C_2H_2O	42.0106	Ketene	B,C	811	246.67	584	177.75	3.29
C_3H_6	42.0470	Propene		207	101.85	656	322.40	2.03
C_2H_5N	43.0422	Aminoethylene		332	177.18	454	242.22	1.88
CO_2	43.9898	Carbon dioxide	B,C	281	199.41	215	153.06	1.41
Br	78.9183	Bromine		20	0.91	95	4.23	22.51
HBr	79.9262	Hydrogen bromide		9	0.63	65	4.38	14.98
HO_3P	79.9663	Metaphosphoric acid	B,C	3	0.78	25	6.55	3.93
HO_2PS	95.9435	Phosphenothioic acid		0	0.11	26	4.60	5.65
I	126.9045	Iodine		29	0.25	60	0.52	116.53
HI	127.9123	Hydrogen iodide		11	0.15	45	0.61	74.61
CIO	154.8994	Iodomethanone		0	0.04	3	0.32	10.28
$C_{10}H_9NO_3S$**	223.0303			20	1.12	5	0.30	18.54
$C_{12}H_8ClNS$**	233.0066	2-Chlorophenothiazine		1	0.06	25	0.83	30.72
I_2	253.8089	Iodine		0	0.00	10	0.03	357.31
$C_{11}H_{10}Cl_2N_2O$**	256.0170			3	0.12	9	0.40	24.93

Table A.3: Priors $P_{\text{frag-comm}}(f)$ for *common fragments* f. Entry "ion mass" is the exact theoretical mass of the protonated fragment. Entries "GNPS/Agilent" indicate the total sum of the peak intensities and total peak count of the fragment in the two datasets. Note that a particular fragment can be very common, yet have relatively small sum of peak intensities, because fragments peaks are consistently of small intensity.

Molecular formula			Total intensity		Total count		
protonated	neutral	Ion mass	GNPS	Agilent	GNPS	Agilent	$P_{\text{frag-comm}}$
$C_3H_6N^+$	C_3H_5N	56.0495	0.00	93.63	0	392	2.40
$C_3H_8N^+$	C_3H_7N	58.0651	0.67	100.53	4	323	2.59
$C_5H_5^+$	C_5H_4	65.0386	0.00	83.35	0	530	2.14
$C_4H_8N^+$	C_4H_7N	70.0651	7.38	56.00	8	313	1.62
$C_4H_{10}N^+$	C_4H_9N	72.0808	0.00	72.92	0	179	1.87
$C_6H_5^+$	C_6H_4	77.0386	1.00	139.35	3	720	3.60
$C_6H_7^+$	C_6H_6	79.0542	0.52	69.85	3	514	1.80
$C_5H_{12}N^+$	$C_5H_{11}N$	86.0964	0.92	71.08	5	128	1.85
$C_7H_7^+$	C_7H_6	91.0542	60.61	252.97	300	720	8.04
$C_6H_6N^+$	C_6H_5N	92.0495	3.92	76.12	31	185	2.05
$C_6H_9O^+$	C_6H_8O	97.0648	10.95	58.00	37	86	1.77
$C_6H_{12}N^+$	$C_6H_{11}N$	98.0964	8.73	74.93	66	139	2.14
$C_8H_7^+$	C_8H_6	103.0542	64.49	34.61	562	241	2.54
$C_7H_5O^+$	C_7H_4O	105.0335	50.43	47.64	178	100	2.51
$C_8H_9^+$	C_8H_8	105.0699	108.25	104.51	580	352	5.45
$C_7H_7O^+$	C_7H_6O	107.0491	62.41	53.68	320	187	2.98
$C_8H_{11}^+$	C_8H_{10}	107.0855	35.23	29.89	171	120	1.67
$C_6H_6NO^+$	C_6H_5NO	108.0444	23.05	48.66	64	76	1.84
$C_7H_9O^+$	C_7H_8O	109.0648	37.15	53.40	161	107	2.32
$C_9H_7^+$	C_9H_6	115.0542	73.68	43.35	618	262	3.00
$C_9H_9^+$	C_9H_8	117.0699	61.67	46.66	371	206	2.78
$C_9H_{11}^+$	C_9H_{10}	119.0855	51.94	40.94	265	190	2.38
$C_8H_9O^+$	C_8H_8O	121.0648	125.09	72.05	394	201	5.05
$C_{10}H_8^+$	$C_{10}H_7$	128.0621	51.79	13.98	305	98	1.69
$C_{10}H_9^+$	$C_{10}H_8$	129.0699	60.60	24.99	425	163	2.19
$C_9H_8N^+$	C_9H_7N	130.0651	75.54	31.39	343	159	2.74
$C_{10}H_{11}^+$	$C_{10}H_{10}$	131.0855	61.12	34.37	277	144	2.45
$C_9H_{10}N^+$	C_9H_9N	132.0808	58.68	21.70	216	99	2.06
$C_8H_7O_2^+$	$C_8H_6O_2$	135.0441	40.37	22.03	176	53	1.60
$C_9H_{11}O^+$	$C_9H_{10}O$	135.0804	42.95	32.87	221	121	1.94
$C_{11}H_{11}^+$	$C_{11}H_{10}$	143.0855	54.61	23.88	288	118	2.01
$C_{10}H_{10}N^+$	$C_{10}H_9N$	144.0808	61.59	20.67	220	99	2.11
$C_{11}H_{13}^+$	$C_{11}H_{12}$	145.1012	57.60	28.15	219	110	2.20
$C_9H_8NO^+$	C_9H_7NO	146.0600	62.09	7.40	242	52	1.78
$C_{10}H_{11}O^+$	$C_{10}H_{10}O$	147.0804	67.16	33.97	247	107	2.59
$C_{11}H_{11}O^+$	$C_{11}H_{10}O$	159.0804	47.36	17.07	230	84	1.65
$C_{10}H_{10}NO^+$	$C_{10}H_9NO$	160.0757	58.53	18.10	221	40	1.96
$C_{13}H_9^+$	$C_{13}H_8$	165.0699	54.17	36.32	255	163	2.32
$C_{13}H_{11}^+$	$C_{13}H_{10}$	167.0855	28.57	36.85	123	65	1.68
$C_{12}H_{11}O^+$	$C_{12}H_{10}O$	171.0804	44.97	28.37	164	62	1.88

Table A.4: Intensity features used by the deep neural network to predict frequency of elements S,Cl,Br,B, and Se. We train three different models for isotope patterns with three, four, and five peaks. Not all features are available in all three models. p_0 is the relative intensity of the monoisotopic peak. p_1, p_2, p_3, and p_4 are the intensity of the subsequent isotope peaks. Intensities sum up to 1.

Feature	Three peaks	Four peaks	Five peaks
p_0	x	x	x
p_1	x	x	x
p_2	x	x	x
p_3		x	x
p_4			x
Minimal peak intensity	x	x	x
Maximal peak intensity	x	x	x
Median of peak intensities	x	x	x
$p_0 + p_2 + p_4$	x	x	x
$p_1 + p_3$		x	x
$\min(p_0, p_2, p_4)$	x	x	x
$\min(p_1, p_3)$		x	x
$\max(p_0, p_2, p_4)$	x	x	x
$\max(p_1, p_3)$		x	x
The following three index features are one-hot encoded:			
Most intensive peak	x	x	x
Second-most intensive peak	x	x	x
Third-most intensive peak	x	x	x
$p_0 - p_1$	x	x	x
$p_0 - p_2$	x	x	x
$p_0 - p_3$		x	x
$p_1 - p_2$	x	x	x
$p_1 - p_3$		x	x
$p_2 - p_3$)		x	x
p_0/p_1	x	x	x
p_0/p_2	x	x	x
p_0/p_3		x	x
p_1/p_2	x	x	x
p_1/p_3		x	x
p_2/p_3		x	x
$(p_0/p_1) - (p_1/p_2)$	x	x	x
$(p_1/p_2) - (p_2/p_3)$		x	x
$(p_0/p_1)/(p_1/p_2)$	x	x	x
$(p_1/p_2)/(p_2/p_3)$		x	x
$p_0 + p_1$	x	x	x
$p_0 + p_1 + p_2$	x	x	x
$p_0 + p_1 + p_2 + p_3$		x	x
$p_1 + p_2 + p_3$		x	x
$p_1 + p_2$	x	x	x
$p_2 + p_3$		x	x
$p_3 + p_4$			x

Table A.5: Mass features used by the deep neural network to predict frequency of elements S,Cl,Br,B, and Se. We train three different models for isotope patterns with three, four, and five peaks. Not all features are available in all three models. m_0 is the mass (in Da) of the monoisotopic peak. m_1, m_2, m_3, and m_4 are the masses of the subsequent isotope peaks.

Feature	Three peaks	Four peaks	Five peaks
m_0	x	x	x
$m_1 - m_0$	x	x	x
$m_2 - m_0$	x	x	x
$m_3 - m_0$		x	x
$m_4 - m_0$			x
$m_2 - m_1$	x	x	x
$m_3 - m_1$		x	x
$m_4 - m_1$			x
$m_3 - m_2$		x	x
$m_4 - m_2$			x
$m_4 - m_3$			x

Table A.6: Some kernels were named differently across publications. We list all kernels, for which the name in this thesis (left column) differs from the name in other publications (right column).

Kernel name in this thesis		ref	Kernel name in publication	
abbrev.	name		abbrev.	name
NSF	node subformula	[58]	SSC	substructure counting
JLB	joined loss binary	[32, 58]	CPJB	common path joined binary
TREEALIGN	tree alignment	[32, 58]	ALIGN	tree alignment
NPP	node probability product	[58]	FIPP	fragment intensity probability product
LPP	loss probability product	[32, 58]	LIPP	loss intensity probability product

Ehrenwörtliche Erklärung

Hiermit erkläre ich

- dass mir die Promotionsordnung der Fakultät bekannt ist,
- dass ich die Dissertation selbst angfertigt habe, keine Textabschnitte oder Ergebnisse eines dritten oder eigenen Prüfungsarbeiten ohne Kennzeichnung übernommen und alle von mir benutzten Hilfsmittel, persönliche Mitteilungen und Quellen in meiner Arbeit angegeben habe,
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe.

Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts haben mich folgende Personen unterstützt:
Sebastian Böcker, Marcus Ludwig, Tim White, Kerstin Scheubert, Marvin Meusel, Franziska Hufsky, Huibin Shen, Eric Bach und Juho Rousu.

Ich habe weder die gleiche, noch eine ähnliche oder eine andere Arbeit an einer anderen Hochschule als Dissertation eingereicht.

Jena, den 08. Juli 2018

Kai Dührkop