Connecting Secondary Metabolites and Biosynthetic Gene Clusters

Minna Oksanen

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 30.7.2021

Supervisor

Prof. Juho Rousu

Advisor

PhD Riikka Huusari

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Minna Oksanen

Title Connecting Secondary Metabolites and Biosynthetic Gene Clusters

Degree programme Master's Programme in Life Science Technologies

Major Bioinformatics and Digital Health **Code of major** SCI3092

Supervisor Prof. Juho Rousu

Advisor PhD Riikka Huusari

Date 30.7.2021 **Number of pages** 47 **Language** English

Abstract

Connecting secondary metabolites and biosynthetic gene clusters has largely been performed using rule-based approaches which require prior knowledge and chemical understanding about the metabolites and biosynthetic gene clusters. This work explores a scenario in which the two could be connected to each other using machine learning methods. Machine learning is considered as it has generalization ability to unseen data. The results show that there is some potential in machine learning methods when using a candidate set for the prediction task where a BGC/metabolite is directly predicted from a metabolite/BGC.

Keywords BGCs, natural products, secondary metabolites, kernel methods, structured output prediction

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Minna Oksanen

Työn nimi Sekundääristen metaboliittien ja biosynteettisten geeniklustereiden yhdistäminen

Koulutusohjelma Master's Programme in Life Science Technologies

Pääaine Bioinformatics and Digital Health **Pääaineen koodi** SCI3092

Työn valvoja Prof. Juho Rousu

Työn ohjaaja Dr Riikka Huusari

Päivämäärä 30.7.2021 **Sivumäärä** 47 **Kieli** Englanti

Tiivistelmä

Sekundääristen metaboliittien ja biosynteettisten geeniklustereiden yhdistämistä on enimmäkseen tehty sääntöpohjaisia lähestymistapoja käyttäen, jotka vaativat ennakkotietoa ja kemiallista ymmärrystä tutkittavista metaboliiteista ja geeniklustereista. Tämän työn tarkoitus on selvittää mahdollisuutta käyttää koneoppimista sekundääristen metaboliittien ja geeniklustereiden yhdistämiseen. Koneoppimista tarkastellaan sen yleistämiskyvyn vuoksi, sillä sitä voidaan käyttää uudenlaiseen dataan, jota ei ole aikaisemmin havaittu. Tulokset viittaavat siihen, että koneoppimisella on jonkinlaista potentiaalia, kun käytössä on kandidaattisetti metaboliitille/biosynteettiselle geeniklusterille jota yritetään ennustaa suoraan metaboliitista/biosynteettisestä geeniklusterista.

Avainsanat biosynteettiset geeniklusterit, sekundääriset metaboliitit, ydinfunktiomenetelmät

Preface

I would like to thank my advisor Dr Riikka Huusari for her insightful commentary, patience and overall awesome spirit. I want to give a big thanks to Prof. Juho Rousu for guiding through this process and sharing his knowledge, as well as the members of his research group KEPACO. Last, I appreciate very much all the suppport and encouragement I have received from my family—especially my sister, parents and brother—and friends—I thank you.

Espoo, 30.7.2021

Minna Oksanen

Contents

Symbols and abbreviations

Symbols

α	scalar	greek lowercase
a	vector, unless an index	latin lowercase
A	matrix	latin uppercase
n	number of (training) samples	

Operators

$\langle \cdot, \cdot \rangle$ inner product

Abbreviations

BGC	biosynthetic gene cluster
CCA	canonical correlation analysis
CV	cross validation
HMM	hidden Markov model
IOKR	input output kernel regression
MIBiG	Minimum information about a biosynthetic gene cluster
MLP	multilayer perceptron
MS2	tandem mass spectrometry
NP	natural product (synonymous to SM in this work)
RKHS	reproducing kernel Hilbert space
SM	secondary metabolite

1 Introduction

Natural products (NPs) are broadly defined as chemical compounds that originate in nature. In the context of this work, a natural product is considered to be a secondary metabolite (SM) that is produced by a microbe. Microbial secondary metabolism produces compounds that are not primarily needed by the organism itself but may otherwise be beneficial for the organism, examples include several antibiotics such as streptomycin and penicillin [31]. Secondary metabolites are the result of millions of years of evolution, and it is therefore not surprising that secondary metabolism is capable of producing a large variety of biochemical products. Due to this versatility, discovering new SMs has been of high interest. Discovering novel natural products is especially important for drug discovery as they are a significant source of pharmaceuticals, although they have been advantageous in many other biomedical processes as well, e.g., in agriculture and in food industry [35].

The genetic materials that code for these NPs are called biosynthetic gene clusters (BGCs). BGCs are packages of physically co-located genes within a microbial genome, and involve genes that are required for the synthesis and transportation of the SM [35].

As the availability of large-scale microbial genomic and metabolomic data is continually increasing, there is potential to develop data-driven methods for connecting BGCs and SMs. Connecting an already found SM to a BGC is perhaps the most common scenario in which linking is performed [27]. However, the reverse scenario, in which a BGC is searched for potentially interesting products, is also relevant.

Having a good understanding of how metabolites and BGCs are connected in silico would facilitate the arduous laboratory work that is required to experimentally validate links. Moreover, the manual work of screening for specific combinations of genes within a genome to retrieve a potentially interesting BGC candidate would be lightened.

Linking BGCs to their corresponding metabolites is very challenging, as the biological and chemical space is not thoroughly understood [53]. A typical BGC consists of tens to hundreds of genes that define the biosynthetic pathway, which can be complex and unpredictable from the genome alone, as the pathway often involves many post-assembly reactions that may be environment-dependent [33].

The purpose of this thesis is to experiment with different kinds of machine learning methods to connect SMs and BGCs. Machine learning involves using an algorithm to learn through experience to generalize to previously unseen data. A well-chosen machine learning algorithm can be suitable for complicated problems that cannot be sensibly programmed using a set of predefined steps. Consequently, machine learning has been applied to various tasks such as recommender systems, language processing, image recognition and medical diagnostics. However, the choice of an appropriate machine learning algorithm is far from trivial. An important reason for considering machine learning in this work is its generalization ability, i.e., unknown metabolites or BGCs in this case, which is generally not achieved to the same extent when using rule-based approaches.

The problem is viewed as:

1. Link prediction, in which two subjects (a BGC and a metabolite) are given and it should be determined whether there is a link between the two or not. A link means roughly that a BGC produces the given metabolite.

2. Ranking the most related subject. This means that for a given metabolite or BGC the most likely target is selected from a candidate set.

The primary dataset used for the experiments is gathered from MIBiG (Minimum information about a biosynthetic gene cluster), as it is to date the most comprehensive database of validated BGCs and their products. Larger datasets containing genomic and metabolomic data from different bacterial strains and corresponding metabolites perceived using tandem mass spectrometry (MS2) are also available but exploring the putative links in these datasets is not in the scope of this thesis as the number of validated links is relatively small. However, some exploration including MS2 data that could be reliably connected to a BGC is performed.

First, the biological background is discussed in Chapter 2. Then, Chapter 3 describes the datasets and features of the data used in this work. Chapter 4 covers the algorithms and Chapter 5 the performance metrics and experimental results. Chapter 6 is reserved for final discussion.

2 Biological background

In this chapter, the biological background of this thesis is covered. First, natural product discovery is discussed as it is the underlying biological motive for linking NPs and BGCs. Second, BGC discovery is discussed shortly. Third, existing approaches that have been used for linking NPs and BGCs are covered.

2.1 Natural product discovery

Natural products can be defined as secondary metabolites of living organisms, typically bacteria, fungi or plants, that are not the product of primary metabolism [58]. Primary metabolism includes energy, reproduction and anabolic pathways while secondary metabolism yields products with a higher function, such as signalling or defence [54]. The distinction between secondary and primary metabolites is thin and depends on the potential application [54]. The vast majority of known secondary metabolites are produced by plants [58] but they are often considered higher organisms than bacteria or fungi and are therefore not included in databases that consider only microbial sources such as NPAtlas [57]. Major microbial genera that produce known natural products include bacteria *Streptomyces* and *Bacillus*, and fungi *Aspergillus* and *Penicillum* [57].

Secondary metabolites are considered to be small molecules with a limited mass [58], typically less than 3000 Da [19]. They form a wide variety of chemical classes and can be divided to groups such polyketides, terpenoids, alkaloids, nonribosomal peptides (NRPs) and saccharides [35]. It is due to their large variety that has led to the successes in using them in diverse applications such as pesticides and pigments. However, the predominant applications are as antimicrobial and antiparasitic agents [58].

Natural products, or natural product derivatives, are an essential part of the pharmaceutical industry where many natural products are used as antibiotics, immunosupressants and antifungals. Out of the 1881 drugs approved by the USFDA (The United States Food and Drug Administration) in the last four decades, only half are purely synthetic or of botanical origin. The rest are natural products, have natural product pharmacophore or are semisynthetic derivatives of natural products [42]. There is also no perceivable decline in that relation, so natural products continue to be of interest when developing new drugs.

Traditionally, natural product discovery has been led by bioactivity screening, in which biological samples are gathered from environment and are extracted directly or cultivated [31]. They are then screened for desired properties using, for example, bioassays, and passed for further structural characterization. Although this is still a prolific way of finding new natural products, many natural products are difficult to extract, or do not show up in any way under laboratory conditions. Some even estimate that over 99% of microbes are not readily culturable [55], which hinders the discovery of their natural product producing potential. Many natural products are also present only in low abundances under laboratory conditions and can therefore be overshadowed by more abundant compounds. Another problem with this conventional

approach is the rediscovery of known products, the so-called low hanging fruit, which led to decreased interest in natural products by pharmaceutical companies in the 1990s. Even though there are solutions to all of the aforementioned problems, such as using a reference database for already known compounds to minimize rediscovery, they are only partial [31].

Another possibility is to regard genomic data as the starting point of natural product discovery [27]. This is also becoming more relevant, as the number of sequenced bacterial and fungal genomes has been increasing dramatically during the last two decades due to improvements in genome sequencing technologies. Currently, there are over 300000 prokaryotic genomes at least partially sequenced in NCBI's (National Center for Biotechnology Information) genome library [1]. This is attributed to the reduction in both cost and time required for sequencing because of next generation sequencing technologies. A central element of genome based natural product discovery is that after predicting a molecular product for a BGC, the connection can be validated by cloning the BGC and moving it to a heterologous host, or by mutating the genes of the original host to elicit gene expression [25].

Although the hope for genomic data to revolutionize the discovery of natural products has been high, the effect has been only relatively modest [35].

2.2 BGC detection in a genome

It has been known for many decades that the genes that code for a secondary metabolite are organized close to each other [27]. A BGC is, therefore, a collection of co-localized genes that are together responsible for the synthesis of a secondary metabolite, including the enzymes, regulatory elements, transporters and at times also resistance genes to prevent host resistance [27].

Predicting BGCs from a genome is fairly well-developed, and there are a plethora of tools for mining BGCs from genomic data, a summary can be found in Table 1. Usually, BGCs are detected using manually created and validated rules specific to each chemical class which is also the recommended way [27].

The most well-known resource for BGC mining is probably antiSMASH (antibiotics & Secondary Metabolite Analysis Shell) [7]. It finds BGCs by using queries, which are usually implemented as profile hidden Markov models (pHMMs), to identify domains or genes that are highly specific for known BGCs. PRISM 4 [52] is a similar tool to antiSMASH, and it is also capable of predicting several BGC classes from bacterial strains.

Clusterfinder [12] and DeepBGC [21] deploy a more machine learning based approach to BGC detecting with a reduced false positive rates, therefore being potentially more exploratory.

2.3 Connecting BGCs and NPs

Efficient linking of BGCs and their corresponding compounds is a key problem in natural product research [53], as verifying links between BGCs and their natural products is cost and time inefficient. Some existing approaches are discussed below.

Table 1: Tools for finding BGCs in a microbial genome.

Software	Target organism	Features
PRISM 4 [52]	Bacteria	Multiple sequence alignment based profile-HMMs for finding BGCs. Chemical tailoring reactions for predicting a molecular structure.
antiSMASH [7]	Bacteria, fungi	A rule-based approach using profile-HMMs from several databases. Able to detect 71 classes of BGCs. Incorporated into many tools such as ARTS, Pep2Path, and BIG-scape.
ClusterFinder [12]	Bacteria	An expanded version of antiSMASH algorithm. Uses HMMs that look for BGC and non-BGC areas based solely on protein family domains.
DeepBGC [21]	Bacteria	A deep learning approach using RNNs (recurrent neural networks) to identify BGCs and a random forest classifier to predict potential chemical activity.

Strategies for connecting natural products and BGCs include targeted genome mining methods in which strains are prioritized based on specific BGCs included in them. This is often a manual process that requires specific chemical knowledge [53]. An example of targeted genome mining is searching for resistance genes when looking for antibiotic compounds. For this purpose, an automated tool called ARTS has been developed [41].

Following [53], automated linking approaches can be roughly divided to feature-based and correlation-based. The assumption for them is that there is paired genomics and metabolomics data. Feature-based approaches find structural properties to detect features in mass spectra, and they have often been developed for connecting certain product types, such as Pep2Path [36] for peptidic NPs and RiPPquest [40] and DeepRiPP [37] for RiPPs. Correlation-based approaches are deployed for paired genomic and metabolomic datasets including several strains. The underlying assumption is that similar BGCs code for similar products. Therefore, molecular products, as measured by using (tandem) mass spectrometry, are therefore clustered to molecular families (MFs) while BGCs form gene cluster families (GCFs) [16]. Based on the number of shared strains of GCFs and MFs, a linking score is calculated. A method combining correlation and feature based scores is presented in [22] for paired genomics and metabolomics datasets containing MS2 spectra and BGCs where a user can get hypothetical links between the MS2 and BGC data they have. Molecular structures need to be known for the feature-based score function limiting

the applicability of that score function to only very similar BGCs whose product is known.

Phylogenomic approaches are also based on the assumption that BGCs sharing similar biosynthetic enzymes should produce similar compounds [27].

There are not many automated methods that connect an identified BGC to a metabolite without corresponding metabolomics data. Although antiSMASH, in addition to predicting BGCs, gives a rough prediction of the product for some classes of BGCs, such as PKSs, NRPs, and certain other peptides, it is not the primary goal of the tool. PRISM 4 [52] is likely the most comprehensive automated software built to predict chemical structures from BGCs acquired from bacterial genomes. It has an emphasis on antibiotic BGCs and uses a library of pre-defined chemical tailoring reactions and HMMs to achieve the structures.

As predicting the complete structure of the product based on BGC information is a very difficult problem, experiments in which the aim is to predict only some of the features of the metabolites have been conducted. For example, predicting whether the metabolite is an antibiotic [15], antifungal [15], anticancer or cytotoxic has been done [21, 52]. Additionally, predicting the structural class of the product has also been the goal [21, 52].

3 Research material

This chapter describes the datasets used in this study and some of the data's pre-processing steps. Two of these datasets are predominantly gathered from MIBiG database, described in subsection 3.1. One is larger and has no MS2 data, the other is smaller and has MS2 data associated with it. In addition to MIBiG data, a "gold standard" dataset is described in section 3.4 as connecting SMs and BGCs using MIBiG data is not common. Therefore, this dataset is taken into account to find out how a well-performing method called IOKR, described in 4.2.7, performs comparatively, the results of which are discussed in section 5.2.3.

Additional tools are used to extract features for the SMs and BGCs, as the raw MIBiG data has to be pre-processed to be used in a machine learning framework. The relationships between different data views, and whether they describe SMs or BGCs, are pictured in Figure 1. The views of the data for BGCs are discussed in subsection 3.2, and for SMs in subsection 3.3.

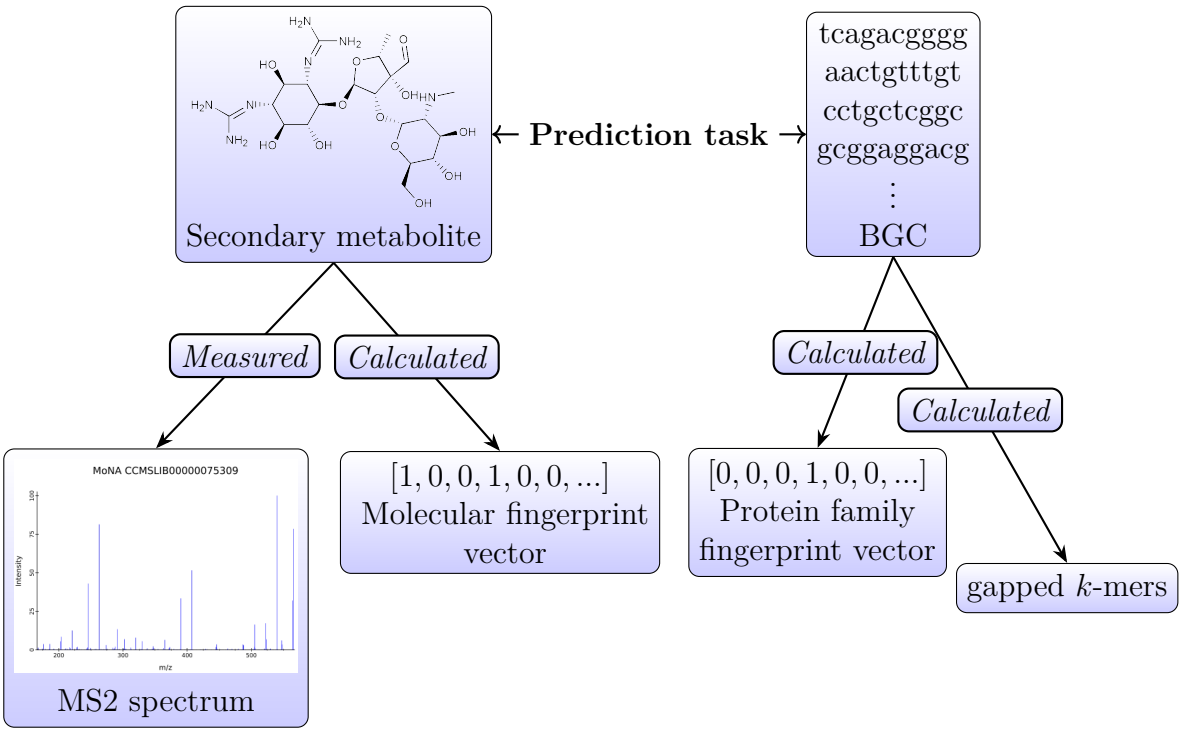


Figure 1: An illustration of the main data views and their relationships using an example metabolite Streptomycin, a widely used antibiotic. The molecular structure and arrows pointing at the data views to describe Streptomycin are pictured alongside the BGC that codes for it. The BGC is depicted as a DNA sequence and the arrows point at the data views associated with the BGC. The prediction task between SMs and BGCs is also displayed, as exploring that is the aim of this work.

3.1 MIBiG database

MIBiG (Minimal Information about a Biosynthethic Gene cluster) [26] is currently the most comprehensive database that contains, in addition to data about the BGC, information about the molecular product the BGC codes for. The database extends the principles of "minimum information" standardization framework MIxS (Minimum Information about any Sequence) in order to collect all information about a BGC into one repository instead of letting the information be scattered around in the literature. MIBiG parameters aim to describe the chemical, genomic and environmental dimensions that characterize a biosynthetic pathway. Therefore, an entry on a BGC contains information about the sequential data in MIxS form, and also reveals the biosynthetic class, chemical structure, and mass of the metabolite. Occasionally, functionality of the metabolite is added which tells, for example, whether the metabolite acts as antifungal, antibacterial or an inhibitor.

Taxonomically, the BGCs in MIBiG are mostly of bacterial (1670 BGCs) or fungal origin (249 BGCs), although there are some that come from plants (19 BGCs) or other eukaryotes. The main origin of BGCs is *Streptomyces* (636 BGCs), a bacterial genus that has been extensively researched [60]. For the purpose of this thesis, many BGCs are disregarded as there is no chemical structure associated with the BGC—the number of those BGCs is 550 according to [26]. Another criterion is for the BGC to have enough protein families, discussed in more detail in 3.2, assigned to it. In this work, it is presumed that having less than or equal to two protein families would not be descriptive enough. On the contrary, as the software used to calculate the protein families allows only 100 sequences at a time, BGCs that have too many protein families assigned to them are disregarded. A summary of the MIBiG dataset used for the experiments after filtering inappropriate BGCs is presented in Table 2. Notably, the number of unique metabolites is higher than the number of unique BGCs. Consequently, one metabolite can be linked to many BGCs, and also vice versa. This has an impact on the prediction task.

There are seven structural categories, bioclasses, for the metabolites, and one metabolite can belong to several categories. The distribution of the different bio-

Table 2: Summary of the MIBiG dataset used in this study.

Subject	Count
Number of unique BGCs	1292
Number of unique metabolites	1654
Number of BGC-metabolite links	1869
BGCs from bacteria	1048
BGCs from fungi	212
BGCs from plants	11
BGCs from other eukaryotes	13
BGCs from archaea	1
BGCs origin uncharacterized	7

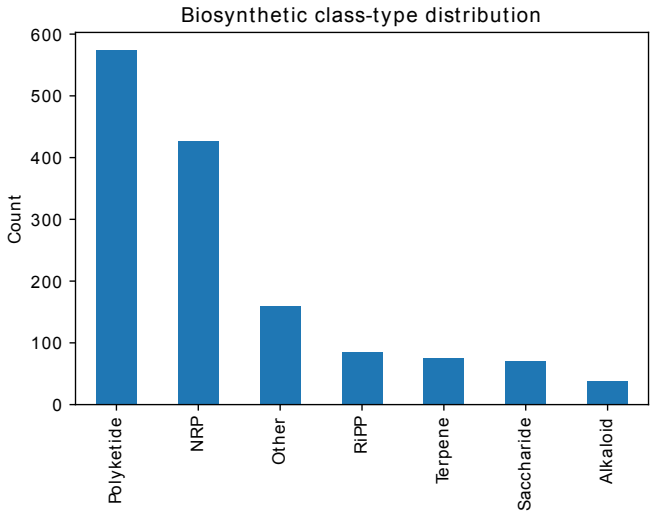


Figure 2: Distribution of the bioclasses in the MIBiG dataset used in this thesis.

classes is shown in Figure 2. Polyketides and nonribosomal peptides (NRP) are the most abundant biosynthetic classes found in MIBiG. This is perhaps due to polyketide synthases and nonribosomal peptide synthases being enzymes that have been extensively searched for due to their involvement in the biosynthesis of therapeutic agents [59]. It has been suggested that saccharides are in fact the most abundant SMs found in prokaryotes [12], so the MIBiG distribution is not necessarily a truthful representation of the true distribution of SMs found in prokaryotes.

3.2 BGC representation

As illustrated in Figure 1, two representations for BGCs are considered: firstly, one-hot encoding them into a binary vector and secondly gapped k-mers. Kernels can be calculated for both of these representations. One-hot encoding means that a feature is either ”on”, i.e., present, in the BGC and the corresponding value is 1. Respectively, when a feature is not present it is coded as a 0. Protein families (pfams) are selected as features since they are commonly used as the features to predict BGCs from a genome [21, 12].

Protein families are found by searching the coding amino acid sequences (FASTA sequences) of a BGC as documented in its MIBiG entry against a protein family database Pfam-A (with expectation level cut-off value at 0.0001) using PfamScan [32].

Similarly, gapped k-mers can be calculated using the coding amino acid sequences. A k-mer is an aminoacid sequence encoded as a string of length k. These are then used to calculate a string kernel between BGCs. The method does not have as strong biological foundation as the pfam approach since the kernel is essentially formed by merely measuring sequence similarities. A more detailed description on how the string kernel is calculated is covered in subsection 4.2.6.

3.3 Molecular representation

Similarly to BGCs, the metabolites can be represented by one-hot encoding them into a binary vector, the result of which is generally called a fingerprint vector. Each bit now describes the presence or absence of a chemical structure or feature, and the vectors can be used for similarity searching, clustering and classification.

The fingerprints selected to describe the metabolites include Klekota Roth fingerprints of length 4860 [29]. They consist of substructure indicators that have been found to be prevalent in commercially available drugs. In addition, substructure of length 307 and the standard fingerprinter of length 1024, defined in the Chemistry Development Kit [61], were added to the fingerprint vector. Usually, substructure fingerprints perform well on small molecules, and secondary metabolites are small molecules although larger than, e.g., completely synthetic drugs [4].

A popular choice in cheminformatics is to use the ECFP-6 (Extended Connectivity Fingerprint) [47], and this is also considered for experiments concerning "gold standard" dataset. ECFP was designed specifically for structure-activity screening, and it has been shown to outperform other fingerprints at target prediction. It finds specific circular substructures around each atom in a molecule, which are predictive of the biological activities of small molecules [10]. The molecular fingerprints are computed using R's implementation of CDK [20].

MS2 data is included in a part of the experiments due to the importance of tandem mass spectrometry in small molecule research [18]. It is the main high-throughput technique for measuring small molecules in a biological sample. Mass spectrometry measures the mass-to-charge ratio and intensity of a compound, and when it is coupled with another mass analyser the result is a tandem mass spectrum. Now, the molecular fragments also have mass-to-charge rations and intensities, which makes it possible to distinguish different isomers. A kernel matrix can be calculated for MS2 spectra, which is also how they are included into the prediction framework.

A prevalent difficulty with including MS2 data is that MS2 data are not currently linked to their BGCs in one easily accessible repository, although this is likely to be fixed in the near future. An already existing database called Paired Omics Data Platform [49] is perhaps the most comprehensive repository containing links between MS2 spectra and BGCs even though the number of confirmed links is at the time just over 100.

More MS2-BGC links than are present in Paired Omics Data Platform can be found by searching different databases for connections. The dataset used in this work with MS2 data is from [22], and it involves MS2 data from GNPS (The Global Natural Product Social Molecular Networking) and MIBiG data for which a matching spectrum could be found. A match was made according to the first part of InChIKeys, which were calculated for both BGCs' molecular products and the molecules MS2 spectra are measured for. The first part of the InChIKeys does not take into account stereoisomerism which is coded in the second part of the key. Using this method for matching, a spectrum could be found for 248 BGCs.

3.4 Gold standard dataset

Currently, there are no published results that exploit only MIBiG data for predicting links between BGCs and metabolites. For the purpose of comparing how well an algorithm used in this work performs in predicting metabolites from BGCs, an additional dataset is considered.

The authors of PRISM 4 [52] manually curated a dataset of BGCs and their corresponding metabolites, and named the resulting dataset ”gold standard” dataset. The dataset consists of 1281 prokaryotic BGCs from different public databases, including MIBiG, ClusterMine360, DoBISCUIT and NRPS-PKS, and also from their own in-house database. The gold standard dataset was gathered to validate the structural as well as functional predictions of the molecular structures computed by PRISM 4. Additionally, the accuracies of the predicted structures of PRISM 4 were compared to those of antiSMASH and NP.searcher.

The dataset differs from the MIBiG dataset in that there are only prokaryotic BGCs whereas MIBiG contains also eukaryotic BGCs. This is somewhat significant as prokaryotic and eukaryotic secondary metabolic pathways are probably quite different, at least prokaryotes generally exhibit more diverse primary metabolism [43]. The dataset is also presumably even more curated than that of MIBiG as the authors reported that there are considerable errors in the public databases they examined, including MIBiG. Hence, it may contain less errors than other datasets.

4 Methods for connecting NPs and BGCs

In this chapter, the machine learning algorithms used in the experiments are explained. First, a brief description of machine learning is provided and the problem is formulated in two ways, as link prediction problem between BGCs and metabolites, and as a learning problem in which BGCs/metabolites are directly predicted from metabolites/BGCs using a candidate set. Then, the algorithms used in the experiments of this thesis are explained. They include kernel based methods, correlation based methods and a multilayer perceptron which is a deep learning method. An overview of the methods is given in Figure 3.

4.1 Preliminaries

Given data $(x_1, y_1), ..., (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, $\mathcal{X}$ denotes the non-empty set of inputs where x_i come from, $\mathcal{Y}$ is the set of outputs where y_i belong, and n is the number of samples. Assuming that the unseen test data come from the same distribution as the training data, learning means the algorithm's ability to generalize to unseen data, that is, given a new input $x \in \mathcal{X}$ the aim is to predict output $y \in \mathcal{Y}$. The goal is hence to a learn a function $f : \mathcal{X} \to \mathcal{Y}$. Methods like this that have labelled data as outputs are called supervised learning algorithms. Supervised learning can be further considered as either classification or regression. In classification, the output belongs to a discrete category. Binary classification includes two classes for the output, often denoted as $\{-1, 1\}$. Regression, on the contrary, has continuous values as its outputs.

Connecting metabolites and their BGCs is considered as a link prediction problem

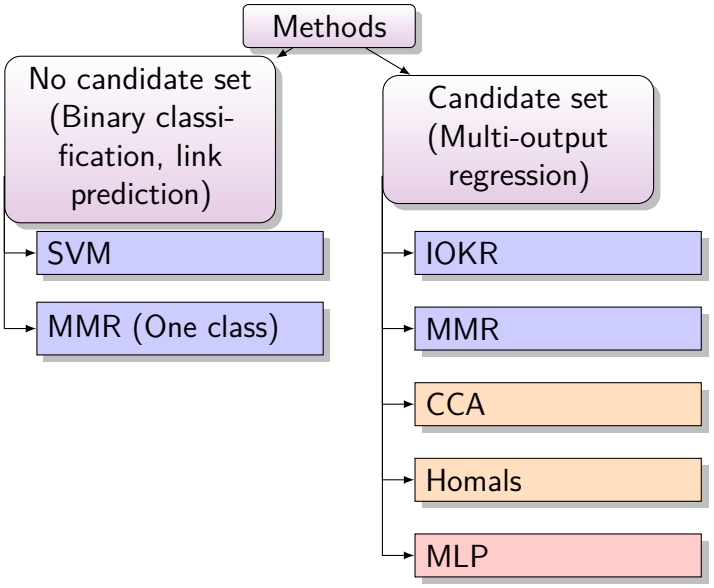


Figure 3: The considered methods are divided to two classes, link prediction (without a candidate set) and multi-output regression (with a candidate set). Kernel based methods are in blue, correlation-based in orange and deep learning in red.

and a prediction problem where the best-matching candidate for a BGC/metabolite from a candidate set defines a connection. Here, both BGCs and metabolites can be viewed as an input x and an output y. Link prediction can be thought as binary-classification, in which two elements are given, and a link is or is not determined between them, more formally $f_{lp}\colon \mathcal{X}_{Met} \times \mathcal{X}_{BGC} \to \{-1, 1\}$. Intrinsically, it does not learn one-to-one connections which should be beneficial as there are many connections in the data that are one-to-many and many-to-one. In contrast, multi-output regression tries to determine the features of one element based on the other. In this scenario, the most likely output is selected from a candidate set containing representations of the output category, and it can be expressed as $f_{mor}\colon \mathcal{X}_{Met} \to \mathcal{X}_{BGC}$, or in the opposite direction as $f_{mor}\colon \mathcal{X}_{BGC} \to \mathcal{X}_{Met}$.

4.2 Kernel methods

Kernel methods are one of the most popular machine learning techniques. There are good reasons for this, for example, the theoretical foundation of the algorithms is well-understood, kernel methods are widely applicable to various data types, and nonlinear models are easily achieved. Additionally, one is able to learn regularized smooth functions in RKHS (reproducing kernel Hilbert space). Kernel methods, unlike many popular deep learning based methods, are also a reasonable choice in applications where the number of samples is relatively small and the data may be very high-dimensional.

A central part of kernel methods is an attractive feature, the so-called *kernel trick*, which essentially means that by using a positive definite kernel function, it is possible to use linear learning algorithms to predict non-linear relations in data. This is beneficial, as linear learning algorithms are well-developed, but the dependencies in the data are often non-linear. The kernel gives an inner product in a (usually) high dimensional feature space using the input features, but does not directly calculate the feature expansion. This reduces the amount of operations needed to classify the data in a higher dimensional space.

In the following, some of the basic concepts of kernel methods are covered first, basing the discussion mostly on [48, 51]. Then, the kernel methods based algorithms used in the experiments are explained.

4.2.1 Definition of a kernel and properties

A kernel is a symmetric positive semi-definite function $\kappa\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. For all inputs $x, x' \in \mathcal{X}$

$$\kappa(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}, \tag{1}$$

where $\Phi\colon x \mapsto \kappa(x, \cdot)$ is called a feature map, which maps inputs x and x' into RKHS, also called the feature space $\mathcal{H}$. RKHS is a Hilbert space of functions that has a reproducing kernel, and Moore-Aronszajn theorem [3] states that every positive definite kernel defines a unique RKHS. When learning with kernels, the hypothesis $f_{\mathcal{H}}\colon \mathcal{X} \to \mathbb{R}$ lives in $\mathcal{H}$. RKHS is a Hilbert space that has reproducing property

$$f(x) = \langle f, \kappa(x, \cdot) \rangle, \quad \forall f \in \mathcal{H}, \tag{2}$$

giving the name to the reproducing kernel Hilbert space. In particular, when f is replaced with $\kappa(x', \cdot)$ in (2), we get

$$\kappa(x, x') = \langle \kappa(x, \cdot), \kappa(x', \cdot) \rangle = \langle \Phi(x), \Phi(x') \rangle, \tag{3}$$

which is the *kernel trick*.

When trying to learn or define a function that fits the data, minimizing empirical loss, i.e., the loss on training data, will often lead to overfitting and likely poor generalization to unseen data. This can be fixed by regularization to restrict the class of possible minimizers, and a popular regularized functional is

$$\hat{f}_{\mathcal{H}} = \arg\min_{f} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2, \tag{4}$$

where $\lambda > 0$ is a regularizing weight, $\|f\|^2$ the squared norm in RKHS, and L is an arbitrary loss function which is often chosen to be convex. Different learning algorithms can be achieved by placing different loss functions L on (4), such as a support vector machine or kernel ridge regression.

Representer theorem states that the solution to (4) is

$$\hat{f}_{\mathcal{H}}(x) = \sum_{i=1}^{n} \alpha_i \kappa(x_i, x), \quad \alpha_i \in \mathbb{R}. \tag{5}$$

Representer theorem is significant as it states that a possibly infinite dimensional optimization problem, as $\mathcal{H}$ can be infinite dimensional, can be formulated to be finite dimensional, and the solution is only dependent on the given kernel and training samples.

With a kernel κ and inputs $x_1, ..., x_n \in \mathcal{X}$, the inner products between pairs of data samples mapped to a feature space give an $n \times n$ matrix K, which is either called the Gram matrix or the kernel matrix $K := (\kappa(x_i, x_j))_{ij}$:

$$K := \begin{bmatrix} \kappa(x_1, x_1) & \dots & \kappa(x_1, x_n) \\ \vdots & \ddots & \vdots \\ \kappa(x_1, x_n) & \dots & \kappa(x_n, x_n) \end{bmatrix}.$$

All the information given to a kernel algorithm is retrieved from this matrix and it is often an information bottleneck as calculating the matrix becomes heavy with a high number of samples, especially when using naive approaches. Several approaches exist to overcome this, most famous of which is probably the Nyström approximation [34].

4.2.2 Support Vector Classification

Support vector machine (SVM) is an algorithm for solving (4) with a specific loss function. The ideas for SVM were made famous by Cortes and Vapnik [14] in the 1990's. SVM and its numerous variants continue to be of high relevance especially in classification tasks for small to medium sized datasets. Extensions of SVM for both regression and multilabel classification exist, but here the focus is on binary classification $y \in \{-1, 1\}$.

The classical soft-margin support vector realization of (4) is retrieved by placing a hinge loss function on the predicted outputs, where hinge loss is defined as

$$L_{\text{hinge}}(y, f(x)) = \max(0, 1 - yf(x)). \tag{6}$$

The objective (4) now becomes

$$\hat{f}_{\mathcal{H}} = \arg\min_{f} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2. \tag{7}$$

SVMs were first described using a perspective of convex optimization theory. A hyperplane

$$f(x) = \langle w, \Phi(x) \rangle + b \tag{8}$$

separates the two classes, and the distance (margin) between the classes is maximized. Here b is a scalar. Maximizing the margin between the two classes equals to minimizing $\frac{1}{2} w^\top w$, corresponding to the regularization term $\lambda \|f\|^2$ in (7). The classifier is then $\text{sign}(f(x))$. It is formulated as a constrained optimization problem so that the constraints $y_i(\langle w, \Phi(x_i) \rangle + b) \geq 1 - \xi_i$ are drawn from the fact that $f(x) = \langle w, \Phi(x_i) \rangle + b$ is the decision boundary, and $y \in \{-1, 1\}$. This corresponds to the first part of the equation (7), namely that $L_{hinge} = 0$ when the constraint $y_i(\langle w, \Phi(x_i) \rangle + b) \geq 1$ is true. The constrained optimization problem is now:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + \eta \sum_{i}^{n} \xi_i \\ \text{subject to} \quad & y_i(\langle w, \Phi(x_i) \rangle + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, .., n. \end{aligned} \tag{9}$$

There is a trade-off in (9) between the amount of misclassifications controlled with regularization parameter η and minimizing the regularization term $\frac{1}{2} w^T w$. Slack variables ξ_i make the model soft-margin instead of hard-margin. Hard-margin means that each training point must be classified correctly by the minimizer. In practice, this is not possible for nonseparable data, and slack variables are introduced to relax the constraints. Soft-margin also prevents overfitting, as it allows for misclassifications, whereas in hard-margin SVM a single outlier can greatly influence the decision boundary. Hinge loss L_{hinge} can be seen to define the magnitude of the slack variable, $\xi_i = 1 - y_i f(x_i)$. Therefore, $\xi_i > 1$ for misclassified samples, and $0 < \xi_i < 1$ for samples that are close to the decision boundary, and $\xi_i = 0$ for correctly classified samples.

The formulation in (9) is also known as the primal form, and it can be expressed in so-called dual form using Lagrange multipliers α_i. The dual form is

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \kappa(x_i, x_j) \alpha_i \alpha_j + \sum_{j=1}^{n} \alpha_j \\ \text{subject to} \quad & \sum_{i}^{n} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq \eta, i = 1, ..., n, \end{aligned} \tag{10}$$

which can be solved using quadratic programming. Further, the decision function can be derived to be

$$f(x) = \text{sign}\left(\sum_{i \in SV} y_i \alpha_i \langle \Phi(x), \Phi(x_i) \rangle + b\right) = \text{sign}\left(\sum_{i \in SV} y_i \alpha_i \kappa(x, x_i) + b\right), \quad (11)$$

where SV denotes the set of support vectors. Support vectors are those samples x_i whose Lagrange multipliers α_i are positive. Only those samples whose Lagrange multipliers are positive define the margin.

4.2.3 Support Vector Classification in the experiments

To adapt to a link prediction framework, a Kronecker product kernel $K \in \mathbb{R}^{n_m n_b \times n_m n_b}$, where n_m is the number of metabolites in data and n_b the number of BGCs, has to be constructed. A Kronecker product kernel is commonly used for pairwise datasets. The matrices for which a Kronecker product kernel is defined are kernel matrices for metabolites $A \in \mathbb{R}^{n_m \times n_m}$ and BGCs $B \in \mathbb{R}^{n_b \times n_b}$. A Kronecker product kernel is defined for the two matrices A and B so that it forms a block-matrix where each element in A is multiplied by matrix B. This is defined as

$$A \otimes B = K \quad (12)$$

$$K := \begin{bmatrix} \kappa(a_1, a_1)B & \dots & \kappa(a_1, a_n)B \\ \vdots & \ddots & \vdots \\ \kappa(a_n, a_1)B & \dots & \kappa(a_n, a_n)B \end{bmatrix}.$$

The prediction function for a test pair (a, b) can be expressed as

$$f(a, b) = \sum_{i=1}^{n} \alpha_i \kappa((a_i, b_i), (a, b)) = \alpha^\top k, \quad (13)$$

where k is a column vector containing kernel values between each training pair a_i and b_i and test pair (a, b), and α is learnt by minimizing an objective function.

As the full-sized Kronecker product kernel between BGCs and metabolites would be in the size range of $\mathbb{R}^{10^6 \times 10^6}$, it becomes very slow to calculate and a straightforward implementation would not be suitable anymore. Using a so-called generalized vec trick introduced in [2] it is possible to reduce the amount of computations for the Kronecker kernel based methods by not explicitly calculating the large kernel. The loss function choice for the SVM implementation proposed in [2] is slightly different to the previously mentioned hinge loss, as it is the squared hinge loss defined as

$$L_2(y, f(x)) = \frac{1}{2} \sum_{i=1}^{n} \max(0, 1 - f(x_i) y_i)^2. \quad (14)$$

This loss function is chosen instead of hinge loss because it gives an objective function that is differentiable and suitable for the proposed optimization algorithm.

4.2.4 Maximum margin regression

Maximum margin regression (MMR) [56] extends the ideas of binary SVM to structured output learning, i.e., when the output is not a scalar but arbitrary, for example a vector. The inputs are now mapped to feature space as $\phi\colon \mathcal{X} \mapsto \mathcal{H}_\phi$ and outputs to label space as $\psi\colon \mathcal{Y} \mapsto \mathcal{H}_\psi$. The goal is to learn a function f that acts on the feature space, and returns a prediction for the output in the label space:

$$\begin{aligned} f(\phi(x)) &= W\phi(x) + b, \\ y &= \psi^{-1}(f(\phi(x))). \end{aligned} \tag{15}$$

This is achieved by interpreting the normal vector of the separating hyperplane w as a linear operator $W\colon \mathcal{H}_\phi \to \mathcal{H}_\psi$ which projects the feature space into the label space.

The dual of (15) has form

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j\langle\phi(x_i),\phi(x_j)\rangle\langle\psi(y_i),\psi(y_j)\rangle + \sum_{i=1}^{n}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i(\psi(y_i))_t, \quad t = 1,...,\dim(\mathcal{H}_\psi) \\ & 0 \le a_i \le \eta, \quad i = 1,...,n, \end{aligned} \tag{16}$$

which resembles closely the SVM dual form (10).

4.2.5 Maximum margin regression in the experiments

MMR is used in the experiments as a way to define a metabolite/BGC directly from BGC/metabolite and as a link prediction method. In link prediction, given that the dataset is very unbalanced in the number of positive and negative links, a one-class classification framework is the focus. For predicting a corresponding metabolite/BGC for a metabolite/BGC, a candidate set of $\mathcal{Y}^*$ is considered including all of the candidates in the dataset. The bias term b is dropped.

First, in link prediction the prediction is made according to

$$y = \text{sign}(\sum_{i=1}^{n}\alpha_i\langle\psi(y),\psi(y_i)\rangle\langle\phi(x_i),\phi(x)\rangle). \tag{17}$$

For structured output prediction a candidate set is introduced and the best-matching candidate is found by:

$$\begin{aligned} y &= \arg\max_{y\in\mathcal{Y}^*}\psi(y)^\top W\phi(x) \\ y &= \arg\max_{y\in\mathcal{Y}^*}\sum_{i=1}^{n}\alpha_i\langle\psi(y),\psi(y_i)\rangle\langle\phi(x_i),\phi(x)\rangle. \end{aligned} \tag{18}$$

4.2.6 Kernel functions

In the following, the kernel functions used in the experiments are described. When considering multiple views for data, e.g., gapped k-mers and protein families, multiple kernel learning is practiced so that each kernel has a uniform weight, and the result is a sum of these kernels.

Tanimoto kernel is used as it proved to be the most efficient kernel in most of the learning scenarios. It can be defined for binary one-hot-encoded vectors x and y as

$$\kappa_{\text{Tanimoto}}(x, y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{19}$$

String kernels for BGCs are calculated using a program called FastSK, as described in [6]. A regular k-mer string kernel calculates similarities between sequences x and y based on length k subsequences, i.e., k-mers. It has a feature space that is enormous when the sequence lengths for x and y are large, and by introducing gaps it is possible to reduce the size of the feature space.

The formula for calculating the gapped string kernel κ_{GSK} for sequences (here amino acid sequences) x and y is

$$\kappa_{\text{GSK}}(x, y) = \sum_{\gamma \in \theta_{g,m}} c_x(\gamma) c_y(\gamma) \tag{20}$$

where $\theta_{g,m}$ is the set of gapped k-mers, g is overall feature length where of length k informative positions are considered, m is the number of mismatch positions in the g-mers. Particularly, $g = k + m$. Function $c_x(\gamma)$ gives the count of gapped k-mers γ in x, and $c_y(\gamma)$ respectively the count of gapped k-mers in y.

For example, with a sequence "AGGCT" and $k = 2$, $m = 1$ and $g = 3$ the considered features are "AGG", "GGC", and "GCT". Since $m = 1$ and $k = 2$ the possible gapped $k-$mers are "_GG", "_GC", "_CT", when m is positioned in the first place, "A_G", "G_C", "G_T" when m is in the second place and "_GG", "_GC" and "_CT" when m is in the third place.

The complete algorithm for κ_{FSK} is

$$\kappa_{\text{FSK}}(x, y) = \sum_{i}^{\binom{g}{m}} \sum_{\gamma \in \theta_i} c_x(\gamma) c_y(\gamma), \tag{21}$$

where θ_i is the set of gapped k-mers of ith combination of m mismatch positions. The kernel κ_{FSK} is then a sum of the product of counts of the same gapped k-mers of x and y. The difference to equation (20) is that the kernel is decomposed into a set of independent counting operations which allows for parallelizing the algorithm thus making the algorithm faster.

All of the coding amino acid sequences in the BGCs are used as sequences to calculate the string kernel, and afterwards the kernel values are averaged according to the number of aminoacid sequences in each BGC to result in a kernel matrix of size $\mathbb{R}^{n_b \times n_b}$, where n_b is the total number of BGCs. The steps are depicted in Figure 4. As this is a modification of the original string kernel given by κ_{FSK}, further examination should be made to make sure it is a valid kernel.

Indicating the presence of aminoacids in a BGC_i can be decoded into a binary vector d^{is}, which is 1 if aminoacid a_s is in BGC_i and otherwise 0. The value of the kernel matrix at position ij is formulated as

$$\begin{aligned} K_{ij} &= \frac{1}{\#\text{available}} \sum_{s=1}^{m} \sum_{t=1}^{m} \kappa((BGC_i, a_s), (BGC_j, a_t)) \\ &= \frac{1}{\#\text{available}} \sum_{s=1}^{m} \sum_{t=1}^{m} \langle d^{is}, d^{jt} \rangle \kappa_{FSK}(a_s, a_t), \end{aligned} \tag{22}$$

where m denotes the total number of aminoacid sequences. In (22) the first part after summation is a linear kernel of the binary features multiplied by the aminoacid kernel. A kernel multiplied by a kernel is also a kernel. The $\frac{1}{\#\text{available}}$ is the averaging part, and should be a kernel also. To make sure it is a kernel, it should be symmetric and positive semi-definite. It is symmetric. It is also positive semi-definite, as

$$\begin{aligned} \#\text{available} &= \sum_{s=1}^{m} \sum_{t=1}^{m} d^{is} d^{jt\top} = \langle d^i, d^i \rangle \langle d^j, d^j \rangle \\ \frac{1}{\#\text{available}} &= \langle d^i, d^i \rangle^{-1} \langle d^j, d^j \rangle^{-1} \end{aligned} \tag{23}$$

which is one element of the matrix and a product of two scalars. The full kernel matrix can be written as $BB^\top$ where

$$B = \begin{bmatrix} \langle d^1, d^1 \rangle^{-1} \\ \vdots \\ \langle d^{n_b}, d^{n_b} \rangle^{-1} \end{bmatrix},$$

and the matrix is positive semi-definite.

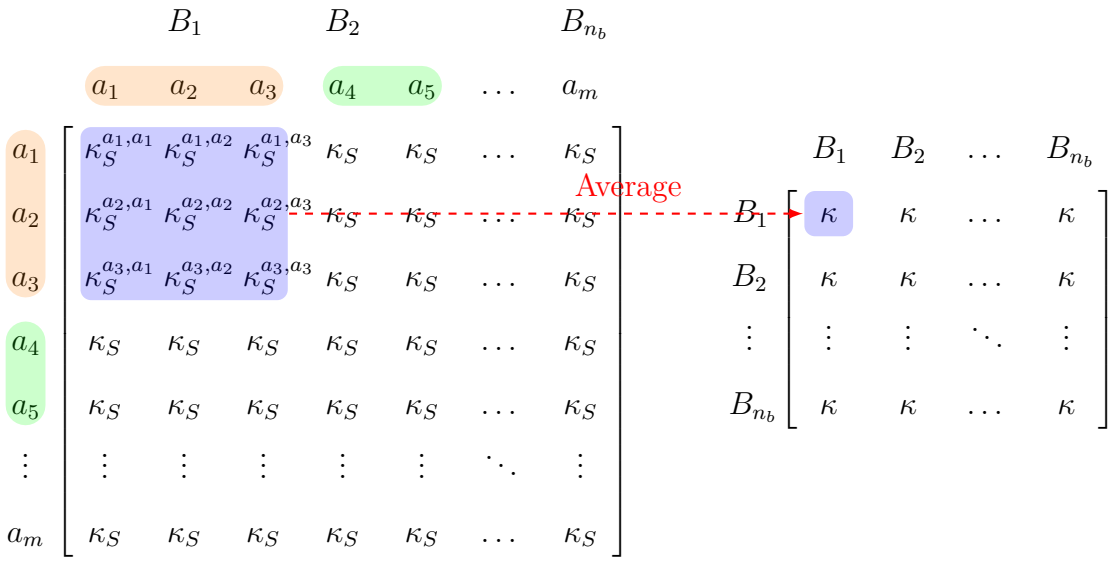


Figure 4: Processing after achieving the FastSK values κ_S for amino acid sequences a_i. B_i represents a single BGC. The number of amino acid sequencies vary BGC to BGC. On the right is the final string kernel used in the experiments.

4.2.7 Input Output Kernel Regression

Perhaps the most common output value of supervised learning is a scalar. Multi-output learning (MOL) is a branch of machine learning that deals with output spaces that are vector valued. This is the scenario when trying to directly predict a metabolite/BGC from each other. Input output kernel regression was introduced in [9] as a novel generalization of the kernel dependency estimation [13]. The method was shown to be both fast and accurate in [8] for identifying metabolites from MS2 spectra, and [8] is also mostly followed in this section.

In short, input output kernel regression learns mappings between structured inputs and structured outputs with operator-valued kernels (OVKs), which have an output that is a bounded linear operator, i.e., $\mathcal{K}\colon \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{H}_y, \mathcal{H}_y)$. OVKs generalize the scalar-valued kernels to operator-valued kernels, and they satisfy symmetry and positive semi-definiteness conditions much like scalar-valued kernels. IOKR is based on the definition of an output scalar-valued kernel κ_y and an input operator-valued kernel $\mathcal{K}_x$.

IOKR consists of two steps: the first one is output kernel regression, in which function h is learnt between the input space $\mathcal{X}$ and the feature space $\mathcal{H}_y$, which is the RKHS associated with κ_y. The second step is to learn or, in this case more so to define, a function g from $\mathcal{H}_y$ to the output set $\mathcal{Y}$. This is called the pre-image problem (an inverse-problem). The motivation is that learning $\mathcal{X} \to \mathcal{Y}$ is difficult as $\mathcal{Y}$ can be anything, but $\mathcal{X} \mapsto \phi(y)$ can be done using OVKs.

An illustration of IOKR workflow is shown in Figure 5.

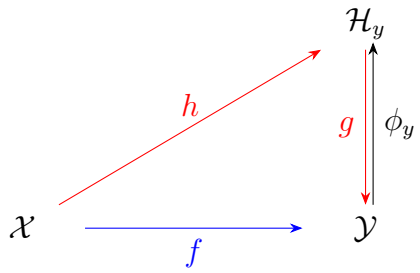


Figure 5: Diagram of the IOKR phases. Function f is defined through space $\mathcal{H}_y$ instead of directly learning it.

4.2.7.1 Output Kernel Regression

Output kernel regression benefits from RKHS theory devoted to vector-valued functions introduced in [38, 50]. The function h's values are elements in Hilbert space $\mathcal{H}_y$, $h\colon \mathcal{X} \to \mathcal{H}_y$ and are searched in RKHS with reproducing kernel $\mathcal{K}_x$ as the mapping $\mathcal{X} \to \mathcal{H}_y$ can be done only for each individual element in $\mathcal{H}_y$ when using earlier introduced scalar-valued kernels. Therefore, the reproducing kernel is now operator-valued, instead of scalar-valued. Assuming that the dimension d of $\mathcal{H}_y$ is finite, the operator-valued kernel $\mathcal{K}_x$ is a matrix of size $nd \times nd$ with $d \times d$ sized blocks as values, and n training samples.

The models for h are of the form

$$h(x) = \sum_i \mathcal{K}_x(x, x_i)c_i, \quad c_i \in \mathcal{H}_y, x_i \in \mathcal{X}. \tag{24}$$

The function h is found by minimizing a regularized optimization problem with a least squares loss function

$$\arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} ||h(x_i) - \phi_y(y_i)||^2_{\mathcal{H}_y} + \lambda||h||^2_{\mathcal{H}}. \tag{25}$$

By placing (24) in (25) and computing the derivative [38] showed that vectors c_i verify for finite $\mathcal{H}_y$

$$\text{vec}(c_n) = (\lambda I_{nd} + G_{\mathcal{X}_n})^{-1}\text{vec}(\Phi_{\mathcal{Y}_n}), \tag{26}$$

where $c_n = (c_1, ..., c_n)$, $\Phi_{\mathcal{Y}_n} = (\phi_y(y_1), ..., \phi_y(y_n))$, I_{nd} = an identity matrix and $G_{\mathcal{X}_n}$ is the kernel matrix given by the operator-valued kernel on the training set.

4.2.7.2 Pre-image step

When the output kernel is normalized, the closest y is found in the candidate set $\mathcal{Y}^*$ by calculating

$$\hat{f}(x) = \arg\max_{y\in\mathcal{Y}^*} \langle \hat{h}(x), \Phi_y(y)\rangle_{\mathcal{H}_y}. \tag{27}$$

Given that the operator-valued kernel $\mathcal{K}_x$ is assumed to be of form $\kappa(x, x')I_d$, the closest output y in the candidate set can be found by

$$\hat{f}(x) = \arg\max_{y \in \mathcal{Y}_*} (k_{Y_n})^T (\lambda I_n + K_{X_n})^{-1} k_{X_n}, \tag{28}$$

where K_{X_n} is the scalar valued kernel on the inputs, and k_{Y_n} and k_{X_n} correspond to the column vectors

$$k_{Y_n} = \begin{bmatrix} \kappa_y(y_1, y) \\ \vdots \\ \kappa_y(y_n, y) \end{bmatrix}, k_{X_n} = \begin{bmatrix} \kappa_x(x_1, x) \\ \vdots \\ \kappa_x(x_n, x) \end{bmatrix}.$$

Notably, equation (28) returns a score value (a scalar) for each of the candidates $y \in \mathcal{Y}^*$. A candidate set of output structures is therefore a necessity for IOKR to function as it is.

4.3 Correlation based algorithms

The following two algorithms, canonical correlation analysis (CCA) and homogeneity analysis (homals), are closely related to each other in that they aim to maximize the correlation between variables in multivarate datasets, thus extracting common features between the datasets.

Unsupervised learning which focuses on finding patterns in the data, often aiming at identifying similar groups within the data, or for visualization purposes, is in addition to supervised learning a major orientation in machine learning. Especially canonical correlation analysis (CCA) is best-known for being considered as an unsupervised learning method. However, it can be used in a regression task as well.

CCA is regularly applied with only two datasets, and therefore used that way also here, although there are ways to extend it to more than two data views. Homals is extended in this work to three data views. CCA and homals are also applicable considering the data are high-dimensional, and the methods can be seen as ways of reducing the dimensionality of the data.

4.3.1 Canonical correlation analysis

First described by Hotelling in 1936 [24], canonical correlation analysis (CCA) draws from paired sets of data and finding the maximum correlation between linear combinations of the data. It can also be understood as a way of estimating weights that are used to predict one set from the other. CCA can be therefore viewed as an extension of multivariate regression analysis without the assumption of causal asymmetry.

Let data be $X_t \in \mathbb{R}^{n \times d_t}, t = 1, ..., N$. That is, the data are real values represented with n samples and d_t variables, and there are N data views. We seek to find linear transformation matrices $W_t \in \mathbb{R}^{d_t \times m}$ that maximize the correlation between the paired datasets X_t, thus finding new variables that explain the variability within and

between the datasets. A common way of presenting the objective for two data views is an inner product form:

$$\begin{aligned} &\max \quad \langle X_2 W_2, X_1 W_1 \rangle \\ &\text{w.r.t } W_t \in \mathbb{R}^{d_t \times m} \\ &\text{such that } \langle X_t W_t, X_t W_t \rangle = I_m, \ t = 1, 2. \end{aligned} \tag{29}$$

4.3.2 CCA in the experiments

After learning the matrices W_t that fulfil the previous conditions, predictions can be made for known X_1 through

$$X_{(2,pred)} = X_1 W_c + \epsilon, \tag{30}$$

with coefficients $W_c = W_1 L_2^\top$ where L_2 are so-called loadings that are closely related to W_2 but with different normalization [46].

The data are represented using kernel matrices, and the features of a metabolite/BGC are kernel function values, which are also sometimes called landmark representation features. Pearson correlation coefficients are calculated between predictions and each element in a candidate set and the highest correlation value yielding candidate is chosen as the prediction.

4.3.3 Homals algorithm

Homals algorithm is built around the concept of homogeneity analysis, also called multiple correspondence analysis, and it is presented for example in [39] as a part of so-called Gifi system. The goal in homogeneity analysis is to minimize departure from homogeneity, which is measured with a loss function (31). As in CCA, correlations between variables in multivariate datasets are maximized. Homals algorithm places different restrictions on the solution than CCA, aiming for optimal scaling. Another motivation for homogeneity analysis has been that it can produce a low-dimensional representation of the data and, therefore, be used as a graphic method.

Homals can be used in a strict and broad sense, strict requiring categorical data while the broad sense does not. It is used in a broad sense here, as the data is represented using kernels.

Gifi introduced the loss function in least squares form and euclidean distances instead of correlation and inner product form. The objective is now:

$$\begin{aligned} &\min \tfrac{1}{N} \textstyle\sum_{t=1}^{N} \|Z - X_t W_t\|^2 \\ &\text{w.r.t } W_t \in \mathbb{R}^{d_t \times m} \\ &\text{such that } \langle Z, Z \rangle = m I_m, \quad 1^T Z = 0, t = 1, ..., N. \end{aligned} \tag{31}$$

In (31) $Z = \mathbb{R}^{n \times m}$ is a common factor matrix, and m represents the dimension of the common factor space. $W_t = \mathbb{R}^{d_t \times m}$ represents the weight components in the item space. The minimization problem (31) can be solved by using, for example, the alternating least squares (ALS) algorithm, as the suffix -als in the name homals suggests.

4.3.4 Homals in the experiments

After obtaining the optimal common factors $Z \in \mathbb{R}^{n \times m}$ and weights $W_t \in \mathbb{R}^{d_t \times m}$ the prediction step can be performed in many ways. For predictions from view 1 (known) to view 2 (unknown), the closest example is found in a candidate set. This is presented for two views here for simplicity, but it is easy to generalize. Define common factor $\tilde{z}$ as $W_1^T x^{(1)}$. The upper index means that the x vector belongs to that view. Find the closest $x_i^{(2)}$ in the candidate set by

$$\tilde{x}^{(2)} = \arg\min_i \|\tilde{z} - W_2^T x_i^{(2)}\|^2. \tag{32}$$

As with CCA, the data are represented using kernel matrices only instead of one-hot encoded vectors, and the best-matching candidate is found by Pearson correlation coefficients.

4.4 Multilayer perceptrons

Multilayer perceptrons (MLPs), also known as feed-forward neural networks, are one of the most simple kinds of deep learning algorithms. Neural networks are based on (often many) layers that are connected to each other and hierarchically extract features from the data, and it is loosely inspired by the parallel architecture found in brains. Even though many of the concepts of deep learning were presented already in the 20th century, such as perceptrons in the 1950s and multilayer perceptrons in the 1980s, it gained significant popularity when larger datasets ($n \gtrsim 5000$) became more available in the 21st century coupled with increased computational resources. Currently, deep learning is an efficient approach in many machine learning tasks, especially in image classification, speech recognition and natural language processing. This section follows largely [17].

MLPs are universal function approximators and can therefore approximate any function f when nonlinear activations functions are applied. More specifically, they are also universal approximators of vector-valued functions [23] when the width wd of the network is large enough. Adequate widths have been proposed for certain function classes, e.g. for $L^P(\mathbb{R}^{dx}, \mathbb{R}^{dy})$ it is $wd_{\min} = \max\{d_x + 1, d_y\}$ where d_x is the input dimension and d_y is the output dimension [44]. An MLP is therefore appropriate for the task of predicting BGCs/metabolites from each other considered in this work.

An MLP involves at least an input layer, one hidden layer, and an output layer of nodes, and it is fully connected. Fully connected means that in an MLP every node is connected with a weight w_{ij} to all the nodes in the following layer. The value of each node can be formulated for given input x as

$$f(x) = A(Wx + b) \tag{33}$$

where W defines the weights, b is the bias and A is the chosen activation function. $f(x)$ is then passed for the next layer where it is multiplied it with a specific weight w_{ij} for each node in the next layer, unless $f(x)$ is an output layer value.

The network can be mathematically formulated as a chain of functions: for three layers where $f^{(1)}$ is the first layer of the network, $f^{(2)}$ the second layer, $f^{(3)}$ the third layer, the result is a network

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))). \tag{34}$$

An activation function has to be chosen for the network, and it can vary layer by layer. The purpose of activation functions is to introduce non-linearity to the model, as without them the model would output just a linear combination of the inputs. Some common activation functions are sigmoid, tanh, and rectified linear unit (ReLU). ReLU is currently the recommended standard activation function, defined as

$$A_{\mathrm{ReLU}}(z) = \max(0, z), \tag{35}$$

where z is the input to the node, corresponding to $Wx + b$ in equation (33).

Backpropagation is the underlying method that describes how the parameters w and b in an MLP are learnt through gradients. The goal is again to miminize a loss function L (for example, squared mean error) with respect to parameters w, b. Backpropagation does this by adjusting weights and biases in the network by using the chain rule of calculus to compute the derivatives of chained functions. Suppose x is a real number, functions $f, g \in \mathbb{R} \to \mathbb{R}$, and $y = g(x), z = f(g(x)) = f(y)$. The chain rule now states that

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}. \tag{36}$$

Backpropagation calculates the chain rule in an efficient way, starting from the last layer of nodes and progressing towards the first layer. Forward propagation gives the value of loss function L, and with backpropagation the weights w and biases b are updated according to

$$\begin{aligned} w &\leftarrow w - \lambda\frac{dL}{dw} \\ b &\leftarrow b - \lambda\frac{dL}{db}, \end{aligned} \tag{37}$$

where λ is a predefined learning rate parameter.

Although the goal of optimizing the parameters is to find the global minimum of a loss function, often it is not found. This is a result of the loss function of MLPs being nonconvex. It used to be thought that potentially finding a local minimum is a major drawback of deep learning and MLPs. In practice the optimizing algorithm is not trapped in local minima but saddle points which have to be escaped [45], and local minima are found to be often good enough [11].

Stochastic gradient descent (SGD) and its variants are often used as the optimization algorithms in deep learning networks. SGD uses a sub-sample, or the vanilla SGD only one sample, of the training data to calculate the updated gradients.

Adam (Adaptive Moment Estimation) [28] is an extension of SGD that does not have a single learning rate. Adam is used as the optimizing algorithm in the MLP experiments of this thesis due to its fast convergence and established status as the

default training algorithm in deep learning, although its generalization performance is not as good as that of SGD [62].

Neural networks are sometimes said to be black-boxes partly due to the huge amount of parameters (often millions) that are learnt, making the model very hard to interpret. It is quite difficult to explain the amount of contribution of each variable in the model without using some tool that is built for that purpose, such as SHAP [30]. In addition, defining an appropriate network architecture is trial and error, and an architecture is often quite task-specific.

4.4.1 MLP in the experiments

The network used in the experiments is shown in Figure 6. Through experimenting with different network depths and layer widths, in the end only one hidden layer of size 500 is chosen. Like in CCA and homals, the inputs and outputs are rows of kernel matrices. It is also completely feasible to use the one-hot encoded feature vectors of BGCs and metabolites as inputs and outputs, and this was also considered but the results were poor.

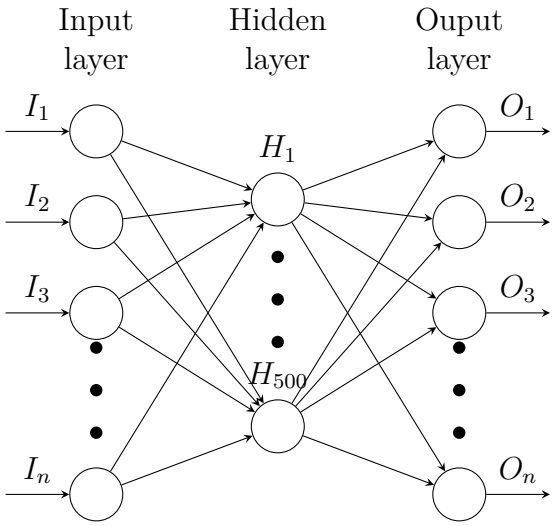


Figure 6: Illustration of the MLP used in the experiments, n denotes the number of training samples.

5 Experiments

In this chapter, the cross-validation setup and the metrics used to measure performance of the algorithms are explained first. The achieved results are then discussed in the context of the datasets that were used.

5.1 Cross-validation and metrics

In the following, the cross-validation scenarios and the performance metrics used in this work are explained.

5.1.1 Cross-validation

Cross-validation is used as it is meant to test the model's ability to generalize to unseen data, and therefore guide in the determination of which model is the best. In the experiments, 10-fold cross-validation is performed so that the data is randomly partitioned into 10 sections, constructing 10 different data splits for testing and training. Each time one of these 10 sections will act as a test set while the rest are for training. No surveillance is made in making sure that different biosynthetic classes are equally represented in each fold.

10-fold cross-validation is primarily performed in this work so that no BGC or metabolite in the training set is also in the test set. That is, nothing about the connections of a BGC/metabolite is known in the test set. This makes the prediction task more difficult compared to a scenario where the test and train sets have BGCs/metabolites in common. This approach is sometimes called stratified cross-validation which is not the classic method of CV where data is merely randomly split. The reason for preferring stratified cross-validation is that it is the most realistic scenario a prediction is made in, as the space of potential BGCs and SMs is enormous and it is not likely that there is prior knowledge about their connections.

Conversely, higher prediction accuracies are achieved if that condition is relaxed, which is also shown for the best performing methods. The relaxation is such that the target metabolite/BGC for a BGC/metabolite can be already in the training set but not vice versa.

5.1.2 Performance metrics

The metrics used to test the performance of each algorithm depend on the learning scenario. For link prediction, the chosen performance metrics are accuracy, recall (sensitivity), precision and F1-score. For predicting the BGC/metabolite directly from each other, top-k performance and Tanimoto coefficient (Jaccard index) are suitable choices.

Denoting "number of" as #, "True" meaning that the prediction is correct, and "False" meaning that the prediction is not correct. Assuming that the classes are represented with $\{-1, 1\}$, "positive" means that the predicted value of the element is positive, and "negative" means that the predicted value is negative. The metrics for link prediction are listed and defined in Table 3.

Table 3: Performance metrics for link-prediction.

Metric	Definition
Accuracy	$\frac{\#\text{True predictions}}{\#\text{All predictions}}$
Recall	$\frac{\#\text{True positives}}{\#\text{True positives} + \#\text{False negatives}}$
Precision	$\frac{\#\text{True positives}}{\#\text{True positives} + \#\text{False positives}}$
F1-score	$0.5 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

Table 4: Performance metrics for prediction with a candidate set.

Metric	Definition
Top-k performance	The correct output (BGC/metabolite connected to a metabolite/BGC) is ranked in top-k of candidates
Tanimoto coefficient between a and b	$\frac{N_c}{N_a+N_b-N_c}$, N_c = #common bits on in a and b N_a = #bits on in a N_b = #bits on in b

In link prediction, accuracy is perhaps not as descriptive as the other metrics because of the imbalanced class distribution in the data. This gives the chance for any classifier that always predicts negative to yield a decent accuracy. Therefore, precision and recall are more relevant metrics. Precision measures the rate of true positives against all samples that are labeled as positive. Recall measures the rate of true positives against all actual positive samples. In other words, precision shows the proportion of true positives, and recall shows the proportion of captured positives. F1-score is the harmonic mean of recall and precision, and combines the two in one metric.

The candidate set prediction metrics are presented in Table 4. Now, top-k performance is a reasonable choice, and top-1, top-5, top-10 and top-20 results are considered. Tanimoto coefficient is suitable for calculating the similarity between two one-hot encoded vectors. It is especially suitable for comparing how similar two molecules are [5], which is also how it is used in this thesis.

Standard deviation is also calculated between each fold.

5.2 Results

The results for the experiments are sectioned according to the data that are used. MIBiG dataset without MS2 data is examined most thoroughly. First, it is used in the link prediction setting and then in the prediction task for finding a BGC/metabolite for a metabolite/BGC with a candidateset. Furthermore, the effects of different CV settings and views of data are shown.

Then, a smaller MIBiG dataset with MS2 data is considered. The effect of having MS2 data is viewed for three methods in the prediction task of predicting BGCs from metabolites.

Finally, "gold standard" dataset is considered to find out to what extent IOKR compares to PRISM 4.

The best results are shown in the tables with gray highlighting, if applicable, and ± denotes one times standard deviation.

5.2.1 MIBiG, no MS2 data

Starting with link prediction, the results for SVM and MMR using stratified cross-validation are shown in Table 5. They are not good, which is to be expected with such difficult and unbalanced data. Interestingly, MMR learns the positive links well (good recall) but at the cost of many false positives (low precision). SVM has good accuracy but it still fails to classify almost all positive links.

The results for predicting BGCs from metabolites are shown in Tables 6–8. With stratified CV, MLP is the most successful method, followed by IOKR and homals. MLP predicts particularly well the correct BGC (top-1) compared to the other methods. When the cross-validation setting is made easier both MLP and IOKR benefit from the easier learning scenario, as shown in Table 7. The improvements are significant for both of the methods but especially for IOKR. Using gapped k-mers, g-kmers, together with protein family fingerprint vectors as features appears beneficial for at least homals and MLP, as is shown in Table 8. This is rather unexpected since having only g-kmers as features for BGCs does not yield even nearly as good results as using only protein families.

The results for predicting metabolites from BGCs are shown in Tables 9–11. Overall, they are not as good as in the reverse direction. With stratified CV, IOKR is the most effective method. When the CV setting is relaxed, both IOKR and MLP again improve their prediction accuracy, as is shown in Table 10. The effect of views of data for BGCs is not evident in this prediction direction, as can be seen from Table 11.

The fairly significant standard errors between folds are partly explained by the fact that different biosynthetic classes are predicted with different success rates, and the classes are unevenly distributed among folds. The distribution of badly predicted metabolites from BGCs using IOKR for different bioclasses is shown in Figure 7.

Table 5: Results for link prediction.

Method	accuracy	recall	precision	f1
SVM	1.00±0.00	0.04±0.01	0.02±0.01	0.02±0.01
MMR (one-class)	0.52±0.00	0.96±0.02	1.5-e4±0.00	1.5-e4±0.00

Table 6: Predicting BGCs from metabolites, stratified CV.

Method	top-1	top-5	top-10	top-20
IOKR	0.01±0.01	0.34±0.05	0.45±0.06	0.54±0.05
MMR	0.02±0.01	0.09±0.02	0.14±0.03	0.18±0.04
CCA	0.01±0.01	0.03±0.03	0.05±0.02	0.08±0.02
Homals	0.06±0.03	0.22±0.06	0.24±0.06	0.27±0.06
MLP	0.13±0.03	0.36±0.05	0.49±0.06	0.59±0.03

Table 7: Predicting BGCs from metabolites, relaxed CV setting.

Method	top-1	top-5	top-10	top-20
IOKR	0.35±0.02	0.57±0.03	0.63±0.03	0.71±0.03
MLP	0.29±0.03	0.55±0.03	0.66±0.02	0.73±0.02

Table 8: Predicting BGCs from metabolites, the effect of data views on BGCs.

Data views for BGCs	Method	top-1	top-5	top-10	top-20
g-kmer + protein family	Homals	0.06±0.03	0.22±0.06	0.24±0.06	0.27±0.06
protein family	Homals	0.04±0.02	0.16±0.04	0.19±0.04	0.20±0.05
g-kmer + protein family	IOKR	0.00±0.01	0.28±0.04	0.42±0.05	0.54±0.04
protein family	IOKR	0.01±0.01	0.34±0.05	0.44±0.06	0.54±0.05
g-kmer + protein family	MLP	0.13±0.03	0.36±0.05	0.49±0.06	0.59±0.03
protein family	MLP	0.07±0.03	0.26±0.03	0.37±0.04	0.47±0.04

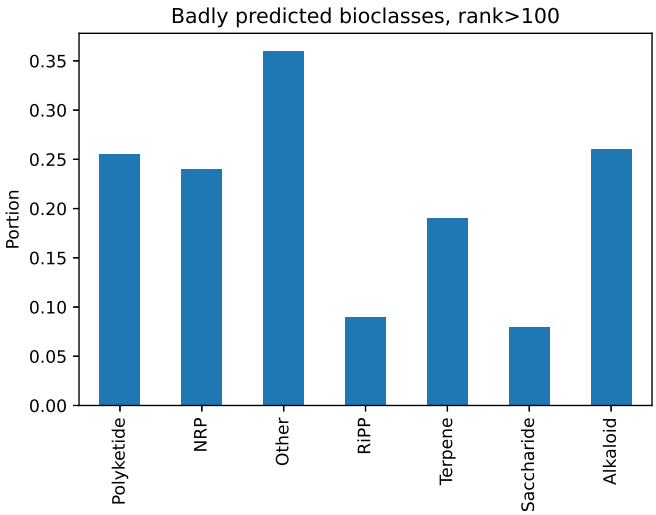


Figure 7: Using IOKR to predict metabolites from BGCs. The portions of different bioclasses in badly predicted (rank>100) compounds are shown.

Table 9: Predicting metabolites from BGCs, stratified CV.

Method	top-1	top-5	top-10	top-20
IOKR	0.01±0.01	0.24±0.04	0.35±0.04	0.47±0.04
MMR	0.01±0.01	0.09±0.02	0.14±0.03	0.22±0.05
CCA	0.01±0.01	0.03±0.02	0.05±0.02	0.09±0.03
Homals	0.01±0.00	0.05±0.05	0.06±0.02	0.09±0.03
MLP	0.06±0.02	0.20±0.04	0.29±0.06	0.38±0.06

Table 10: Predicting metabolites from BGCs, relaxed CV setting.

Method	top-1	top-5	top-10	top-20
IOKR	0.11±0.01	0.33±0.04	0.44±0.04	0.54±0.05
MLP	0.09±0.02	0.28±0.04	0.39±0.04	0.47±0.04

Table 11: Predicting metabolites from BGCs, the effect of data views on BGCs.

Data views for BGCs	Method	top-1	top-5	top-10	top-20
g-kmer + protein family	Homals	0.00±0.00	0.04±0.02	0.06±0.04	0.08±0.05
protein family	Homals	0.00±0.00	0.04±0.02	0.06±0.04	0.08±0.05
g-kmer + protein family	IOKR	0.01±0.01	0.23±0.05	0.34±0.05	0.45±0.04
protein family	IOKR	0.01±0.01	0.24±0.04	0.35±0.04	0.47±0.04
g-kmer + protein family	MLP	0.06±0.02	0.20±0.04	0.29±0.06	0.38±0.06
protein family	MLP	0.05±0.01	0.19±0.03	0.29±0.04	0.38±0.05

Table 12: Predicting BGCs from metabolites with MS2 data, the effect of data views on metabolites.

Data views for metabolites	Method	top-1	top-5	top-10	top-20
MS2+molecular fingerprints	Homals	0.04±0.02	0.27±0.20	0.32±0.19	0.38±0.17
molecular fingerprints	Homals	0.04±0.07	0.29±0.20	0.34±0.18	0.43±0.02
MS2+molecular fingerprints	IOKR	0.01±0.02	0.20±0.08	0.34±0.18	0.43±0.18
molecular fingerprints	IOKR	0.01±0.01	0.30±0.05	0.40±0.07	0.58±0.07
MS2+molecular fingerprints	MLP	0.02±0.01	0.05±0.02	0.09±0.04	0.14±0.04
molecular fingerprints	MLP	0.08±0.07	0.45±0.13	0.59±0.12	0.72±0.11

5.2.2 MIBiG, with MS2 data

The dataset [22] comprising two views for metabolites, i.e., MS2 data and molecular fingerprints and their corresponding BGCs is considered separately. The two views for metabolites provide with an opportunity to examine how MS2 data could be integrated into the prediction of BGCs from metabolites, given that the connections between MS2 data and the molecular fingerprints are known. MS2 data does not seem

to be complementary to molecular fingerprint data when predicting the corresponding BGC of a metabolite, as it does not enhance the top-k performance. It is worth noting that the top-k scores even with this smaller dataset are comparable to those presented earlier which were achieved by using more data. The results are shown in Table 12.

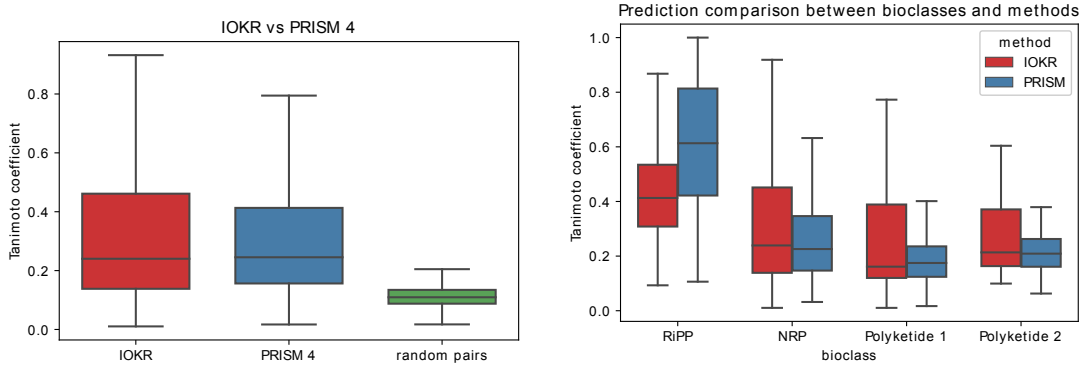
5.2.3 Gold standard dataset

The gold standard dataset [52] is used to make comparisons between IOKR and PRISM 4 metabolite predictions for the BGCs. The learning scenario is quite different between the two, IOKR uses a candidate set to score the most likely output, and PRISM4 uses its rule-based approach for determining the most likely chemical structure. ECFP6 fingerprints are used to represent metabolites.

Tanimoto coefficients are calculated between the highest ranked metabolite, predicted by either IOKR or PRISM, and the true metabolite. Tanimoto coefficients are also calculated between random metabolites in the dataset, denoted as ”random pairs”. A high tanimoto coefficient implies that the predicted structure/fingerprint vector is close to the correct structure. The results are shown in Figures 8(a) and 8(b). The boxes in the figures show interquartile ranges with medians drawn as horizontal lines, and whiskers at most more/less than 1.5 times the interquartile range.

It can be seen from Figure 8(a) that the medians of the two methods, IOKR and PRISM, are very close to each other when considering all of the bioclasses. Figure 8(b) shows that the medians are close for NRPs and polyketides whereas RiPPs are predicted better by PRISM.

Notably, even though the number of possible molecular candidates is increased to include all of the secondary metabolites in the Natural Product Atlas ($n = 29007$) [57], the Tanimoto coefficient distribution does not change for the worse. This suggests that assuming ECFP6 and Tanimoto coefficient are valid indicators of chemical properties, IOKR performs rather well at predicting relevant chemical properties



(a) All of the bioclasses are plotted in one box plot.

(b) The most abundant bioclasses are plotted.

Figure 8: IOKR vs PRISM with two splits of the data.

from BGCs.

6 Discussion

The aim of this work was to explore the suitability of machine learning algorithms for the task of predicting links/connections between secondary metabolites and BGCs. The results suggest that there is some potential in machine learning to aid in the connection of metabolites and BGCs in the proposed way of predicting metabolites, as expressed by molecular fingerprints, and BGCs directly from each other using a candidate set. Especially IOKR and MLP were fairly successful in this, even with a smaller dataset of 248 BGCs. Similar results have been achieved by PRISM 4 and correlation methods although the approaches are very different to the ones proposed here.

Having previous knowledge about which connections already exist between a metabolite or a BGC greatly enhanced the prediction scores. This is to be expected, as all of the existing approaches for the task rely heavily on previous knowledge of already known compounds or biosynthetic pathways. This is a fault that is recognized also by [53] and the authors of PRISM 4, and leads to bias towards the study of variants of those compounds which naturally reduces the chances of discovering novel NPs.

Furthermore, it was shown that different views for BGCs can improve the prediction results although there is probably still plenty of room for improvements both in data quality and feature representations—even though the choice of algorithm had a greater impact on the results in this work than feature representations. By data quality primarily MIBiG data are meant as the quality of the entries had variation.

Overall, some machine learning algorithms are up to an extent applicable even to datasets of this level of difficulty, and potentially may one day gain more popularity in actual applications regarding natural product discovery/connecting SMs to their BGCs.

References

[1] National library of medicine (us) national center for biotechnology information. https://www.ncbi.nlm.nih.gov/genome/browse/#!/prokaryotes/. Accessed: 2021-07-14.

[2] Antti Airola and Tapio Pahikkala. Fast kronecker product kernel methods via generalized vec trick. *IEEE transactions on neural networks and learning systems*, 29(8):3374–3387, 2017.

[3] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[4] Richard Bade, Ho-Fung Chan, and Jóhannes Reynisson. Characteristics of known drug space. natural products, their derivatives and synthetic drugs. *European journal of medicinal chemistry*, 45(12):5646–5652, 2010.

[5] Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):1–13, 2015.

[6] Derrick Blakely, Eamon Collins, Ritambhara Singh, Andrew Norton, Jack Lanchantin, and Yanjun Qi. Fastsk: fast sequence analysis with gapped string kernels. *Bioinformatics*, 36(Supplement 2):i857–i865, 12 2020.

[7] Kai Blin, Simon Shaw, Alexander M Kloosterman, Zach Charlop-Powers, Gilles P van Wezel, Marnix H Medema, and Tilmann Weber. antismash 6.0: improving cluster detection and comparison capabilities. *Nucleic acids research*, page 1, 2021.

[8] Céline Brouard, Huibin Shen, Kai Dührkop, Florence d'Alché Buc, Sebastian Böcker, and Juho Rousu. Fast metabolite identification with input output kernel regression. *Bioinformatics*, 32(12):i28–i36, 2016.

[9] Céline Brouard, Marie Szafranski, and Florence d'Alché Buc. Input output kernel regression: Supervised and semi-supervised structured output prediction with operator-valued kernels. *Journal of Machine Learning Research*, 17(176):1–48, 2016.

[10] Alice Capecchi, Daniel Probst, and Jean-Louis Reymond. One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome. *Journal of Cheminformatics*, 12(1):1–15, 2020.

[11] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. 2015.

[12] Peter Cimermancic, Marnix H Medema, Jan Claesen, Kenji Kurita, Laura C Wieland Brown, Konstantinos Mavrommatis, Amrita Pati, Paul A Godfrey,

Michael Koehrsen, Jon Clardy, et al. Insights into secondary metabolism from a global analysis of prokaryotic biosynthetic gene clusters. *Cell*, 158(2):412–421, 2014.

[13] Corinna Cortes, Mehryar Mohri, and Jason Weston. A general regression technique for learning transductions. In *Proceedings of the 22nd international conference on Machine learning*, pages 153–160, 2005.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] Alexander Crits-Christoph, Nicholas Bhattacharya, Matthew R. Olm, Yun S. Song, and Jillian F. Banfield. Transporter genes in biosynthetic gene clusters predict metabolite characteristics and siderophore activity. *bioRxiv*, 2020.

[16] James R Doroghazi, Jessica C Albright, Anthony W Goering, Kou-San Ju, Robert R Haines, Konstantin A Tchalukov, David P Labeda, Neil L Kelleher, and William W Metcalf. A roadmap for natural product discovery based on large-scale genomics and metabolomics. *Nature chemical biology*, 10(11):963–968, 2014.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[18] GA Nagana Gowda and Danijel Djukovic. Overview of mass spectrometry-based metabolomics: opportunities and challenges. *Mass Spectrometry in Metabolomics*, pages 3–12, 2014.

[19] Shaily Goyal, Kishan Gopal Ramawat, and Jean-Michel Mérillon. Different shades of fungal metabolites: an overview. *Fungal metabolites*, 1:29, 2016.

[20] Rajarshi Guha and Miguel Rojas Cherto. rcdk: Integrating the cdk with r, 2017.

[21] Geoffrey Hannigan, David Prihoda, Andrej Palicka, Jindrich Soukup, Ondrej Klempir, Lena Rampula, Jindrich Durcak, Michael Wurst, Jakub Kotowski, Dan Chang, Rurun Wang, Grazia Piizzi, Gergely Temesi, Daria Hazuda, Christopher Woelk, and Danny Bitton. A deep learning genome-mining strategy for biosynthetic gene cluster prediction. *Nucleic acids research*, 47, 08 2019.

[22] Grímur Hjörleifsson Eldjárn, Andrew Ramsay, Justin JJ Van Der Hooft, Katherine R Duncan, Sylvia Soldatou, Juho Rousu, Rónán Daly, Joe Wandy, and Simon Rogers. Ranking microbial metabolomic and genomic links in the nplinker framework using complementary scoring functions. *PLoS computational biology*, 17(5):e1008920, 2021.

[23] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[24] Harold Hotelling. Relations between two sets of variables. *Biometrika*, 28(3-4):321–377, 12 1936.

[25] Liujie Huo, Joachim J Hug, Chengzhang Fu, Xiaoying Bian, Youming Zhang, and Rolf Müller. Heterologous expression of bacterial natural product biosynthetic pathways. *Natural product reports*, 36(10):1412–1436, 2019.

[26] Satria A Kautsar, Kai Blin, Simon Shaw, Jorge C Navarro-Muñoz, Barbara R Terlouw, Justin J J van der Hooft, Jeffrey A van Santen, Vittorio Tracanna, Hernando G Suarez Duran, Victòria Pascal Andreu, Nelly Selem-Mojica, Mohammad Alanjary, Serina L Robinson, George Lund, Samuel C Epstein, Ashley C Sisto, Louise K Charkoudian, Jérôme Collemare, Roger G Linington, Tilmann Weber, and Marnix H Medema. MIBiG 2.0: a repository for biosynthetic gene clusters of known function. *Nucleic Acids Research*, 48(D1):D454–D458, 10 2019.

[27] Emma Kenshole, Marion Herisse, Michael Michael, and Sacha J Pidot. Natural product discovery through microbial genome mining. *Current Opinion in Chemical Biology*, 60:47–54, 2021.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Justin Klekota and Frederick P Roth. Chemical substructures that enrich for biological activity. *Bioinformatics*, 24:2518–2525, 11 2008.

[30] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[31] Yunzi Luo, Ryan E Cobb, and Huimin Zhao. Recent advances in natural product discovery. *Current opinion in biotechnology*, 30:230–237, 2014.

[32] Fábio Madeira, Young Mi Park, Joon Lee, Nicola Buso, Tamer Gur, Nandana Madhusoodanan, Prasad Basutkar, Adrian RN Tivey, Simon C Potter, Robert D Finn, et al. The embl-ebi search and sequence analysis tools apis in 2019. *Nucleic acids research*, 47(W1):W636–W641, 2019.

[33] Loïc Martinet, Aymeric Naômé, Benoit Deflandre, Marta Maciejewska, Déborah Tellatin, Elodie Tenconi, Nicolas Smargiasso, Edwin De Pauw, Gilles P van Wezel, and Sébastien Rigali. A single biosynthetic gene cluster is responsible for the production of bagremycin antibiotics and ferroverdin iron chelators. *Mbio*, 10(4):e01230–19, 2019.

[34] Giacomo Meanti, Luigi Carratino, Lorenzo Rosasco, and Alessandro Rudi. Kernel methods through the roof: handling billions of points efficiently. *arXiv preprint arXiv:2006.10350*, 2020.

[35] Marnix H Medema and Michael A Fischbach. Computational approaches to natural product discovery. *Nature chemical biology*, 11(9):639–648, 2015.

[36] Marnix H Medema, Yared Paalvast, Don D Nguyen, Alexey Melnik, Pieter C Dorrestein, Eriko Takano, and Rainer Breitling. Pep2path: automated mass spectrometry-guided genome mining of peptidic natural products. *PLoS computational biology*, 10(9):e1003822, 2014.

[37] Nishanth J. Merwin, Walaa K. Mousa, Chris A. Dejong, Michael A. Skinnider, Michael J. Cannon, Haoxin Li, Keshav Dial, Mathusan Gunabalasingam, Chad Johnston, and Nathan A. Magarvey. Deepripp integrates multiomics data to automate discovery of novel ribosomally synthesized natural products. *Proceedings of the National Academy of Sciences*, 117(1):371–380, 2020.

[38] Charles A Micchelli and Massimiliano Pontil. On learning vector-valued functions. *Neural computation*, 17(1):177–204, 2005.

[39] George Michailidis and Jan De Leeuw. The gifi system of descriptive multivariate analysis. *Statistical Science*, pages 307–336, 1998.

[40] Hosein Mohimani, Roland D Kersten, Wei-Ting Liu, Mingxun Wang, Samuel O Purvine, Si Wu, Heather M Brewer, Ljiljana Pasa-Tolic, Nuno Bandeira, Bradley S Moore, et al. Automated genome mining of ribosomal peptide natural products. *ACS chemical biology*, 9(7):1545–1551, 2014.

[41] Mehmet Direnç Mungan, Mohammad Alanjary, Kai Blin, Tilmann Weber, Marnix H Medema, and Nadine Ziemert. Arts 2.0: feature updates and expansion of the antibiotic resistant target seeker for comparative genome mining. *Nucleic acids research*, 48(W1):W546–W552, 2020.

[42] David J. Newman and Gordon M Cragg. Natural products as sources of new drugs over the nearly four decades from 01/1981 to 09/2019. *Journal of natural products*, 83(3):770–803, 2020.

[43] A Oren. Metabolic diversity in prokaryotes and eukaryotes. *Biological science fundamentals and systematics-Volume II*, page 40, 2009.

[44] Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. *arXiv preprint arXiv:2006.08859*, 2020.

[45] Razvan Pascanu, Yann N Dauphin, Surya Ganguli, and Yoshua Bengio. On the saddle point problem for non-convex optimization. *arXiv preprint arXiv:1405.4604*, 2014.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[47] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[48] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[49] Michelle A Schorn, Stefan Verhoeven, Lars Ridder, Florian Huber, Deepa D Acharya, Alexander A Aksenov, Gajender Aleti, Jamshid Amiri Moghaddam, Allegra T Aron, Saefuddin Aziz, et al. A community resource for paired genomic and metabolomic data mining. *Nature chemical biology*, 17(4):363–368, 2021.

[50] E Senkene and Arkady Tempel'man. Hilbert spaces of operator-valued functions. *Mathematical transactions of the Academy of Sciences of the Lithuanian SSR*, 13(4):665–670, 1973.

[51] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[52] Michael A. Skinnider, Chad W. Johnston, Mathusan Gunabalasingam, and *et al*. Comprehensive prediction of secondary metabolite structure and biological activity from microbial genome sequences. *Nature Communications*, 11(6058), 2020.

[53] Sylvia Soldatou, Grimur Hjorleifsson Eldjarn, Alejandro Huerta-Uribe, Simon Rogers, and Katherine R Duncan. Linking biosynthetic and chemical space to accelerate microbial secondary metabolite discovery. *FEMS Microbiology Letters*, 366(13), 06 2019.

[54] Maria Sorokina and Christoph Steinbeck. Review on natural products databases: where to find data in 2020. *Journal of cheminformatics*, 12(1):1–51, 2020.

[55] Wolfgang R Streit and Ruth A Schmitz. Metagenomics–the key to the uncultured microbes. *Current opinion in microbiology*, 7(5):492–498, 2004.

[56] Sandor Szedmak, John Shawe-Taylor, et al. Learning via linear operators: Maximum margin regression. In *In Proceedings of 2001 IEEE International Conference on Data Mining*. Citeseer, 2005.

[57] Jeffrey A Van Santen, Grégoire Jacob, Amrit Leen Singh, Victor Aniebok, Marcy J Balunas, Derek Bunsko, Fausto Carnevale Neto, Laia Castaño-Espriu, Chen Chang, Trevor N Clark, et al. The natural products atlas: an open access knowledge base for microbial natural products discovery. *ACS central science*, 5(11):1824–1833, 2019.

[58] Ramasamy Vijayakumar and Suresh SS Raja. *Secondary Metabolites: Sources and Applications*. BoD–Books on Demand, 2018.

[59] Christopher T Walsh and Michael A Fischbach. Natural products version 2.0: connecting genes to molecules. *Journal of the American Chemical Society*, 132(8):2469–2493, 2010.

[60] AC Ward and N EE Allenby. Genome mining for the search and discovery of bioactive compounds: the streptomyces paradigm. *FEMS microbiology letters*, 365(24):fny240, 2018.

[61] Egon L Willighagen, John W Mayfield, Jonathan Alvarsson, Arvid Berg, Lars Carlsson, Nina Jeliazkova, Stefan Kuhn, Tomáš Pluskal, Miquel Rojas-Chertó, Ola Spjuth, et al. The chemistry development kit (cdk) v2. 0: atom typing, depiction, molecular formulas, and substructure searching. *Journal of cheminformatics*, 9(1):1–19, 2017.

[62] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven HOI, et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *arXiv preprint arXiv:2010.05627*, 2020.