University of Nottingham
UK | CHINA | MALAYSIA

UNIVERSITY OF NOTTINGHAM

DOCTORAL THESIS

Kreĭn Space Methods for Structured Data

Author:

Joseph REDSHAW

Supervisor:

Prof. Jonathan HIRST

A thesis submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

2024

Declaration of Authorship

I, Joseph REDSHAW, declare that this thesis titled, "Kreĭn Space Methods for Structured Data" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

J. Redshaw

Date:

18-01-2024

Abstract

Learning from structured data, including sequences and graphs, is a significant and central challenge in machine learning that has far-reaching applications across numerous disciplines, including chemistry and biochemistry. Kernel methods are undoubtedly an essential tool in this challenge. Their use of a similarity function, known as a *kernel function*, facilitates learning complex relationships from data of arbitrary structure. However, many expressive notions of similarity are not valid kernel functions, meaning they are not applicable to standard kernel methods. Kreĭn space methods are a potential solution to this problem, as they generalise kernel methods to a much larger class of similarity functions. In this thesis, we explore the application of Kreĭn space methods to structured data. Focusing on problems in chemistry and biochemistry, in which structured data and domain-specific similarity measures are commonplace, we investigate to what extent Kreĭn space methods can be utilised to develop supervised learning models for structured data. In particular, we develop models to identify translation initiation sites in nucleic acid sequences, predict the yield of a carbon-nitrogen cross coupling reaction and identify peptides exhibiting antimicrobial properties. We find that the resulting performance of the models is highly dependent on the choice of similarity function and that Kreĭn space methods outperform standard kernel methods in some, but not all, of the domains considered.

Publications

(i) **J. Redshaw**, D. S. Ting, A. Brown, J. D. Hirst, and T. Gärtner (2023). "Kreĭn support vector machine classification of antimicrobial peptides". *Digital Discovery* 2, pp. 502–511.

(ii) A. L. Haywood, **J. Redshaw**, M. W. D. Hanson-Heine, A. Taylor, A. Brown, A. M. Mason, T. Gärtner, and J. D. Hirst (2021). "Kernel methods for predicting yields of chemical reactions". *Journal of Chemical Information and Modeling* 62, pp. 2077–2092.

(iii) A. L. Haywood, **J. Redshaw**, T. Gärtner, A. Taylor, A. M. Mason, and J. D. Hirst (2020). "Machine Learning for Chemical Synthesis". In: *Machine Learning in Chemistry: The Impact of Artificial Intelligence*. The Royal Society of Chemistry, pp. 169–194.

Relevance and Contributions Publication *(i)* is incorporated into the thesis as Chapter 7. As first author, the majority of the publication was contributed by J.Redshaw. Publication *(ii)* is not incorporated into the thesis, but inspired the experiments in Chapter 4. The main contributions of J.Redshaw were: software, machine learning support, review of drafts. Publication *(iii)* is a book chapter that is not incorporated into the thesis. The main contributions of J.Redshaw were: writing (original draft, review, editing).

Acknowledgements

I extend my heartfelt appreciation to Prof. Jonathan Hirst and Prof. Thomas Gärtner. Working with both of you during my studies has been an immense privilege, and I firmly attribute the success of this thesis to your guidance. Jonathan, a special mention is owed to you for your meticulous feedback on numerous drafts. I am grateful to colleagues and collaborators, specifically Alexe Haywood, Daren Ting, Alex Brown, Andy Mason and Adam Taylor. Your invaluable discussions have played a pivotal role in shaping significant portions of this thesis.

Finally, I want to convey my gratitude to my loved ones. Josh, George, and Elise, thank you for your unwavering belief in me. Mia, I could not have done this without your faith and encouragement. Mum, Dad and Dave, thank you for everything.

Contents

List of Figures

List of Tables

List of Abbreviations

AAC	Amino acid composition.
ACC	Accuracy.
AMP	Antimicrobial peptide.
AMR	Antimicrobial resistance.
AUC	Area under the ROC curve.
C-N	Carbon-nitrogen.
CBM	Compression-based similarity measure.
CCM	Compression complexity measure.
DBAASP	Database of antimicrobial activity and structure of peptides.
ECFP	Extended-connectivity fingerprints.
ERM	Empirical risk minimisation.
ETC	Effort-to-compress.
GKM	Gapped k-mer.
IKFD	Indefinite kernel Fisher discriminant.
IKQD	Indefinite kernel quadratic discriminant.
KSVM	Kreĭn support vector machine.
LA	Local alignment score.
LEV	Levenshtein distance.
LK-SVM	Loosli Kreĭn support vector machine.
MCC	Matthews correlation coefficient.
MD	Molecular dynamics.
MIC	Minimum inhibitory concentration.
ML	Machine learning.
OMA	Optimal molecular assignment.
PPM	Prediction by partial matching.
RKHS	Reproducing kernel Hilbert space.
RKKS	Reproducing kernel Kreĭn space.
RMSE	Root mean-squared error.
RRM	Regularised risk minimisation.
SAR	Structure-activity relationship.
SCK	String compression kernel.
SMILES	Simplified molecular input line system.
SMO	Sequential minimal optimisation.
SVM	Support vector machine.
SVR	Support vector regression.
TIS	Translation initiation site.
VCLSM	Variance-constrained least squares method.

For my parents.

Chapter 1

Introduction

Advancements in modern technology have facilitated the production and collection of vast amounts of data from various sources. With the ever-increasing production rate, automating the extraction of knowledge contained within data has become progressively more important. Machine learning is the field of study concerned with precisely this idea: the automated extraction of knowledge from data.

Computational algorithms lie at the core of machine learning. Many of these algorithms assume that data can be represented in Euclidean space (Shalev-Shwartz and Ben-David, 2014). Namely, each instance can be represented as a vector in n-dimensional space. Data not meeting this assumption are known as *structured data*, and are ubiquitous in many fields. Examples include images and videos in computer vision, graph-structured data in chemoinformatics and sequence data in bioinformatics and finance (Passerini, 2013). The inherent structure embedded in these data types contains a wealth of information that can significantly benefit machine learning algorithms. However, exploiting this information is a non-trivial task.

Kernel methods are a class of machine learning algorithms that, among other things, present a principled set of techniques for extracting and exploiting structure in data. They work through a pairwise similarity function known as the *kernel function*. This function embeds the data into a high-dimensional Euclidean space in which a machine learning algorithm can operate (Schölkopf and Smola, 2002). This embedding, however, is performed implicitly. Pairwise evaluation of the kernel function is all that is required. Hence, if one can define and evaluate a kernel function for a given data type, one can perform machine learning directly on that data type.

Numerous expressive notions of similarity exist for structured data, including, for example, the graph-edit distance for graphs, sequence alignment scores for sequence data and dynamic time warping for time series (Gao et al., 2010; Smith and Waterman, 1981; Müller, 2007). Nevertheless, many of these do not satisfy the requirements of a kernel function and, therefore, cannot be incorporated into kernel methods. Henceforth, we refer to similarity functions of this kind as *indefinite kernels*. Recent advancements, however, have generalised several kernel methods to work with a larger class of similarity functions. These methods, known as Kreĭn space methods, operate similarly to kernel methods but with less stringent requirements on which similarity functions are applicable.

The present thesis aims to investigate the efficacy of Kreĭn space methods applied to structured data. We place particular emphasis on problems from chemoinformatics and bioinformatics, in which structured data often

provides a natural description of observed phenomena. In the following section, we introduce the subject of the present thesis. Subsequently, we provide an outlook on the thesis content, summarising the results and achievements.

1.1 Objectives

Learning from structured data is an essential concept in machine learning that has far-reaching applications in many scientific domains. Algorithms to perform this task must fully exploit the information contained within the structure in order to learn meaningful relationships. Through their use of an indefinite kernel function, Kreĭn space methods are well-equipped to perform this task. The fields of bioinformatics and chemoinformatics can significantly benefit from the successful application of these methods since structured data often provides a natural description of observed phenomena. Furthermore, expressive, domain-specific similarity measures in these fields can easily be incorporated into Kreĭn space methods. Hence, the objective of the present thesis is *to understand the efficacy and applicability of Kreĭn space methods applied to non-trivial learning problems originating from bioinformatics and chemoinformatics.* To do so, we focus on the tasks of:

(i) Identification of translation initiation sites (TIS) in sequences of *Arabidopsis Thaliana* mRNA.

(ii) Prediction of chemical reaction yields in a set of Buchwald-Hartwig amination reactions.

(iii) Classification of Antimicrobial Peptides (AMP).

Task (*i*) is the problem of identifying which AUG triple in an mRNA sequence starts the translation process. Translation is a crucial component of gene expression, a fundamental life process by which information encoded in genes is transformed into functional gene products, such as RNA and proteins (Volgin, 2014). Irregularities in translation initiation can lead to many human diseases, including cancers and metabolic disorders (Sonenberg and Hinnebusch, 2009; Jackson et al., 2010; Hershey et al., 2012). In eukaryotes, translation is not always initiated by the first AUG triple, implying that context information surrounding the triple also plays a role. Hence, automated identification of TIS is a task well-suited to machine learning that can benefit genetics research and further the understanding of genetic disorders (Kapur and Ackerman, 2018). We choose this task since it is clearly of biological importance and, using the representation of mRNA molecules as biological sequences, naturally lends itself well to the framework of Kreĭn space methods for structured data.

Task (*ii*) represents a problem with broad applications in the pharmaceutical industry. The Buchwald-Hartwig amination is a palladium-catalysed carbon-nitrogen (C-N) cross-coupling of amines and aryl halides (Torborg and Beller, 2009; Magano and Dunetz, 2011; Ruiz-Castillo and Buchwald, 2016). The aromatic amine products play a vital role in synthesising small drug-like molecules (Vitaku et al., 2014). We choose this task since accurate quantification of reaction yield could support the drug discovery process and streamline laboratory experiments. It is also well suited to structured data methods, owing to the molecular graph representation of molecules. The

optimal molecular assignment kernel (OMA) (Fröhlich et al., 2005) is an indefinite kernel that can directly exploit this representation. Therefore, we consider whether Kreĭn space methods, equipped with the OMA kernel, can provide a suitable methodology for predicting chemical reaction yield.

Task (*iii*) involves the classification of AMPs, also known as host defence peptides. They are a class of molecules that play a crucial role in the innate immune system (Mookherjee et al., 2020; Hancock et al., 2016; Ting et al., 2022). These peptides are typically composed of 12 to 50 amino acids and are highly effective against a wide range of microorganisms such as bacteria, viruses, fungi, and parasites. Moreover, they are less likely to develop antimicrobial resistance (AMR) due to their rapid membrane permeabilising activity (Mookherjee et al., 2020; Ting et al., 2021a; Mayandi et al., 2020). Given their broad-spectrum and rapid antimicrobial activity, researchers are exploring AMPs as a potential solution to the growing AMR problem, a major global health concern (Murray et al., 2022; Ali et al., 2022). Therefore, the computational classification of peptides that exhibit antimicrobial activity could significantly speed up the discovery and development of AMPs for clinical use. Aside from the potential benefits that the automated classification of AMPs could deliver, this is another task for which learning from structured data is a suitable framework. Peptides can be represented as biological sequences, a representation that can be utilised by standard, domain-specific similarity measures such as the sequence alignment algorithm of Smith and Waterman (1981). Given that many of these similarity measures are, in fact, indefinite kernels, the classification of AMPs naturally lends itself

to the learning paradigm of Kreĭn space methods for structured data.

1.2 Overview and Contributions

In this section we provide an overview of the structure of the thesis and detail the contributions we have made to the fields of bioinformatics, chemoinformatics and machine learning research. The remaining thesis consists of 7 chapters, which are broken down as follows:

Chapter 2, Review of Current Literature The next chapter reviews the current literature pertaining to kernel methods and their application to structured data. We focus our discussion on a few key papers relevant to problems involving molecules and biological sequences. In particular, we begin by reviewing kernel methods applied to molecules and reactions. We centre our discussion on works utilising the SMILES, molecular fingerprint or molecular graph representation of molecules. This is followed by a review of literature applying kernel methods to problems involving biological sequences, in which we focus on methods that utilise alignment, compression and string kernels. Finally, we discuss the current literature comprising indefinite kernel methods. We focus on two categories of approaches: those that transform an indefinite kernel into one which is positive-definite and those that work directly with an indefinite kernel, Kreĭn space methods.

Chapter 3, Machine Learning This chapter covers the background mathematical theory of some fundamental elements of machine learning, which

are required to understand the remaining chapters. We start by providing an overview of learning theory and how a predictive function can be learnt from the principle of regularised risk minimisation (RRM). This is followed by a first-principles derivation of the linear support vector machine (SVM) and the linear support vector regression (SVR) algorithms. We further show how the theory of convex optimisation can be employed to reveal deeper insights into SVMs. We conclude the chapter by demonstrating how SVMs can learn non-linear decision functions through the use of kernels.

Chapter 4, Kreĭn Space Methods for Structured Data Using the theory developed in Chapter 3, this chapter presents the main methodological contributions of the thesis. Initially, we discuss the fundamental theory of Kreĭn spaces and the notion of a Reproducing Kernel Kreĭ Space (RKKS). Then, we present the first major methodological contribution of the chapter. In particular, we provide the first formal derviation of the dual formulation of a support vector machine (SVM) in a Kreĭn space. The notion of *duality*, which we make explicit in section 3.2.3, states that some optimisation problems can be viewed from either of two perspectives, the *primal* or the *dual*. Transforming the original problem, the primal, into the dual can reveal new insights and, in some cases, simplify the problem. The Kreĭn-SVM, a generalisation of the SVM to indefinite kernels, was initially proposed by Oglic and Gärtner (2019) in its primal form. We provide the first complete derivation of the Kreĭn-SVM dual, showing that it can be solved via quadratic programming. This is followed by the second major methodological contribution of the thesis.

In particular, we apply the notion of Kolmogorov Complexity (Cover, 1999; Kolmogorov, 1965) to develop a novel, indefinite string kernel, *the string compression kernel* (SCK). The Kolmogorov complexity of an arbitrary digital object is the length of the shortest computer program, in a given programming language, which produces the object as output. Whilst conceptually simple, it is not a computable function. However, it can be well-approximated by real-world compression software. The SCK makes use of this observation in order to define a similarity measure over strings, using a number of common compression algorithms to approximate the Kolmogorov Complexity.

Chapter 5, Identification of Translation Initiation Sites This chapter provides the first empirical contribution of the thesis. The Kreĭn-SVM and SCK developed in Chapter 4, alongside a dataset of *Arabidopsis Thaliana* mRNA sequences, are utilised to develop classification models that identify the location of TIS codons. Using several standard compression algorithms, we compare our approach to the state-of-the-art positive-definite kernel method for this task, an SVM equipped with the locality-improved (LI) kernel (Zien et al., 2000). Our methodology is initially validated on a dataset of synthetic strings, distributed according to a k-order Markov model. This particular distribution is well-suited to the PPM compression function since, internally, it uses a k-order Markov model to encode the sequence of characters. Indeed, our experiments confirm this, as the PPM model drastically outperforms the LI baseline. Our methodology is then evaluated on the real-world

TIS dataset. In this case, the PPM compression algorithm (Cleary and Witten, 1984) leads to the best-performing Kreĭn-SVM model, achieving similar results to the positive-definite LI baseline.

Chapter 5, Prediction of C-N Cross Coupling Reaction Yield This chapter presents another empirical contribution of the thesis. Utilising the regression algorithm of Oglic and Gärtner (2018), equipped with the OMA kernel of Fröhlich et al. (2005), we develop and evaluate a Kreĭn space model to predict the yield in a set of Buchwald-Hartwig amination reactions. Models were developed using an open-source combinatorial dataset, in which each reaction consisted of an additive, aryl halide, base, ligand and experimentally validated yield (Ahneman et al., 2018). Our approach was compared to a baseline support vector regression model equipped with the positive-definite Weisfeiler-Lehman graph kernel, a methodology that has previously been demonstrated to perform well on this dataset (Haywood et al., 2021). In a set of in-sample experiments, our methodology performs similarly to the baseline, with approaches achieving a small error. However, in a set of out-of-sample experiments, in which certain additives were completely removed from the training set, our approach, as well as the baseline, suffers a degradation in performance. The results demonstrate that the proposed methodology is applicable when operating in known regions of chemical space but fails to suitably generalise to unknown regions.

Chapter 7, Classification of Antimicrobial Peptides This chapter presents the main empirical contribution of the thesis. Utilising the Kreĭn-SVM developed in Chapter 4, in conjunction with standard sequence alignment measures, we develop models to classify peptides as antimicrobial. More specifically, we utilise the Smith-Waterman alignment score and Levenshtein distance to develop models capable of detecting both *general* and *species-specific* antimicrobial activity in peptides. We utilise two datasets from the literature to evaluate our methodology in identifying peptides exhibiting general antimicrobial activity. We compare against two in-house positive-definite baselines and pre-established models associated with each dataset. On both datasets, either the Smith-Waterman alignment score or Levenshtein distance leads to the best-performing model. Utilising another dataset from the literature, we further assess our methodology by developing models capable of identifying activity against specific species. We test these models on a small set of in-house, experimentally validated peptides. In this case, one of the positive-definite baselines outperforms our proposed approach. Nevertheless, the general and species-specific models are made freely available as web applications at `http://comp.chem.nottingham.ac.uk/KreinAMP`.

Chapter 8, Conclusion The final chapter summarizes the content of the previous chapters and suggests possible avenues to improve our work.

Chapter 2

Review of Current Literature

This section reviews the current literature on kernel methods and their application to structured data. Given the domain of the tasks laid out in the previous chapter, we focus our discussion on the application of kernel methods to problems in bioinformatics and chemoinformatics in which structured data representations have been utilised.

We start our review with a discussion of the application of kernel methods to chemoinformatics, focusing on the SMILES string, molecular fingerprint and molecular graph representation of molecules. We choose these representations since they are commonplace in chemoinformatics (Wigh et al., 2022) and, as we will see, are widely used when applying kernel methods to chemoinformatics. Following this, we move on to review the application of kernel methods to bioinformatics. We focus our discussion on approaches that use alignment, compression, and string kernels. Chapters 5 and 7 both use these approaches, and we have aimed to keep the discussion relevant to our work. In terms of the specific applications to chemoinformatics and bioinformatics, most of the review discusses supervised learning. In

particular, the prediction of physical, chemical and structural properties of molecules, reactions and biological sequences. The subject of the thesis is wholly centred around supervised learning, and, as before, we have focused on keeping the discussion relevant.

The review's final section discusses various methods to learn functional relationships with indefinite kernels. We group the methods into two categories: those that modify the kernel function and those that modify the learning algorithm. The former category of methods transforms an indefinite kernel function into one that is positive definite. The latter category of methods devise learning algorithms that naturally incorporate indefinite kernel functions; we refer to these as Kreĭn Space methods.

In order to provide context for the rest of the chapter, the following section provides a qualitative description of kernel methods. For the sake of brevity, a complete, quantitative description is left until Chapter 3.

2.1 Kernel Methods for Structured Data

Kernel methods are a class of machine learning algorithms, the most notable example being the Support Vector Machine (SVM). They have been successfully applied to several real-world problems and are considered a standard tool for a machine learning practitioner. Much of this success stems from the kernel trick, which facilitates the modelling of non-linear relationships with linear algorithms through the utilisation of a *kernel function*. The class encompasses algorithms to solve many problems encountered in machine learning,

including, for example, classification, regression and dimensionality reduction, to name but a few (Schölkopf and Smola, 2002). A standard undergraduate statistics course will cover linear algorithms to solve all the aforementioned problems, for instance, logistic regression, linear regression and PCA, respectively. However, each of these algorithms has an equivalent kernel method, namely, kernel logistic regression, kernel ridge regression and kernel PCA. These can be seen as a non-linear generalisation of their linear counterparts. Indeed, many well-known linear algorithms can be viewed as a special case of a more general kernel method.

Using a kernel method as opposed to its linear counterpart is beneficial for a number of reasons. As previously stated, kernel methods are capable of modelling arbitrary, non-linear relationships. Many real-world problems require greater flexibility than a linear method can provide. Allowing for greater modelling flexibility increases the types of problems that can be solved with a given algorithm. As a small example, consider modelling the function $f(x) = x^2$ with linear regression. Try as one might, the resulting model would not be able to capture the true functional relationship. On the other hand, there are a plethora of kernel methods equipped to solve this problem.

Another benefit of using a kernel method is the applicability to non-vectorial data. By this, we mean data that cannot be naturally represented in a table. Common examples, some of which we discuss in more detail in later sections, include strings, graphs and trees. Linear algorithms assume that instances of data are represented as n-dimensional vectors. Without first representing,

say, a set of strings as a set of vectors, it would not be possible to apply any of the standard linear algorithms. Furthermore, there are many such possible representations, and it is not a trivial task to choose one that is suitable to the given problem. On the other hand, kernel methods are directly applicable to non-vectorial data. As long as a kernel function exists for the considered data type (which, in most cases, is true), it is straightforward to apply a kernel method directly to non-vectorial data, thus bypassing the need to find a suitable representation.

As one would expect, the added benefits of using a kernel method does not come without its drawbacks and we would be remiss without mentioning these. The transition from linear to kernel methods, whilst allowing for greater modelling flexibility, also increases the complexity of deriving a suitable model. Inherent to all kernel methods are hyperparameters. As opposed to other parameters of the model, which we refer to as *weights*, hyperparameters are those which are decided *a priori*. The process of computing model weights, commonly known as fitting a model, is defined by the learning algorithm and is therefore usually abstracted away from the end user. With many kernel methods, fitting a model is a deterministic procedure that will always return the same set of weights given the same input. That is, given a set of inputs, the learning algorithm is a procedure to fit a model. On the other hand, the procedure of choosing hyperparameters sits on top of fitting a model. A practitioner wants to select the best set of hyperparameters such that, when provided to the learning algorithm, the fitted model is in some sense optimal. This is a task that cannot be overlooked, since the resulting model can

be sensitive to the choice of the provided hyperparameters. *Hyperparamter optimisation*, the procedure of choosing a set of optimal hyperparameters for a learning algorithm, is largely performed in an experimental fashion. Whilst the fundamental process of selecting suitable hyperparameters is performed via experimentation, there have been many recent developments to automate and streamline this (Shahriari et al., 2015; Chapelle et al., 2002; Yu and Zhu, 2020). AutoML, a sub-field of machine learning, considers learning algorithms as hyperparameters. More specifically, AutoML performs the simultaneous selection of a machine learning algorithm and its hyperparameters, thereby abstracting away many of the repetitive tasks one faces when building a machine learning model (Yu and Zhu, 2020; Karmaker et al., 2021; He et al., 2021). There are three main sources of hyperparameters in a kernel method. The learning algorithm itself usually defined with at least one hyperparameter. Furthermore, there typically exist many appropriate kernel functions for the data type considered, the decision of which kernel function to use can also be viewed as one of selecting a hyperparameter. Finally, most kernel functions themselves include hyperparameters. Compared to standard linear algorithms with no hyperparameters, this added complexity can seem daunting to the end user. However, it is often the case that the improved modelling capabilities outweigh the increased overhead of hyperparameter optimisation.

Another main drawback of using kernel methods is that of computational complexity. Fundamental to kernel methods is the kernel matrix. This is a matrix whose elements consist of evaluation of the kernel function at each

pair of instances. Hence, the size of this matrix grows as $O(N^2)$ for a data set of size N. With modern data sets consisting of tens of millions of instances, application of kernel methods can be computationally prohibitive. In their standard implementation, it is necessary to compute and store the whole kernel matrix as a stage in fitting a kernel method. Furthermore, a number of kernel methods require the inverse of this matrix; an operation that scales as $O(N^3)$. These issues, however, have been understood for many years. There exist numerous approximation methods to alleviate the computational burdens associated with the kernel matrix (Rahimi and Recht, 2007; Bach, 2013; Rudi et al., 2015; Rudi and Rosasco, 2017; Rudi et al., 2017; Oglic and Gärtner, 2017).

Having provided an overview of kernel methods, we proceed to discuss their application to structured data. A broad overview of kernel methods for structured data can be found in the following references: Passerini (2013), Gärtner (2003), Schölkopf et al. (2004), Schaid (2010), Borgwardt (2011), Saunders and Demco (2007), Ralaivola et al. (2005), and Akutsu and Nagamochi (2011).

2.1.1 Molecules and Reactions

The synthesis of new molecules is fundamental to the development of the field of synthetic chemistry. However, synthesising a molecule is both a time-consuming and costly procedure that is usually carried out on a trial-and-error basis. This long-winded process has the potential to be automated and optimised, which in turn could reduce expenditure and allow synthetic

chemists to spend more time designing molecules. This section discusses current computational approaches to problems in synthetic chemistry, focusing on kernel methods and their various molecular representations.

SMILES

The Simplified Molecular Input Line System (SMILES) is a chemical notation language that uniquely specifies two-dimensional molecular structures (Weininger, 1988). The uniqueness of the representation is a consequence of the rules that govern the SMILES specification. As such, a SMILES string is a string of ASCII characters that uniquely represents a molecule. The basic rules for constructing a SMILES string are given in Table 2.1. It is important to note that, although each valid SMILES string corresponds to a unique molecule, the converse is not true; each molecule can have more than one valid SMILES representation. It is for this reason that the canonical SMILES exists. This defines an extension to the SMILES representation in which each molecule is assigned one, unique SMILES string. Figure 2.1 indicates all the valid SMILES representations of a toluene molecule, as well as the canonical representation.

Molecular feature	Representation
Non-aromatic atoms	Element symbols, first letter in upper case letters
Single bonds	- (but may be omitted)
Double bonds	=
Branching	Denoted using ()
Ring closure	Label (with the same number) atoms that were connected to each other
Aromatic atoms	Lower case letters

TABLE 2.1: Basic rules for constructing a SMILES string (Hirst et al., 2023).

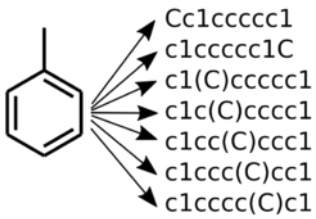


FIGURE 2.1: A toluene molecule accompanied by its numerous SMILES representations. The first SMILES is the canonical SMILES (Bjerrum, 2017).

SMILES are often used to store molecules in a database, since their condensed representation allows for minimal storage overhead and efficient querying. However, their one-to-one mapping between molecules has prompted researchers to consider whether predictive models can be developed based solely on the SMILES string. Perhaps the first authors to consider this idea were Swamidass et al. (2005), in which they compare one, two and three dimensional kernels for small molecules on a number of real-world classification data sets. The one-dimensional kernels are simply string kernels taking SMILES strings as input (Leslie et al., 2003). These are used in conjunction with an SVM in order to classify compounds indicating carcinogenicity in mice and rats, mutagenicity in *Salmonella typhimurium* and inhibition of cancer growth across human tumour cell lines. Interestingly, in all cases, the resulting models achieve accuracies much greater than that of random guessing. However, generally speaking, the one-dimensional kernels do not produce models as accurate as those with two-dimensions. Cao et al. (2012) observe similar results for the task of compound toxicity classification. They

compare a string kernel on SMILES strings with a Gaussian kernel on common molecular descriptors and find both methods perform comparably, resulting in accurate classifiers. It is surprising to observe that the encoding of molecules as strings retains sufficient information to produce accurate classifiers for such a diverse set of problems.

Fingerprints

Molecular fingerprints are a class of representations of (usually 2D) molecular structures that represent a molecule as a bit vector. This class can be further subdivided into different sets that are defined through the means in which the fingerprint is calculated. For the purpose of this thesis, we focus on two of the main fingerprinting methods, namely, substructure keys-based fingerprints and circular fingerprints (Cereto-Massagué et al., 2015).

Substructure keys-based fingerprints set the bits in the fingerprint vector using predefined substructures. The number of bits is determined by the number of predefined substructures, with a bit i being set to 1 if substructure i is present in the molecule, and set to 0 otherwise. One popular method is the MACCS (MDL) fingerprint. It comes in two variants, one with 166 and the other with 960 predefined substructures (Cereto-Massagué et al., 2015; Durant et al., 2002). The shorter one is the more common of the two since it covers most of the interesting chemical features for drug discovery and virtual screening (Cereto-Massagué et al., 2015). Circular Fingerprints work by analysing a neighbourhood of each atom (all paths of a set length originating

from an atom) in a molecule, with most variants based on the Morgan algorithm (Morgan, 1965). The substructure contained within the neighbourhood is hashed to produce the relevant bit index (Cereto-Massagué et al., 2015). One notable and common method is the Extended-Connectivity Fingerprints (ECFP) (Rogers and Hahn, 2010). This method is defined by a diameter parameter, which controls the size of the neighbourhood of each atom. An immediate improvement over substructure keys-based fingerprints is that ECFP can represent novel structural classes, since they not defined *a priori* (Rogers and Hahn, 2010).

Since fingerprints map a molecule to a bit vector, they naturally induce a kernel function that is simply the inner product of the bit vectors. However, a more common approach is to use fingerprints in conjunction with the Tanimoto coefficient (Tanimoto, 1958), as this also induces a kernel function. Furthermore, the vectorial representation of a fingerprint can be combined with any common vector kernel function, including, for instance, the Gaussian kernel. Haywood et al. (2021) explore this idea in more detail, in which they perform a detailed comparison of numerous molecular descriptors and vector kernel functions. Included in the comparison are MACCS fingerprints, circular fingerprints as well as the Tanimoto coeffcient. The comparison is performed for the task of yield prediction in the Buchwald-Hartwig amination reaction (Ahneman et al., 2018), using the support vector regression (SVR) algorithm. Of the various descriptors tested, fingerprints were rated the highest for generalisability, outperforming a set of expensive quantum chemical descriptors. Fingerprints are also used in the work of Ullrich et

al. (2016), in which they are combined with other descriptors to predict the binding affinity of 20 ligands. They observe substantial improvements when combining multiple descriptors as opposed to any one single descriptor. In one experiment in particular, they find that combining MACCS and ECFP fingerprints produces models with smaller prediction errors than those using either fingerprint separately.

Extensions to fingerprints have also been considered by Swamidass et al. (2005). Through the use of a hash function, standard circular fingerprints produce bit vectors of a predetermined length. It may be the case that more substructures exist than the length of the bit vector. Hence, the compression to a predetermined length is a source of information loss. To circumvent this issue, the authors propose to use long bit vectors with a unique bit position reserved for each possible path. They compare these two-dimensional representations with one and three dimensional representations. From an analysis over numerous classification tasks, they conclude that two-dimensional fingerprints may be preferred.

Graphs

Molecular graphs, first considered by Cayley (1874), describe molecules as a vertex and edge labelled mathematical graph. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges. Using a vertex labelling function $f_{\mathcal{V}}$, each vertex is labelled by its atomic element. Similarly, the edge labelling function $f_{\mathcal{E}}$ labels each edge by its bond type. Whilst atomic elements and bond types are standard labels for the vertices and edges, it is

possible to provide more contextual information. Ceroni et al. (2007) describe a kernel function that accounts for two and three dimensional structural information in small molecules. In this case, the vertices of the molecular graph are labelled with the atom type, partial charge and its three-dimensional coordinates. In a set of experiments to classify compounds active against HIV as well as those that inhibit cancer growth, they found that providing both the two (atom type, partial charge) and three dimensional information produced more accurate models than using either separately. Fröhlich et al. (2005) also explore this idea, in which they define a kernel function that accounts for numerous properties of atoms and bonds. This kernel, known as the Optimal Assignment kernel, compares two molecules by finding a maximal similarity mapping from the atoms of one molecule to another. It does so by comparing not only the properties of the considered atom but also those of its surrounding neighbourhood. In a set of molecular regression and classification experiments, they find their approach produces less error than a baseline approach that also uses the atom and bond properties, but does not perform the optimal assignment. We discuss their approach in more detail in section **??**.

As we have seen, embedding contextual information into the molecular graph can aid in the development of predictive models. The previously described approaches were defined for small molecules, yet the concept of a graph is more general than the two-dimensional description of a molecule. There exist numerous graph kernels that, whilst not designed with molecules in mind, have been shown to excel in tasks relating to chemoinformatics. One

notable example is the Weisfeiler-Lehman kernel function (Shervashidze et al., 2011). This kernel, which we discuss further in **??**, is inspired by the Weisfeiler-Lehman test of isomorphism on graphs (Leman and Weisfeiler, 1968). For two graphs to be isomorphic, they must contain the same number of vertices and be connected in the same way. That is, they must be topologically equivalent. Whilst no efficient algorithm exists for deciding graph isomorphism, the resulting Weisfeiler-Lehman kernel has proven to be very useful in all manner of machine learning problems involving graphs. In their comparison of numerous molecular descriptors, Haywood et al. (2021) also consider the Weisfeiler-Lehman kernel. They rank this descriptor as second best in terms of generalisability, placing only behind molecular fingerprints. Both methods are descriptors that account for structural information, hence there is certainly benefit to using the molecular graph representation.

All the approaches we have previously considered are problems of predicting properties of a given molecule. Oglic et al. (2018) turn this problem on its head and, instead, search for a molecule with a given property. This approach, known as *lead discovery*, is a crucial stage of the drug development life-cycle (Hughes et al., 2011). It is a time-consuming process that is usually performed in a lab but can greatly benefit from computational assistance (Sliwoski et al., 2014). Oglic et al. develop a novel algorithm to explore the space of possible molecules in search of those that exhibit large binding affinity to a specific receptor, using the Weisfeiler-Lehman kernel as a molecular descriptor.

2.1.2 Biological Sequences

Understanding the structure and function of the molecules which make up and control living organisms is a significant aim of molecular biology. Two categories of molecules of particular interest to the field are nucleic acids and proteins. These are both chain molecules, consisting of a small set of individual components which, when joined together in a chain, form complex molecules that dictate many of the biological processes in living organisms. The chain-like structure allows one to represent these molecules as sequences of characters from a given alphabet. Indeed, a DNA molecule can be represented as a sequence defined over the alphabet $\Sigma = \{A, C, G, T\}$. Similarly, a protein can be viewed as a sequence of characters taken from the alphabet of amino acids. The representation of nucleic acids and proteins as sequences facilitates their study through the lens of sequence analysis. In what follows, we discuss the application of kernel methods to biological sequence problems, focusing on those utilising sequence alignment, compression and string kernels.

Alignment

Pairwise sequence alignment is a fundamental tool in bioinformatics, aiming to establish a correspondence between the characters in two sequences. It has allowed practitioners to analyse the similarity of pairs of biological sequences and detect ancestral, structural or functional similarities between them. The general aim of an alignment is to associate the characters of two

Sequences	Global Alignment	Local Alignment
HEAGAWGHEE	HEAGAWGHEE	AWGHE
PAWHEAE	---PAWHEAE	AW-HE

TABLE 2.2: An example of the optimal pairwise alignments of two amino acid sequences. The global and local alignments were computed using the EMBOSS Needle and EMBOSS Water services, respectively (Madeira et al., 2022). Both methods used the BLOSUM62 scoring matrix and default gap penalties The identical regions are displayed in green and gaps are denoted by "-".

sequences such that a given measure of similarity is maximised. The resulting alignment is known as the *optimal alignment*. Each association between two characters induces a score, and the sum of these scores is maximised when computing the optimal alignment. However, deciding the score for a pair of characters is a non-trivial task that can drastically affect the resulting alignment. In the context of amino acid sequences, BLOSUM matrices are a standard choice since they provide scores that reflect observed evolutionary relationships (Henikoff and Henikoff, 1992). When performing an alignment, a practitioner must decide between a global and local alignment. A global alignment produces a correspondence between all characters in both sequences. Using a global alignment implicitly assumes that the two sequences are related in their entirety. A local alignment does not make this assumption, as it finds the two maximally aligned subsequences of the considered sequences. An example of both is provided in Table 2.2.

Sequence alignments are limited in kernel methods since the resulting alignment score is not a valid kernel function (see Definition 3.3.5), meaning they cannot directly be applied to a kernel method. However, one can implicitly incorporate alignment scores into kernel methods through certain

computational tricks. The pairwise-SVM method is one such trick, in which, for a data set of N sequences, a sequence is represented by a length N vector containing the optimal alignment scores of that sequence with every other sequence in the data set (Tsuda, 1999). Under this vector representation, a standard SVM can be applied. Liao and Noble (2003) use the pairwise-SVM method to detect homologous proteins, demonstrating that it significantly outperforms all other considered approaches, including the state-of-the-art. One competing method in their comparison is the pairwise-SVM using BLAST scores (Altschul et al., 1990). The BLAST algorithm is a heuristic alignment tool that sacrifices optimality for computational efficiency. As one would expect, the model using non-optimal alignments is far inferior to its optimal counterpart. This has also been observed in other protein classification tasks (Kocsor et al., 2006; Shah et al., 2008). Vert et al. (2004) generalise the optimal local alignment score by defining a similarity measure that accounts for all possible local alignments instead of just the optimal one. Whilst this doesn't lead to a valid kernel function, it can still be incorporated under the pairwise-SVM framework. Their experiments indicate that the generalised score leads to more accurate models than the optimal local alignment score.

Compression

In the context of computer science, compression is the process of encoding a digital object in a representation that is smaller than its original format. In particular, since all digital objects can be represented as a sequence of bits,

compression aims to represent the object with fewer bits by identifying and removing redundant information. Many statistical and machine learning algorithms operate in a similar way. Indeed, the process of dimensionality reduction can be viewed as one of data compression (Van Der Maaten et al., 2009). With modern compression algorithms being extremely efficient, it is natural to consider whether they can be incorporated into a machine learning process (Gupta et al., 2017). Kocsor et al. (2006) investigate this idea in the context of SVMs. They utilise a compression-based similarity measure (CBM) that quantifies the relative improvement in compression of a given string when another string is provided as input. In a set of protein homology detection experiments, they find that CBM-based models outperform those based on BLAST scores. However, the CBM-based models fall short of models incorporating the optimal alignment score. Yet, interestingly, when using a combination of CBMs and BLAST scores, the resulting models match and, sometimes, outperform those based on the optimal alignment score. Tangent to the notion of compression are compression-complexity measures (CCMs) such as the Lempel-Ziv complexity (Lempel and Ziv, 1976). These are functions which take a string as input and produce a number which, in some sense, quantifies its complexity. Munagala et al. (2022) make use of CCMs to classify coronavirus sequences that cause COVID-19. Utilising the both Lempel-Ziv and Effort-to-Compress (ETC) complexity measures (Nagaraj et al., 2013), they are able to correctly identify 98% of coronavirus sequences. Whilst not related to biological sequences or kernel methods, we mention the work of Melville et al. (2007) as it presents an interesting application of

CBMs. Melville et al. develop a compression-based tool to perform similarity searching in a database of chemical compounds. Using only the SMILES representation of a molecule and *gzip* compression software (which comes pre-installed on most Unix-based operating systems), they develop a tool capable of accurate similarity searching that is competitive with state-of-the-art techniques based on molecular fingerprints.

String Kernels

The use of string kernel models in computational biology is common practice, primarily due to their simplicity, predictive power and computational efficiency. Perhaps the simplest example of a string kernel is the spectrum kernel (Leslie et al., 2001). This kernel considers the k-spectrum of a given string, which is the set of all k-length subsequences it contains. Under this kernel, a string is mapped to a vector representation indexed by all possible k-length subsequences of a given alphabet Σ. For a given k-length subsequence in the alphabet, the value at the corresponding index of this representation is the number of occurrences of that subsequence in the k-spectrum of the string. Evaluation of the kernel for a pair of strings is simply the inner-product of the two vectors. Leslie et al. (2001) test the spectrum kernel's efficacy for remote homology detection in protein sequences. They find that the spectrum kernel performs comparably with some state-of-the-art homology detection methods (at the time of publication). It does, however, fall behind the best performing method.

The introduction of the spectrum kernel encouraged the creation of numerous string kernels that were largely inspired by it. The mismatch kernel (Leslie et al., 2003) is one such kernel that is an extension of the spectrum kernel that also allows for a certain number of mismatches. Using the same vector representation as the spectrum kernel, the (k, m)-mismatch kernel counts the occurrences of a given k-length subsequence in a string, as well as those that differ from it by at most m mismatches (substitution of characters). Leslie et al. (2003) perform the same remote homology detection experiment as that of the spectrum kernel. Their results indicate that the mismatch kernel performs as well as the best-known method, representing a substantial improvement over the spectrum kernel. Further modifications to the spectrum kernel include the substitution kernel, wildcard kernel and gappy kernel (Leslie et al., 2004). The substitution kernel is analogous to the mismatch kernel but only allows for mismatches whose probability of occurring exceeds a predetermined threshold. Occurrence probabilities are estimated from alignment substitution matrices and, hence, are biologically well-founded (Schwartz, 1978; Henikoff and Henikoff, 1992). The wildcard kernel augments the alphabet with a wildcard character, representing the presence of any symbol. Occurrences are counted whilst allowing for a maximum number of wildcard characters per subsequence. The gappy kernel, which is similar in style to the mismatch kernel, counts the occurrences of k-length subsequences but also allows for a certain number of gaps. Whilst Leslie et al. (2004) show that all variants lead to accurate models on protein homology tasks, the resulting models are usually difficult to interpret. Shrikumar et al. (2019) develop a

method to explain the output of gappy kernel models and utilise it in detecting motifs that play an essential role in transcription factor binding.

The previously mentioned *k*-spectrum kernels define a general family of kernels that have found applications in all manner of biological sequence tasks. However, for certain tasks, kernels crafted for the problem at hand can often lead to superior models. Zien et al. (2000) describe a kernel designed explicitly for the task of TIS identification. The resulting kernel, known as the locality-improved kernel, reflects the biological hypothesis that local correlations play a more critical role than distant correlations in the TIS process. Hence, the locality-improved kernel is designed to emphasise local correlations more. In a set of experiments, it is demonstrated that the specifically designed kernel outperforms two previous state-of-the-art methods. We discuss this kernel in greater detail in Section 5.2.2. Similarly, Rätsch and Sonnenburg (2004) describe a kernel designed to reflect important contextual information governing the location of splice sites in nucleotide sequences. Crucially, as is the case with TIS identification, local correlations play a key role, and they design a kernel, known as the weighted degree kernel, to reflect this. In an experiment comparing many state-of-the-art splice site models, they find that the weighted degree and locality-improved kernels are best suited to the task.

The generic string kernel proposed by Giguere et al. (2013) can be viewed as a generalisation of the weighted degree kernel, as well as numerous others, that incorporates the physicochemical properties of amino acids into its computation. The kernel acts similarly to the *k*-spectrum kernels in that small

Input Sequence		2-spectrum representation				
		AA	AB	BA	BB	
ABBA	$\longrightarrow$	[0	1	1	1	]
AABBB	$\longrightarrow$	[1	1	0	2	]

TABLE 2.3: The 2-spectrum representation of the strings ABBA and AABBB under the alphabet $\Sigma = \{A, B\}$.

subsequences of an amino acid sequence are compared to one another. However, this comparison is performed on the basis of the physicochemical properties of the considered subsequences. Furthermore, the comparisons are weighted by the distance in the starting positions of the two subsequences, with those occurring in similar positions providing more significant contribution. Giguere et al. combine the generic string kernel with one designed for protein binding pockets (Hoffmann et al., 2010) and another designed for protein secondary structure (Qiu et al., 2007) in order to estimate the binding affinity of peptide-protein pairs. In a set of experiments, the proposed method outperforms the kernel method baselines as well as previous state-of-the-art models. In a separate work, Giguere et al. apply the generic string kernel to the problem of finding peptides of maximal antimicrobial activity (Giguere et al., 2015). The specific structure of the kernel allows them to pose the problem of constructing a peptide with high activity as one of finding a path of maximal length in a graph. In a comparison against a random selection of peptides, the average and maximum activity of the peptides generated by their approach was much greater than those that were randomly sampled. An *in vitro* experiment, in which several identified peptides were synthesised and measured against *Escherichia coli* and *Staphylococcus aureus*,

confirmed the validity of their method.

2.2 Indefinite Kernel Methods

Standard kernel methods operate using a pairwise similarity function known as the kernel function. In order for a given similarity function to be a valid kernel function, it must be a symmetric, positive-definite function (see definition 3.3.3). Whilst it is trivial to transform a similarity function into one that is symmetric, many expressive notions of similarity do not satisfy the positive-definite requirement. This is particularly true for structured data, in which domain-specific similarity measures are commonplace. Similarity functions of this kind, known as indefinite kernels, cannot directly be applied to standard kernel methods. This issue can be circumvented, but requires either modification of the indefinite kernel or modification of the kernel method. In this section, we discuss current approaches to learning with indefinite kernels.

2.2.1 Transformation Methods

The first class of approaches that we consider are those which transform an indefinite kernel into one which is positive-definite. The general motivation for these methods is that the indefinite part of the kernel is merely a nuisance to be removed. Whilst it has been shown that this assumption does not always hold (Oglic and Gärtner, 2018) and, indeed, useful information

resides in the indefinite part of the kernel, transformation methods are still an attractive approach to facilitate learning with indefinite kernels.

An indefinite kernel can be identified as one whose resulting kernel matrix (see section 3.3.3) has negative eigenvalues. A matrix of this form is known as *indefinite*, whereas a matrix whose eigenvalues are non-negative is known as *positive semi-definite*. The class of approaches we consider in this section are those which transform the indefinite kernel matrix into one which is positive semi-definite. These can broadly be grouped into two categories: spectrum transformation approaches and proxy matrix approaches. The former can be categorised as methods which operate on the set of eigenvalues of the indefinite kernel matrix, producing a new set of non-negative eigenvalues. The latter encapsulates approaches which look for a positive semi-definite matrix that is, in some sense, representative of the indefinite kernel matrix. Given the importance of eigenvalues in the proceeding discussion, we first clarify exactly what is meant by the eigenvalue of a matrix.

Eigenvalues

Given a square matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, a vector $\mathbf{x} \in \mathbb{R}^n$ is called an *eigenvector* of $\mathbf{K}$ if, for some $\lambda \in \mathbb{R}$, the following holds:

$$\mathbf{K}\,\mathbf{x} = \lambda\,\mathbf{x}\,. \tag{2.1}$$

That is, an eigenvector of $\mathbf{K}$ is a vector which, when transformed by $\mathbf{K}$, results in a scalar multiple of itself. The scalar multiple λ is known as the corresponding *eigenvalue*. The complete set of eigenvalues of a matrix is known as the *spectrum* of the matrix. In general, the eigenvalues and eigenvectors of a matrix are complex-valued. However, for a real, symmetric matrix, they are guaranteed to be real-valued. Furthermore, every $n \times n$ matrix has exactly n eigenvector, eigenvalue pairs. This fact leads to the well-known *eigendecomposition* of a matrix. Namely, let $\mathbf{V} \in \mathbb{R}^{n \times n}$ be a matrix whose ith column equals the ith eigenvector of $\mathbf{K}$ and let $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ be the diagonal matrix[1] whose ith diagonal element equals the ith eigenvalue of $\mathbf{K}$. The eigendecomposition states that $\mathbf{K}$ can be factorised as

$$\mathbf{K} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^{-1}. \tag{2.2}$$

Furthermore, if $\mathbf{K}$ is a real, symmetric matrix then $\mathbf{K}$ can be factorised as

$$\mathbf{K} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^{T}. \tag{2.3}$$

Spectrum Transformation

The class of spectrum[2] transformation approaches perform operations on the eigenvalues of the indefinite kernel matrix, resulting in a matrix whose eigenvalues are non-negative. In the remainder of this discussion, we consider a real, symmetric kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ with eigendecomposition

[1] A diagonal matrix is one whose only non-zero elements are those along the main diagonal.

[2] The spectrum of a matrix is defined as set of its eigenvalues.

$\mathbf{K} = \mathbf{V}\,\mathbf{\Sigma}\,\mathbf{V}^T$.

Perhaps the most intuitive approach, the *spectrum clip* transformation simply sets all negative eigenvalues to zero. It boasts the attractive property of producing the nearest positive semi-definite matrix, in terms of the Frobenius norm (Higham, 1988). The resulting kernel matrix $\mathbf{K}_{\text{clip}}$ is expressed as

$$\mathbf{K}_{\text{clip}} = \mathbf{V}\,\mathbf{\Sigma}\,\text{diag}(I_{\lambda_0>0}, \ldots, I_{\lambda_n>0})\,\mathbf{V}^T, \tag{2.4}$$

where I is an indicator function.

In a similar vein, the *spectrum flip* transformation maps each eigenvalue to its absolute value. Retention of the absolute values can be useful in situations where the negative eigenvalues contain important information (Pękalska et al., 2004). The resulting kernel matrix $\mathbf{K}_{\text{flip}}$ is written as

$$\mathbf{K}_{\text{flip}} = \mathbf{V}|\mathbf{\Sigma}|\,\mathbf{V}^T, \tag{2.5}$$

where $|\mathbf{\Sigma}|$ denotes the diagonal matrix whose elements are the absolute values of $\mathbf{\Sigma}$.

The *spectrum shift* transformation translates the whole spectrum of the matrix such that its minimal value is 0. That is, let ν be the minimal eigenvalue of K. The spectrum shift transformation maps each eigenvalue λ_i to $\lambda_i - \nu$. The resulting kernel matrix $\mathbf{K}_{\text{shift}}$ can be concisely expressed as

$$\mathbf{K}_{\text{shift}} = \mathbf{V}(\mathbf{\Sigma} - \nu\,\mathbf{I})\,\mathbf{V}^T, \tag{2.6}$$

where $\mathbf{I}$ is the identity matrix. Whilst this transformation does not change the similarity of any two distinct instances, it does increase all self-similarities in the kernel matrix.

The *spectrum square* transformation, which maps each eigenvalue to its square, is the most computationally efficient approach, since it does not require the eigendecomposition of the kernel matrix. To see this, observe that the resulting kernel matrix $\mathbf{K}_{\text{square}}$ can be expressed as

$$\mathbf{K}_{\text{square}} = \mathbf{V}\,\boldsymbol{\Sigma}^2\,\mathbf{V}^T = \mathbf{K}^2\,. \tag{2.7}$$

Hence, to compute spectrum square, it is sufficient to square the original kernel matrix.

Wu et al. (2005) performed a comparative analysis of spectrum clip, flip and shift over eight classification data sets, each equipped with an indefinite kernel. They observed that clip is to be preferred only when the ratio of the smallest to largest eigenvalue was close to zero. In the case that this ratio was large in magnitude, flip exhibited a superior performance. This observation provides evidence that negative eigenvalues of large absolute value may constitute more information than just noise. Both flip and shift worked best in five out of eight data sets, with shift being preferred in situations when the ratio of the smallest to largest eigenvalue was small in magnitude.

The spectrum transformation approaches represent a simple solution to the handling of indefinite kernels. However, as we have seen, the resulting performance of the approaches are dependent on the spectrum of the kernel

matrix. From the point of view of a practitioner, a blanket method to cover all bases would be to test each spectrum transformation and select the one that performs best. It would be preferred if one could automatically select the optimal transformation for a given problem. Chen et al. (2009) follow this line of reasoning and design an algorithm that simultaneously learns an SVM classifier and a spectrum transformation. More specifically, the authors find a vector $\boldsymbol{\alpha}$ such that the resulting kernel matrix $\mathbf{K}_{\text{learnt}}$, expressed as

$$\mathbf{K}_{\text{learnt}} = \mathbf{V}\,\boldsymbol{\Sigma}\,\text{diag}(\boldsymbol{\alpha})\,\mathbf{V}^T, \tag{2.8}$$

is positive semi-definite. It is clear to see that clip, flip, shift and square are all special cases of this more general spectrum transformation. In a comparison against clip, flip and shift, Chen et al. demonstrate that their method is among the top performing methods on five out of the six classification data sets. Moreover, on some data sets, the learnt spectrum modification achieves statistically significant improvements over the simple spectrum transformations.

Proxy Matrices

Similarly to spectrum transformations, the class of proxy matrix approaches produce a positive semi-definite matrix, the *proxy matrix*, from an indefinite input. However, instead of operating solely on the eigenvalues of the indefinite kernel matrix, proxy matrix approaches search for a positive semi-definite matrix that is close the original indefinite matrix. Moreover, this is

performed whilst simultaneously inferring a decision function. The method of Chen et al. (2009) for learning a spectrum transformation can be considered as a special case of a proxy matrix approach, in which the proxy matrix is endowed with a specific structure according to Equation (2.8). More generally, however, the only requirements of a proxy matrix is that it is positive semi-definite. This general approach is addressed by Luss and d'Aspremont (2007), in which a proxy matrix is sought that minimises the Frobenius norm between itself and the indefinite kernel matrix. Whilst their main focus is a proxy matrix SVM, algorithms are also developed for the problems of regression and anomaly detection. In a set of experiments across four classification data sets, Luss and d'Aspremont compare their approach to spectrum clip, flip and shift. It is observed that the proxy matrix approach performs best when the original indefinite kernel matrix has negative eigenvalues that are large in magnitude. We note, however, that no tests of significance were performed. Even so, the flexibility to find a proxy matrix for the problem at hand represents a substantial improvement over the fixed spectrum transformations.

Zhan et al. (2019) apply the anomaly detection algorithm of Luss and d'Aspremont for the task of virtual screening. More specifically, using a 3D alignment score as an indefinite kernel between chemical compounds, Zhan et al. (2019) aim to identify whether an input compound is a drug candidate for a given drug target. Comparing to spectrum clip, flip, shift and square across numerous data sets, they observe that the proxy matrix anomaly detection algorithm is the best performing method in the majority of cases.

2.2.2 Kreĭn Space Methods

As discussed in the previous section, transformation methods approach the problem of learning with indefinite kernels as one of learning with an appropriate positive-definite surrogate. While these methods facilitate the learning process, they still require the additional complexity of choosing an appropriate surrogate. This additional preprocessing step is inherent to their design since transformation methods aim to apply indefinite kernels to learning algorithms designed for positive-definite kernels. An arguably preferable approach is to design learning algorithms for indefinite kernels explicitly. This section discusses Kreĭn space methods, a set of learning algorithms that utilise the geometry of a Kreĭn space to incorporate an indefinite kernel directly. We qualitatively discuss the algorithms, leaving their theoretical description to Section 4.1.

Developing a learning algorithm incorporating an indefinite kernel function constitutes learning a function in a Kreĭn space. Whilst the theory of Kreĭn spaces has long been known (Bognár, 1974), their first application to machine learning is due to Ong et al. (2004). The authors recognise the importance of Kreĭn spaces and devise iterative algorithms to perform regression with indefinite kernels. Whilst they only present results on toy problems, their work represents a remarkable leap forward in indefinite kernel methods.

Haasdonk and Pekalska (2008) present Indefinite Kernel Fisher Discriminant (IKFD), an algorithm to perform classification with indefinite kernels.

Similarly to linear discriminant analysis, IKFD learns a classifier that maximises the between-class scatter matrix [3] whilst minimising the within-class scatter matrix. The authors show that the solution resembles that of Kernel Fisher Discriminant (Ghojogh et al., 2019), and hence can be solved efficiently. In a set of experiments on both toy and real-world data sets, Haasdonk and Pekalska demonstrate the superiority of IKFD in comparison to two other baseline indefinite kernel classifiers (Duin and Pekalska, 2005; Haasdonk, 2005). In another work, the same authors extend IKFD to one in which the class scatter matrices can vary between the classes (Pekalska and Haasdonk, 2008). The algorithm, known as Indefinite Kernel Quadratic Discriminant (IKQD), is compared to IKFD and two other baseline indefinite kernel classifiers on many binary and multi-class classification data sets. Whilst there are no clear winners between the two discriminant learners, both generally outperform the baseline classifiers.

When discussing kernel-based classifiers, the most well-known algorithm is the SVM. Loosli et al. (2015) present an algorithm, which we refer to as the Loosli Kreĭn-SVM (LK-SVM), that exploits the properties of a Kreĭn space to learn an SVM-like classifier with an indefinite kernel. A standard SVM is formulated as a convex optimisation problem (see Section 3.2.3 for more information) and, therefore, admits a globally optimal solution. On the other hand, the LK-SVM is formulated as a stabilisation problem, with its solution only guaranteed to be locally optimal. Furthermore, Loosli et al. show that their approach can equivalently be solved via a standard SVM using

[3] The scatter matrix is the covariance matrix estimated from a sample of data.

the spectrum flip kernel matrix followed by a linear transformation. Nevertheless, in a set of experiments comparing the LK-SVM to several indefinite SVM approaches, the LK-SVM performs favourably. Furthermore, the authors provide a detailed explanation of why one should and should not use their algorithm.

Whilst kernel methods are primarily known for their supervised learning algorithms, there do exist several unsupervised kernel methods, including, for instance, kernel k-means and kernel PCA (Dhillon et al., 2004; Schölkopf et al., 1997). Less attention has been placed on unsupervised indefinite kernel methods. However, kernel PCA has been successfully generalised to the indefinite kernel setting (Zafeiriou, 2012). Similarly to standard PCA, Zafeiriou develops an algorithm to compute the principal components of the data in a Kreĭn space. The projection of the data onto the principal components defines a nonlinear mapping that can reveal hidden structures. The algorithm is tested in a classification setting by coupling the nonlinear mapping with a simple nearest neighbour classifier, demonstrating highly accurate classifiers.

Another exciting line of work in Kreĭn space methods is the study of their application to large data sets. Positive definite and indefinite kernel methods require computation and storage of the kernel matrix. For a data set of size N, these operations grow as $O(N^2)$ in run time and memory. Furthermore, many kernel methods require solving a linear system involving the kernel matrix, an operation that, in its worst case, grows as $O(N^3)$ in run time. Hence, applying kernel methods to large data sets is infeasible in

their naive implementation. This fact has prompted researchers to consider approximate kernel methods that exhibit better scaling properties but retain the predictive performance of the exact method. The two most prominent approaches in this line of work are Nyström subsampling and random features (Bach, 2013; Rahimi and Recht, 2007), both of which have indefinite kernel extensions. Oglic and Gärtner (2019) derive the first-known technique for scaling up indefinite kernel methods, in which they provide a mathematically complete Nyström subsampling method. This derivation is accompanied by large-scale least squares and SVM-like algorithms for indefinite kernels. The results demonstrate not only the computational efficiency of Nyström subsampling but also the accuracy of the approximated model. Similarly, Liu et al. (2021) have devised a random feature approximation scheme for indefinite kernels. Once again, the results are promising, and their method performs comparably with Nyström subsampling.

The work of Oglic and Gärtner (2019) and, indeed, their earlier work (Oglic and Gärtner, 2018), tackles learning with indefinite kernels from a subtly different perspective. Oglic and Gärtner consider a different regularisation scheme to what is considered by Ong et al. (2004) and Loosli et al. (2015). The previous approaches considered regularisation directly in the Kreĭn space. However, this leads to non-convex optimisation problems that can exhibit exponentially many local optima. Thus, the problems being solved are one of stabilisation, as opposed to minimisation. The resulting models are not guaranteed to be globally optimal. It has been empirically observed that models resulting from solving the stabilisation problem fail to

generalise to unseen instances (Oglic and Gärtner, 2018). On the other hand, the regularisation scheme proposed by Oglic and Gärtner leads to optimisation problems that exhibit globally optimal solutions and good generalisation properties. When discussing the Kreĭn-SVM in Equation (4.8), we follow the regularisation scheme of Oglic and Gärtner.

Chapter 3

Machine Learning

This chapter covers the background mathematical theory of some fundamental elements of machine learning, which are required to understand the remaining chapters. Section 3.1 provides an overview of learning theory, centering the discussion on *supervised learning*. We discuss how to measure the quality of a prediction, as well as how the risk minimisation framework allows one to learn a predictive function. Section 3.2 presents algorithms to learn linear functions for the tasks of classification and regression, known as Support Vector Machines (SVMs). We further show how the theory of convex optimisation can be employed to reveal deeper insights into SVMs. In particular, how the resulting decision function depends only on a subset of instances known as *support vectors*, from which the algorithm gets its name. We conclude the chapter by showing how SVMs can be generalised to learn non-linear functions through the use of *kernels*. Section 3.3 provides the relevant theory of kernel functions and their associated reproducing kernel Hilbert spaces (RKHS). This theory is used to derive the SVM from the point of view of risk minimisation in a RKHS.

3.1 Learning Theory

3.1.1 Loss Functions

In the pursuit of solving a machine learning task and discovering an optimal prediction model, a loss function is generally employed to measure the suitability of a single model. In particular, a significant component of training a model corresponds to minimisation of a given loss function. In the context of supervised learning, in which one is trying to model a relationship between input-output pairs, the loss function should quantify the difference between a prediction and its corresponding true output.

Definition 3.1.1 (Loss Function)**.** Let $\mathcal{Y}$ be a label space. The non-negative function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is called a loss function if $\ell(y, y) = 0$ for all $y \in \mathcal{Y}$. (Schölkopf and Smola, 2002)

In particular, given a data example $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ and a hypothesis $f : \mathcal{X} \to \mathcal{Y}$, the loss function $\ell(y, f(\mathbf{x}))$ will output zero if and only if the observation y agrees with the prediction $f(\mathbf{x})$. Different output spaces constitute different loss functions and there exist many applicable loss functions for a given space. In the following, we describe some common examples when solving a binary classification or regression problem.

Binary Classification

In the setting of binary classification, we describe the output space $\mathcal{Y}$ as the set $\{-1, +1\}$. For our purposes, let the hypothesis $f : \mathcal{X} \to \mathbb{R}$ be a real-valued function such that $\text{sign}(f(\mathbf{x}))$ is the class prediction for instance x.

The simplest way to measure the loss of a prediction $f(\mathbf{x})$ is to incur a value of 1 for an incorrect prediction; otherwise, there is no penalty. This is known as the 0-1 loss function and is given in Definition 3.1.2

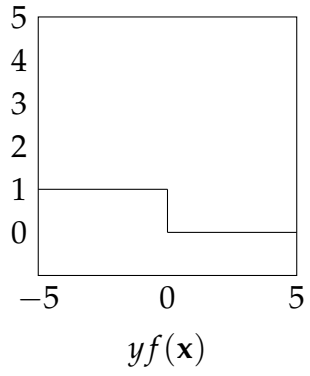
Definition 3.1.2 (0-1 Loss)**.** Let $\mathcal{Y} = \{-1, +1\}$ be the binary classification label space, $\mathcal{X}$ be an instance space, $f : \mathcal{X} \to \mathbb{R}$ and $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. The 0-1 loss function $\ell_{0\text{-}1} : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}^+$ is defined as

$$\ell_{0\text{-}1}(y, f(\mathbf{x})) = \begin{cases} 0, & \text{if } y = \text{sign}(f(\mathbf{x})), \\ 1, & \text{otherwise.} \end{cases}$$

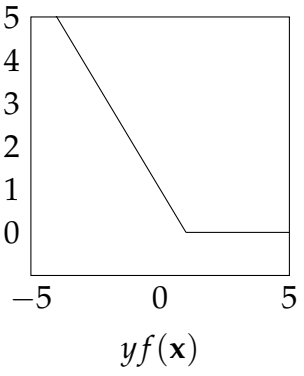
Whilst arguably the most intuitive, the 0-1 loss is difficult to use in practice since it is non-convex (see Definition 3.2.3) and discontinuous. It also ignores any notion of confidence from a prediction. For instance, by defining the absolute value $|f(\mathbf{x})|$ as the confidence of a prediction, one can apply greater penalisation to predictions which are *confidently incorrect*. The hinge loss, presented in Definition 3.1.3 and utilised throughout the thesis, is a notable example. Figure 3.1 depicts both the 0-1 loss and hinge loss.

Definition 3.1.3 (Hinge Loss)**.** Let $\mathcal{Y} = \{-1, +1\}$ be the binary classification label space, $\mathcal{X}$ be an instance space, $f : \mathcal{X} \to \mathcal{Y}$ and $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. The Hinge loss function $\ell_{\text{hinge}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is defined as

$$\ell_{\text{hinge}}(y, f(\mathbf{x})) = \max(0, 1 - yf(\mathbf{x})) = \begin{cases} 0, & \text{if } yf(\mathbf{x}) \geq 1, \\ 1 - yf(\mathbf{x}), & \text{otherwise.} \end{cases}$$



(A) 0-1 Loss



(B) Hinge Loss

FIGURE 3.1: Classification loss functions.

Regression

When performing regression, our output space is defined as the real numbers $\mathcal{Y} = \mathbb{R}$. A loss function in this case should quantify the *closeness* of a prediction. A very common example would be the squared loss, given in Definition 3.1.4 as the squared distance between a prediction and its true value.

Definition 3.1.4 (Squared Loss)**.** Let $\mathcal{Y} = \mathbb{R}$ be the regression label space, $\mathcal{X}$ be an instance space, $f : \mathcal{X} \to \mathcal{Y}$ and $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. The squared loss function $\ell_2 : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is defined as

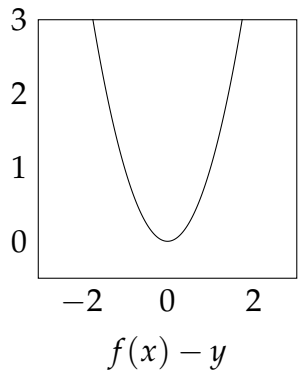
$$\ell_2(y, f(\mathbf{x})) = (f(\mathbf{x}) - y)^2.$$

Whilst being widely applicable, the squared loss is non-negative for any prediction which does not exactly equal its true value. In some cases, one may be satisfied if the prediction is within a given tolerance. As seen in Definition 3.1.5, the ε-insensitive loss function is equal to zero for all predictions

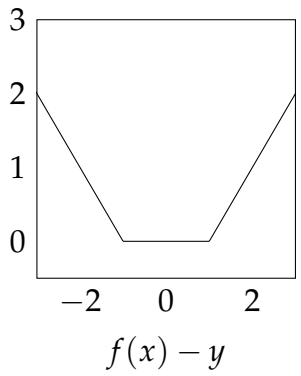
that are a within a distance of ε of their true value. Examples of both the aforementioned loss functions are given in Figure 3.2.

Definition 3.1.5 (ε-insensitive Loss)**.** Let $\mathcal{Y} = \mathbb{R}$ be the regression label space, $\mathcal{X}$ be an instance space, $f : \mathcal{X} \to \mathcal{Y}$, $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ and $\varepsilon > 0$. The ε-insensitive loss function $\ell_\varepsilon : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is defined as

$$\ell_\varepsilon(y, f(\mathbf{x})) = \max(0, |f(\mathbf{x}) - y| - \varepsilon)$$



(A) Squared Loss



(B) ε-insensitive Loss

FIGURE 3.2: Regression loss functions with $\varepsilon = 1$.

3.1.2 Empirical Risk Minimisation

Having defined the notion of a loss function, an approach to selecting a predictive model is to choose one that minimises a given loss function. Suppose that data examples $(\mathbf{x}, y)$ belong to the product space $\mathcal{X} \times \mathcal{Y}$ and are generated according to the joint distribution $\mathbb{P}(\mathbf{x}, y)$. As noted in previous works

(Vapnik, 1999; Schölkopf and Smola, 2002), one can aim to minimise the *expected risk*, defined as

$$R(f) = \mathbb{E}_{\mathbb{P}}\left[\ell(y, f(\mathbf{x}))\right] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, f(\mathbf{x})) \quad d\mathbb{P}(\mathbf{x}, y). \tag{3.1}$$

The minimiser of Equation (3.1) would yield the ideal predictive model. However, it is assumed that the underlying probability distribution $\mathbb{P}$ is unknown. Instead, one can minimise an estimate of $\mathbb{E}_{\mathbb{P}}$ from a sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, such that each $(\mathbf{x}_i, y_i) \sim \mathbb{P}$. The *empirical risk*, defined below, estimates the expected risk for the sample S

$$R_{\text{emp}}(f) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i)). \tag{3.2}$$

So far, we have made no reference to the hypothesis $f : \mathcal{X} \to \mathcal{Y}$. In practice, one often restricts it to belong to a certain class of functions $\mathcal{H}$, commonly referred to as the *hypothesis space*. The objective is to determine the best function of the hypothesis space with respect to the empirical risk, which gives rise to the principle of *empirical risk minimisation*.

Definition 3.1.6 (Empirical Risk Minimisation, Schölkopf and Smola 2002)**.** Let $\mathcal{H}$ be a space of functions mapping from $\mathcal{X}$ to $\mathcal{Y}$ and ℓ be a loss function. The optimisation problem

$$\underset{f\in\mathcal{H}}{\text{minimise}} \; R_{\text{emp}}(f) \tag{3.3}$$

is called *empirical risk minimisation* (ERM).

It is possible that many candidate functions will minimise the empirical risk. Of these functions, some may *overfit* the training set S. By this, we mean a function that describes well the samples in S but fails to capture the true functional relationship between the inputs and outputs. These effects can be mitigated by imposing additional constraints on the functions in the hypothesis space, commonly referred to as *regularisation*. A notable instance of regularisation is incorporation of the function norm into the objective function. The norm can be considered as a measure of complexity of a function. Including it into the objective function allows one to not only select a function which minimises the empirical risk, but to also select the simplest example (with respect to the norm) of such a function.

Definition 3.1.7 (Regularised Risk Minimisation)**.** Let $\mathcal{H}$ be a hypothesis space and $g : \mathbb{R} \to \mathbb{R}$ be a strictly monotonically increasing function. The functional

$$R_{\text{reg}}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i)) + g(\|f\|_{\mathcal{H}}), \quad f \in \mathcal{H} \tag{3.4}$$

is called the *regularised empirical risk* and its minimisation with respect to f is called regularised risk minimisation (RRM).

The regularised risk functional R_{reg} is a fundamental concept in various machine learning algorithms, where the choice of loss function ℓ and regularisation term $g(\|f\|_{\mathcal{H}})$ can differ depending on the approach. All machine learning algorithms discussed in the following chapters adhere to the RRM principle.

3.2 Support Vector Learning

In the present section, we describe a class of algorithms known as Support Vector Machines (SVMs). These are discriminative learning algorithms, whose output is a hyperplane in an inner product space. For the time being, we will focus our discussion on the space $\mathbb{R}^n$.

3.2.1 Hyperplane Classifiers

Consider the problem of binary classification in $\mathbb{R}^n$: we are given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \{-1, +1\}$. We suppose that our training set is *linearly separable*, meaning there exists some hyperplane which separates the sets $\mathcal{Y}_+ = \{\mathbf{x}_i \,|y_i = +1\}$ and $\mathcal{Y}_- = \{\mathbf{x}_i \,|y_i = -1\}$. A simple learning algorithm for this scenario would be to find a separating hyperplane. Classification is then performed based on which side of the hyperplane an instance resides. However, it is often the case that if one such hyperplane exists, then many more also exist. Ideally we would like to select the optimal hyperplane with respect to some quantity. To do so, we introduce the concept of a margin.

Margin of a Hyperplane

A hyperplane $\mathcal{P}$, parameterised by $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, can be described as the set of points such that

$$\mathcal{P} = \{\mathbf{x} \,|\langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}. \tag{3.5}$$

The vector $\mathbf{w}$ is always normal the hyperplane, Figure 3.3 provides an example in $\mathbb{R}^2$. The margin of $\mathcal{P}$ is defined as the minimum distance from the

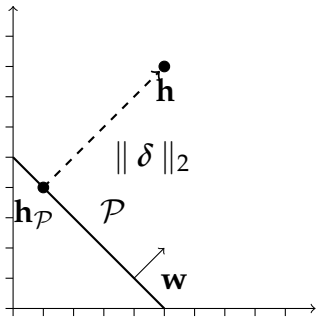


FIGURE 3.3: An example of a hyperplane $\mathcal{P}$ in $\mathbb{R}^2$ and an arbitrary point $\mathbf{h}$. The normal vector to the hyperplane is shown, denoted by $\mathbf{w}$. The orthogonal projection of $\mathbf{h}$ onto $\mathcal{P}$ is shown, denoted by $\mathbf{h}_{\mathcal{P}}$. The distance between $\mathbf{h}$ and $\mathbf{h}_{\mathcal{P}}$ is denoted by $\| \delta \|_2$.

hyperplane to the training instances. To compute the distance, consider an arbitrary point $\mathbf{h} \in \mathbb{R}^n$ and let $\delta \in \mathbb{R}^n$ be the vector from $\mathcal{P}$ to $\mathbf{h}$ such that δ is orthogonal to $\mathcal{P}$. The vector $\mathbf{h}^{\mathcal{P}} = \mathbf{h} - \delta$ is the projection of $\mathbf{h}$ onto $\mathcal{P}$. Figure 3.3 depicts this scenario. Since $\mathbf{h}^{\mathcal{P}} \in \mathcal{P}$, we have that

$$\langle \mathbf{w}, \mathbf{h}^{\mathcal{P}} \rangle + b = \langle \mathbf{w}, (\mathbf{h} - \delta) \rangle + b = 0. \tag{3.6}$$

Furthermore, δ is parallel to $\mathbf{w}$ so we can express $\delta = \alpha \, \mathbf{w}$ for some $\alpha \in \mathbb{R}$. Hence

$$\langle \mathbf{w}, (\mathbf{h} - \alpha \, \mathbf{w}) \rangle + b = 0, \tag{3.7}$$

which implies that

$$\alpha = \frac{\langle \mathbf{w}, \mathbf{h} \rangle + b}{\| \mathbf{w} \|_2^2}. \tag{3.8}$$

The length of δ, which is the orthogonal distance between $\mathbf{h}$ and $\mathcal{P}$, can therefore be expressed as

$$\| \delta \|_2 = \frac{|\langle \mathbf{w}, \mathbf{h} \rangle + b|}{\| \mathbf{w} \|_2}. \tag{3.9}$$

With this in mind, Definition 3.2.1 provides a definition of the margin of a hyperplane.

Definition 3.2.1. Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \{-1, +1\}$ be a linearly separable data set. Furthermore, for $\mathbf{h} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, let $\mathcal{P} = \{\mathbf{x} \,|\, \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$ be a separating hyperplane of S. The margin $\gamma(\mathbf{w}, b)$ of $\mathcal{P}$ with respect to S is defined as

$$\gamma(\mathbf{w}, b) = \underset{\mathbf{x} \in S}{\text{minimise}} \frac{|\langle \mathbf{w}, \mathbf{x} \rangle + b|}{\| \mathbf{w} \|_2}. \tag{3.10}$$

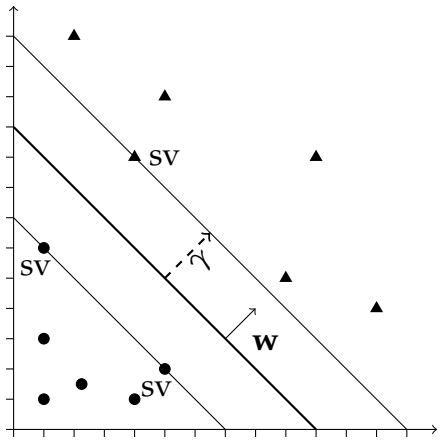


FIGURE 3.4: An example of a maximum-margin hyperplane separating the circles from triangles. The normal vector to the hyperplane is shown, denoted by $\mathbf{w}$. The support vectors are shown, denoted by SV. The margin of the hyperplane is shown and denoted by γ.

Hard Margin SVM

Going back to our example, we wish to select the separating hyperplane that maximises the margin. Requiring that the hyperplane separates the classes is to require that all instances of the positive class reside on one side of the hyperplane and all instances of the negative class reside on the other side. For an instance of the positive class $\mathbf{x}_p$, note that

$$\langle \mathbf{w}, \mathbf{x}_p \rangle + b \geq 0. \tag{3.11}$$

Similarly, for an instance of the negative class $\mathbf{x}_n$, we have

$$\langle \mathbf{w}, \mathbf{x}_n \rangle + b \leq 0. \tag{3.12}$$

Hence, for a separating hyperplane, we have

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 0, \quad \forall i. \tag{3.13}$$

The problem of finding a maximum-margin separating hyperplane for the data set S can be expressed as

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{arg maximise}} \quad \gamma(\mathbf{w}, b) \\ &\text{subject to} \qquad\quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 0 \quad \forall (\mathbf{x}_i, y_i) \in S. \end{aligned} \tag{3.14}$$

In order to solve Equation (3.14), we first note that the margin of a hyperplane is scale invariant so that $\gamma(\beta\,\mathbf{w}, \beta b) = \gamma(\mathbf{w}, b)$ for $\beta \in \mathbb{R}$. Hence, we can

choose $\mathbf{w}, b$ such that

$$\underset{\mathbf{x} \in S}{\text{minimise}} \, |\langle \mathbf{w}, \mathbf{x} \rangle + b| = 1. \tag{3.15}$$

Equation (3.14) can therefore be expressed as

$$\begin{aligned} \underset{\mathbf{w}, b}{\text{maximise}} \quad & \frac{1}{\| \mathbf{w} \|_2} \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 0 \quad \forall (\mathbf{x}_i, y_i) \in S, \\ & \underset{\mathbf{x} \in S}{\text{minimise}} \, |\langle \mathbf{w}, \mathbf{x} \rangle + b| = 1. \end{aligned}$$

The second constraint implies that

$$|\langle \mathbf{w}, x \rangle + b| \geq 1 \tag{3.16}$$

for all $\mathbf{x} \in S$. Hence,

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \forall (\mathbf{x}_i, y_i) \in S. \tag{3.17}$$

Noting that

$$\underset{\mathbf{w}}{\text{maximise}} \, \frac{1}{\| \mathbf{w} \|_2} = \underset{\mathbf{w}}{\text{minimise}} \, \| \mathbf{w} \|_2 = \underset{\mathbf{w}}{\text{minimise}} \, \| \mathbf{w} \|_2^2, \tag{3.18}$$

we can formulate Equation (3.14) as

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{minimise}} \quad \| \mathbf{w} \|_2^2 \\ &\text{subject to} \quad y_i(\langle \mathbf{w}, x_i \rangle + b) \geq 1 \quad \forall (\mathbf{x}_i, y_i) \in S. \end{aligned} \tag{3.19}$$

For a given solution $(\mathbf{w}^\star, b^\star)$ to Equation (3.19), the resulting decision function $f : \mathbb{R}^n \to \{-1, +1\}$ is expressed as

$$f(x) = \text{sign}(\langle \mathbf{w}^\star, \mathbf{x} \rangle + b^\star). \tag{3.20}$$

It will always be the case that some of the instances in S will satisfy the constraints with equality e.g.

$$y_i(\langle \mathbf{w}^\star, \mathbf{x}_i \rangle + b^\star) = 1. \tag{3.21}$$

These instances define the margin of the hyperplane and therefore the resulting hyperplane itself. They are known as *support vectors* and the problem being solved in Equation (3.19) is known as the *hard-margin SVM*. Figure 3.4 illustrates the maximum-margin separating hyperplane, as well as the support vectors, for a toy problem.

Soft Margin SVM

The derivations of the previous section are only applicable when the data set exhibits linear separability, which is often too optimistic of an assumption. When this is not true, we can allow some instances to violate the constraints

of Equation (3.19). In particular, we introduce slack variables

$$\xi_i \geq 0 \quad , i = 1, \ldots, m, \tag{3.22}$$

and modify the constraints to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m. \tag{3.23}$$

For each pair $(\mathbf{x}_i, y_i)$, one can make ξ_i arbitrarily large such that the constraint is always satisfied. In order to avoid this trivial solution, a penalisation term is added to the objective function of Equation (3.19). This leads to the following optimisation problem, known as the *soft margin SVM*,

$$\begin{aligned} \underset{\mathbf{w},b,\xi}{\text{minimise}} \quad & \| \mathbf{w} \|_2^2 + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m, \\ & \xi_i \geq 0, \qquad\qquad\qquad\quad i = 1, \ldots, m, \end{aligned} \tag{3.24}$$

where $C > 0$. Whenever $\xi_i > 0$, a *margin error* occurs in which instance $\mathbf{x}_i$ does not lie on the correct side of the margin. In this case, the value ξ_i represents the perpendicular distance to the margin. The constant C is an *a priori* chosen parameter allowing one to balance the total margin errors with the complexity of the resulting hyperplane.

We note that the constraints of Equation (3.24) can be equivalently written as

$$\xi_i = \begin{cases} 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b), & \text{if} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) < 1, \\ 0, & \text{if} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \end{cases} \qquad i = 1, \ldots, m. \tag{3.25}$$

This can be compactly expressed as

$$\xi_i = \ell_{\text{hinge}}(y_i, \langle \mathbf{w}, \mathbf{x}_i \rangle + b), \tag{3.26}$$

where ℓ_{hinge} is the hinge loss (see Definition 3.1.3). This leads to the following unconstrained formulation of the soft margin SVM, which is an example of RRM.

$$\underset{\mathbf{w},b}{\text{minimise}} \quad \| \mathbf{w} \|_2^2 + C \sum_{i=1}^{m} \ell_{\text{hinge}}(y_i, \langle \mathbf{w}, \mathbf{x}_i \rangle + b). \tag{3.27}$$

3.2.2 Epsilon Tubes

The intuitive idea of learning a maximum-margin separating hyperplane can be extended to the case of regression. In this setting, we are given a training data set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$ and would like to estimate a function $f : \mathbb{R}^n \to \mathbb{R}$ such that

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b. \tag{3.28}$$

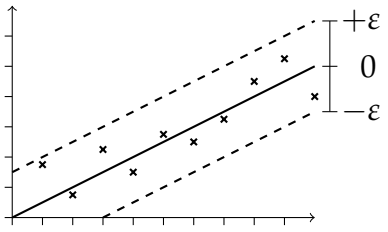


FIGURE 3.5: An example of a set of points and a tube with radius ε enclosing them.

This is the standard problem of linear regression. However, we further impose that the predictions have at most $\varepsilon > 0$ deviation from the true observations. Formally, we require that

$$|f(\mathbf{x}_i) - y_i| \leq \varepsilon, \quad \forall (\mathbf{x}_i, y_i) \in S. \tag{3.29}$$

It may be the case that multiple functions satisfy this property. In order to decide between them, we choose the one with minimum norm. This leads to the problem presented in Equation (3.30), which is known as *support vector regression* (SVR). The result of which is a tube of radius ε fitted to the data, known as an *epsilon tube*.

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimise}} \quad & \| \mathbf{w} \|_2^2 \\ \text{subject to} \quad & \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon \quad \forall (\mathbf{x}_i, y_i) \in S, \\ & y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \varepsilon \quad \forall (\mathbf{x}_i, y_i) \in S. \end{aligned} \tag{3.30}$$

An example of a hyperplane and corresponding epsilon tube is given in Figure 3.5. Since ε is fixed, it can be difficult to choose its value *a priori*. In practice, the value which minimises generalisation error on a validation set

is usually chosen.[1] There will clearly be a value of epsilon such that all the constraints of Equation (3.30) are satisfied. However, due to the presence of outliers and other non-regularities in the training data set, this value may need to be arbitrarily large. In a similar vein to allow some instances to lie on the incorrect side of the margin in the soft margin SVM, we can allow some instances to lie outside the epsilon tube. In this case, we introduce two sets of slack variables

$$\boldsymbol{\xi}_i \geq 0,$$

$$\boldsymbol{\xi}_i^* \geq 0,$$

and modify the constraints of Equation (3.30) to

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon + \boldsymbol{\xi}_i \quad i = 1, \ldots, m,$$

$$y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \varepsilon + \boldsymbol{\xi}_i^* \quad i = 1, \ldots, m.$$

This leads to the more familiar form of the SVR algorithm, presented below

$$\begin{aligned} \underset{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*}{\text{minimise}} \quad & \| \mathbf{w} \|_2^2 + C \sum_{i=1}^{m} (\boldsymbol{\xi}_i + \boldsymbol{\xi}_i^*) \\ \text{subject to} \quad & \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon + \boldsymbol{\xi}_i \quad i = 1, \ldots, m, \\ & y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \varepsilon + \boldsymbol{\xi}_i^* \quad i = 1, \ldots, m, \\ & \boldsymbol{\xi}_i, \boldsymbol{\xi}_i^* \geq 0, \end{aligned} \tag{3.31}$$

[1] It is important to note that generalisation error should not be measured with the $\varepsilon-$insensitive loss. Taking $\varepsilon \to \infty$ would lead to zero loss on all instances in the validation set. An appropriate measure of generalisation error in this case would be the squared loss.

where $C > 0$. Those instances with $\xi_i > 0$ or $\xi_i^* > 0$ lie outside the epsilon tube, with their distance to the tube boundary being ξ_i or ξ_i^*, respectively. The constant C allows one to balance the total deviation with the complexity of the resulting hyperplane. Equation (3.31) can be equivalently expressed in the form of RRM by noting that if $\xi_i > 0$, then it must the case that $\xi_i^* = 0$ and vice versa. Hence each ξ_i can be expressed as

$$\xi_i = \begin{cases} \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i - \varepsilon, & \text{if} \quad \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i > \varepsilon, \\ 0, & \text{if} \quad \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon, \end{cases} \quad i = 1, \ldots, m. \tag{3.32}$$

Similarly, each ξ_i^* can be expressed as

$$\xi_i^* = \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b - \varepsilon, & \text{if} \quad y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b > \varepsilon, \\ 0, & \text{if} \quad y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \varepsilon, \end{cases} \quad i = 1, \ldots, m, \tag{3.33}$$

so that their sum can be written as

$$\xi_i + \xi_i^* = \begin{cases} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i| - \varepsilon, & \text{if} \quad |\langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i| > \varepsilon, \\ 0, & \text{if} \quad |\langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i| \leq \varepsilon, \end{cases} \quad i = 1, \ldots, m. \tag{3.34}$$

This is equivalent to

$$\xi_i + \xi_i^* = \ell_\varepsilon(y_i, \langle \mathbf{w}, \mathbf{x}_i \rangle + b), \quad i = 1, \ldots, m, \tag{3.35}$$

where ℓ_ε is the epsilon-insensitive loss (see Definition 3.1.5). Therefore, the SVR algorithm can be equivalently formulated as

$$\underset{\mathbf{w},b}{\text{minimise}} \quad \| \mathbf{w} \|_2^2 + C \sum_{i=1}^{m} \ell_\varepsilon(y_i, \langle \mathbf{w}, \mathbf{x}_i \rangle + b), \tag{3.36}$$

which is in the form of RRM.

3.2.3 Convex Optimisation

The preceding sections derived SVM algorithms for the tasks of classification and regression. The problems presented in Equations (3.24) and (3.31) are known as *convex optimisation* problems. These are optimisation problems in which the objective function and the feasible set are convex. Convex functions have the attractive property that any local minimiser of the function is also a global minimiser. Furthermore, there are very efficient methods to solve convex optimisation problems. This section describes some useful results related to such problems, which we will later use to our advantage in solving Equations (3.24) and (3.31). Our description follows that of Cristianini, Shawe-Taylor, et al. (2000) and Boyd and Vandenberghe (2004). Initially, we define the notion of a convex set.

Definition 3.2.2 (Convex Set, Boyd and Vandenberghe 2004)**.** A set C is convex if the line segment between any two points in C lies in C, i.e., if for any $\mathbf{x}_1, \mathbf{x}_2 \in C$ and any θ with $0 \leq \theta \leq 1$, we have

$$\theta \, \mathbf{x}_1 + (1 - \theta) \, \mathbf{x}_2 \in C. \tag{3.37}$$

As shown in Definition 3.2.2, a set is convex if the line drawn between any two elements in the set lies completely in the set. Similarly, we call a function convex if the line drawn between any two points on the function lies above the graph of the function. We formally define this notion in Definition 3.2.3.

Definition 3.2.3 (Convex Function, Cristianini, Shawe-Taylor, et al. 2000)**.** A function $f : \mathbb{R}^d \to \mathbb{R}$ is *convex* if for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$,

$$f(\tau\,\mathbf{x}_1 + (1-\tau)\,\mathbf{x}_2) \leq \tau f(\mathbf{x}_1) + (1-\tau) f(\mathbf{x}_2)$$

holds true for all $\tau \in (0,1)$. In the case of a strict inequality, the function is called *strictly convex*.

In order to define a convex optimisation problem, we also need the definition of an *affine* function.

Definition 3.2.4 (Affine Function, Cristianini, Shawe-Taylor, et al. 2000)**.** Let $A \in \mathbb{R}^{n\times m}$ be a real-valued matrix and $b \in \mathbb{R}^n$ a vector. A function $f : \mathbb{R}^m \to \mathbb{R}^n$ with $f(\mathbf{x}) = A\,\mathbf{x} + b$ is said to be *affine*.

We now present the definition of a convex optimisation problem

Definition 3.2.5 (Convex Optimistion Problem, Cristianini, Shawe-Taylor, et al. 2000)**.** Given a convex set $\Omega \subset \mathbb{R}^n$, convex functions $f : \Omega \to \mathbb{R}$, $g_i : \Omega \to \mathbb{R}, i = 1,\ldots,k$, and affine functions $h_i : \Omega \to \mathbb{R}, i = 1,\ldots,m$, the

minimisation

$$\begin{aligned} \text{minimise} \quad & f(\mathbf{x}), && \mathbf{x} \in \Omega, \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, && i = 1, \ldots, k, \\ & h_i(\mathbf{x}) = 0, && i = 1, \ldots, m, \end{aligned} \tag{3.38}$$

is a *convex optimisation problem*.

The function f is called the *objective function*, the g_i are *inequality constraints* and the h_i are *equality constraints*. For problems in which f is differentiable and the $g_i, h_i = 0$, a necessary and sufficient condition for $\mathbf{x}^\star \in \Omega$ to solve Definition 3.2.5 is

$$\nabla f(\mathbf{x}^\star) = 0. \tag{3.39}$$

The concept of *duality* can help in solving the most general form of Definition 3.2.5. However, we must first introduce the *Lagrangian*. For a set $\Omega \in \mathbb{R}^n$ and functions $f : \Omega \to \mathbb{R}$, $g_i : \Omega \to \mathbb{R}, i = 1, \ldots, k$, and $h_i : \Omega \to \mathbb{R}, i = 1, \ldots, m$, consider an optimisation of the form

$$\begin{aligned} \text{minimise} \quad & f(\mathbf{x}), && \mathbf{x} \in \Omega, \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, && i = 1, \ldots, k, \\ & h_i(\mathbf{x}) = 0, && i = 1, \ldots, m,. \end{aligned} \tag{3.40}$$

The Lagrangian $L : \mathbb{R}^n \times \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}$ associated with Equation (3.40) is defined as

$$L(\mathbf{x}, \alpha, \nu) = f(\mathbf{x}) + \sum_{i=1}^{k} \boldsymbol{\alpha}_i \, g_i(\mathbf{x}) + \sum_{i=1}^{m} \boldsymbol{\nu}_i \, h_i(\mathbf{x})$$

Here, the numbers $\boldsymbol{\alpha}_i$ for $i = 1, \ldots, k$, and $\boldsymbol{\nu}_i$ for $i = 1, \ldots, m$, are known as the *Lagrangian multipliers*. Note that there is no requirement for the optimisation problem to be convex. Indeed, the Lagrangian is defined for constrained, non-convex optimisation problems. The basic idea of the Lagrangian is to account for the constraints with a new objective function as a weighted sum of the constraint functions. Given the Lagrangian function, we can define the *Lagrangian dual function* as the infimum of the Lagrangian over $\mathbf{x}$:

$$\theta(\boldsymbol{\alpha}, \boldsymbol{\nu}) = \inf_{\mathbf{x} \in \Omega} L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\nu}) = \inf_{\mathbf{x} \in \Omega} \left(f(\mathbf{x}) + \sum_{i=1}^{k} \boldsymbol{\alpha}_i \, g_i(\mathbf{x}) + \sum_{i=1}^{m} \boldsymbol{\nu} \, h_i(x) \right). \tag{3.41}$$

In the context of the Lagrangian dual function, the problem presented in Equation (3.40) is known as the *primal* problem. It can be shown that, if $\boldsymbol{\alpha}_i \geq 0$ for $i = 1, \ldots, k$, then the optimal value $f(\mathbf{x}^\star)$ of the primal problem is bounded below by the dual function $\theta(\boldsymbol{\alpha}, \boldsymbol{\nu})$ (Boyd and Vandenberghe, 2004). Hence, a strategy to approximately solve the primal problem is to maximise the dual function and obtain the best lower bound possible.

Definition 3.2.6 (Dual Problem, Boyd and Vandenberghe 2004)**.** Assume the primal problem is given by Equation (3.40). The maximisation problem

$$\begin{aligned} &\underset{\boldsymbol{\alpha},\boldsymbol{\nu}}{\text{maximise}} \quad \theta(\boldsymbol{\alpha},\boldsymbol{\nu}), \\ &\text{subject to} \quad \boldsymbol{\alpha}_i \geq 0, \quad i = 1,\ldots,k \end{aligned}$$

is called the *dual problem*. The difference between the optimal value of the primal problem and that of the dual problem, $f(\mathbf{x}^\star) - \theta(\boldsymbol{\alpha}^\star,\boldsymbol{\nu}^\star)$ is called the *duality gap*.

The dual problem is a convex optimisation problem, regardless of whether or not the primal problem is convex. The lower bound property of the dual function ensures the duality gap is non-negative. This is referred to as *weak duality*. *Strong duality* is the case when the duality gap is equal to zero. If strong duality holds, the considered optimisation problem can be equivalently solved in its dual formulation. As shown in Definition 3.2.7, Slater's condition is an example in which strong duality is guaranteed.

Definition 3.2.7 (Slater's Condition, Boyd and Vandenberghe 2004)**.** Consider a convex optimisation problem of the form of Definition 3.2.5. If there

exists a point $\mathbf{x}^\star \in \text{relint}(\Omega)$[2] such that

$$g_i(\mathbf{x}^\star) < 0 \quad i = 1, \ldots, k,$$

$$h_i(\mathbf{x}^\star) = 0 \quad i = 1, \ldots, m,$$

then strong duality holds.

Whilst Slater's condition allows one to equivalently solve the dual formulation, it provides no insight into what that solution will be. For the case of convex optimisation with differentiable objective and constraint functions, the KKT conditions provide a set of necessary and sufficient conditions that the optimal point must satisfy. These are presented in Definition 3.2.8.

Definition 3.2.8 (KKT Conditions, Boyd and Vandenberghe 2004)**.** Consider a convex optimisation problem of the form of Definition 3.2.5 in which the objective function, the inequality constraints and the equality constraints are differentiable. Let $L : \mathbb{R}^n \times \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}$ be the Lagrangian of the problem

$$L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\nu}) = f(\mathbf{x}) + \sum_{i=1}^{k} \boldsymbol{\alpha}_i \, g_i(\mathbf{x}) + \sum_{i=1}^{m} \boldsymbol{\nu}_i \, h_i(\mathbf{x}).$$

Suppose strong duality is satisfied and let $\mathbf{x}^\star \in \Omega$, $\boldsymbol{\alpha}^\star \in \mathbb{R}^k$, $\boldsymbol{\nu}^\star \in \mathbb{R}^m$ be the points at which it is satisfied. Then the following conditions, which are

[2] By $\mathbf{x}^\star \in \text{relint}(\Omega)$, we mean that $\mathbf{x}^\star$ belongs to the relative interior of Ω. For a precise definition, please see Boyd and Vandenberghe, 2004.

known as the *KKT conditions*, hold:

$$
\begin{aligned}
g_i(\mathbf{x}^\star) &\leq 0, \quad i = 1, \ldots, k, && \text{primal feasibility,} \\
h_i(\mathbf{x}^\star) &= 0, \quad i = 1, \ldots, m, && \text{primal feasibility,} \\
\boldsymbol{\alpha}_i^\star &\geq 0, \quad i = 1, \ldots, k, && \text{dual feasibility,} \\
\boldsymbol{\alpha}_i^\star \, g_i(\mathbf{x}^\star) &= 0, \quad i = 1, \ldots, k, && \text{complementary slackness,} \\
\frac{\partial L(\mathbf{x}^\star, \boldsymbol{\alpha}^\star, \boldsymbol{\nu}^\star)}{\partial \, \mathbf{x}} &= 0, && \text{zero derivative.}
\end{aligned}
$$

3.2.4 Dual Formulations

The previous section has highlighted a number of useful results for the theory of convex optimisation. In this section, we use the aforementioned results to present the dual formulations of Equations (3.24) and (3.31).

SVM

Before deriving the dual, it is useful to check whether strong duality holds. The problem presented in Equation (3.24) is a constrained optimisation problem over $\mathbb{R}^N$ for some N. Given that $\text{relint}(\mathbb{R}^N) = \mathbb{R}^N$, checking Slater's condition is equivalent to finding a point $\tilde{\mathbf{w}}, \tilde{b}, \tilde{\boldsymbol{\xi}}$ such that

$$
\begin{aligned}
y_i(\langle \tilde{\mathbf{w}}, \mathbf{x}_i \rangle + \tilde{b}) &> 1 - \tilde{\boldsymbol{\xi}}_i, \quad i = 1, \ldots, m, \\
\tilde{\boldsymbol{\xi}}_i &> 0, \qquad\quad i = 1, \ldots, m,
\end{aligned}
$$

Selecting each $\tilde{\boldsymbol{\xi}}_i > 0$ such that

$$\tilde{\boldsymbol{\xi}}_i > 1 - y_i(\langle \tilde{\mathbf{w}}, \mathbf{x}_i \rangle + \tilde{b}) \quad i = 1, \ldots, m, \tag{3.42}$$

for all $\tilde{\mathbf{w}}, \tilde{b}$ clearly satisfies the constraints. Hence, strong duality holds and Equation (3.24) can be equivalently solved in the dual. Furthermore, the problem is differentiable, implying that the KKT conditions can be applied. Recall from the previous section that in order to define the dual problem, one must first form the Lagrangian. For the soft margin SVM, this is given by

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \quad \| \mathbf{w} \|_2^2 + C \sum_{i=1}^{m} \boldsymbol{\xi}_i + \sum_{i=1}^{m} \boldsymbol{\alpha}_i (1 - \boldsymbol{\xi}_i - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)) - \sum_{i=1}^{m} \boldsymbol{\beta}_i \boldsymbol{\xi}_i \,. \tag{3.43}$$

The zero derivative property of the KKT conditions implies that the optimal $\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star$ solving Equation (3.24) must satisfy

$$\begin{aligned} \frac{\partial L(\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \mathbf{w}} &= 0, \\ \frac{\partial L(\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial b} &= 0, \\ \frac{\partial L(\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \boldsymbol{\xi}} &= 0. \end{aligned}$$

Hence

$$\begin{aligned} \mathbf{w}^\star &= \frac{1}{2}\sum_{i=1}^{m} \boldsymbol{\alpha}_i \, y_i \, \mathbf{x}_i, \\ 0 &= \sum_{i=1}^{m} y_i \, \boldsymbol{\alpha}_i, \\ 0 &= C - \boldsymbol{\alpha}_i - \boldsymbol{\beta}_i, \quad i = 1, \ldots, m. \end{aligned} \tag{3.44}$$

Substituting these expressions into the Lagrangian gives the equivalent dual problem

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximise}} \quad & \sum_{i=1}^{m} \boldsymbol{\alpha}_i - \frac{1}{4}\sum_{i,j=1}^{m} \boldsymbol{\alpha}_i \, \boldsymbol{\alpha}_j \, y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i \, \boldsymbol{\alpha}_i = 0, \\ & 0 \leq \boldsymbol{\alpha}_i \leq C, \quad i = 1, \ldots, m. \end{aligned} \tag{3.45}$$

We can further analyse the solution $\boldsymbol{\alpha}^\star$ by inspecting the complementary slackness property of the KKT conditions, which gives

$$\begin{aligned} \boldsymbol{\alpha}_i^\star (1 - \boldsymbol{\xi}_i - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)) = 0, \quad i = 1, \ldots, m, \\ \boldsymbol{\xi}_i \, \beta_i = 0, \quad i = 1, \ldots, m. \end{aligned}$$

We consider three cases in this setting:

(i) $\boldsymbol{\alpha}_i^\star = C$

(ii) $0 < \boldsymbol{\alpha}_i^\star < C$

(iii) $\boldsymbol{\alpha}_i^\star = 0$

Case (*i*): $\boldsymbol{\alpha}_i^\star = C$ combined with the final condition of Equation (3.44) gives $\boldsymbol{\beta}_i = 0$ and therefore $\boldsymbol{\xi}_i > 0$. We also have that $1 - \boldsymbol{\xi}_i = y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$, meaning instance $\mathbf{x}_i$ must lie on the incorrect side of the margin.

Case (*ii*): $0 \leq \boldsymbol{\alpha}_i^\star \leq C$ also gives $1 - \boldsymbol{\xi}_i = y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$. Furthermore, the final condition of Equation (3.44) shows $\boldsymbol{\beta}_i \neq 0$ and hence $\boldsymbol{\xi}_i = 0$. Therefore, instance $\mathbf{x}_i$ lies directly on the margin.

Case (*iii*): $\boldsymbol{\alpha}_i^\star = 0$ implies $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 1 - \boldsymbol{\xi}_i$. The final condition of Equation (3.44) shows $\boldsymbol{\beta}_i \neq 0$ and hence $\boldsymbol{\xi}_i = 0$. Therefore, instance $\mathbf{x}_i$ is a correctly classified point lying away from the margin.

The three cases considered above provide useful insights into the solution of the dual. In the soft margin case, points belonging to cases (i) or (ii) are known as support vectors. Clearly the solution depends only on the support vectors and is therefore sparse. Furthermore, the influence of any support vector is bounded, since $\boldsymbol{\alpha}_i^\star < C$ for $i = 1, \ldots, m$.

The optimal $\mathbf{w}^\star$ of the primal can be computed according to the first condition of Equation (3.44). The optimal $b^\star$ can be computed by considering, for a point $\mathbf{x}_p$ belonging to case (ii), the following is satisfied

$$1 = y_p(\langle \mathbf{w}^\star, \mathbf{x}_p \rangle + b^\star), \tag{3.46}$$

hence

$$b^\star = y_p - \langle \mathbf{w}^\star, \mathbf{x}_p \rangle. \tag{3.47}$$

Since this equation is satisfied for all points belonging to case (ii), it is preferable to take an average. Note that computation of $\mathbf{w}^\star$ and $b^\star$ involves evaluating the inner product

$$\langle \mathbf{w}^\star, \mathbf{x} \rangle, \tag{3.48}$$

for a given $\mathbf{x}$. Using the first condition of Equation (3.44), this can be expressed as

$$\langle \mathbf{w}^\star, \mathbf{x} \rangle = \frac{1}{2} \sum_{i=1}^{m} \boldsymbol{\alpha}_i^\star \, y_i \langle \mathbf{x}, \mathbf{x}_i \rangle. \tag{3.49}$$

Hence, prediction at a new point $\tilde{\mathbf{x}}$ can be expressed as

$$f(\tilde{\mathbf{x}}) = \langle \mathbf{w}^\star, \tilde{\mathbf{x}} \rangle + b^\star = \frac{1}{2} \sum_{i=1}^{m} \alpha_i^\star y_i \langle \mathbf{x}_i, \tilde{\mathbf{x}} \rangle + b^\star, \tag{3.50}$$

where $b^\star$ is defined as above.

SVR

The dual formulation of the SVR follows a similar derivation to that of the SVM. As shown in Equation (3.51), it constitutes a convex optimisation problem over two variables. For a full derivation, please see (Schölkopf and

Smola, 2002).

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}{\text{maximise}} \quad & \sum_{i=1}^{m} y_i(\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_i^*) - \frac{1}{4}\sum_{i,j=1}^{m}(\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_i^*)(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_j^*)\langle \mathbf{x}_i, \mathbf{x}_j\rangle - \varepsilon\sum_{i=1}^{m}(\boldsymbol{\alpha}_i + \boldsymbol{\alpha}_i^*) \\
\text{subject to} \quad & \sum_{i=1}^{m}(\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_i^*) = 0 \\
& 0 \leq \boldsymbol{\alpha}_i \leq C, \quad i = 1, \ldots, m, \\
& 0 \leq \boldsymbol{\alpha}_i^* \leq C, \quad i = 1, \ldots, m.
\end{aligned} \tag{3.51}$$

For a given solution $\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\alpha}}^*$, the KKT conditions allow one to reveal similar insights to that of the SVM dual. Indeed, by labelling as support vectors the points $\mathbf{x}_p$ such that $|f(\mathbf{x}_p) - y_p| \geq \varepsilon$, it can be shown that both $\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\alpha}}_i^* = 0$ if and only if point $\mathbf{x}_i$ is not a support vector. Hence, the solution vectors are sparse and depend only on points lying on or outside the epsilon tube. The optimal $\tilde{\mathbf{w}}$, and therefore the functional form of $f(\mathbf{x})$, can be expressed entirely in terms of the dual solution:

$$f(x) = \langle \tilde{\mathbf{w}}, \mathbf{x}\rangle + \tilde{b} = \frac{1}{2}\sum_{i=1}^{m}(\tilde{\boldsymbol{\alpha}}_i - \tilde{\boldsymbol{\alpha}}_i^*)\langle \mathbf{x}_i, \mathbf{x}\rangle + \tilde{b}, \tag{3.52}$$

where the optimal $\tilde{b}$ satisfies

$$\begin{aligned}
\tilde{b} &= y_i - \langle \tilde{\mathbf{w}}, \mathbf{x}_i\rangle - \epsilon, \quad \text{for } i \in \{i | 0 < \tilde{\boldsymbol{\alpha}}_i < C\}, \\
\tilde{b} &= y_i - \langle \tilde{\mathbf{w}}, \mathbf{x}_i\rangle + \epsilon, \quad \text{for } i \in \{i | 0 < \tilde{\boldsymbol{\alpha}}_i^* < C\}.
\end{aligned}$$

Solutions

The previous sections derived the optimisation problems that are solved when training an SVM or SVR, but made no indication as to how to solve them. The problems presented in Equations (3.45) and (3.51) are known as quadratic programs (Boyd and Vandenberghe, 2004). These are a class of convex optimisation problem with a quadratic objective function and affine constraints. They are a well studied problem for which a number of general solvers exist, see (Boyd and Vandenberghe, 2004) for more details. Hence, third-party software can be used to solve the problems. Furthermore, specific analysis of the problems can lead to even more efficient solutions. The sequential minimal optimisation (SMO) algorithm takes advantage of the box-constraints on $\boldsymbol{\alpha}_i$ in order to improve computational efficiency (Platt, 1998; Takahashi et al., 2008).

3.3 Kernel Methods

The dual formulations derived in Section 3.2.4 have shown that the resulting decision function of a support vector algorithm depends only on the support vectors. However, one major drawback of both the SVM and SVR algorithm is that the resulting function is linear in the data. This property is very restrictive as to what problems the algorithms can be applied to. Indeed, for the case of the SVM, Figure 3.6 provides an example in which the problem is not solvable by a hyperplane. Similar examples can be imagined for regression, such as trying to model the function $f(\mathbf{x}) = \mathbf{x}^2$. We refer to problems of

this kind as *nonlinear problems*.

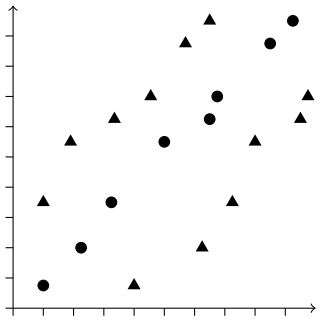


FIGURE 3.6: An example classification problem that cannot be solved with a hyperplane. The circles belong to one class and the triangles to the other.

Kernel methods define a principled methodology to model nonlinear problems. The main idea being that nonlinear problems may be solvable by linear methods in a new space. More specifically, suppose we want to classify vectors in $\mathbb{R}^n$. We are given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$ such that S represents a nonlinear problem. Let us define the function $\phi : \mathcal{X} \to \mathcal{H}$ such that $\phi(\mathbf{x})$ maps $\mathbf{x}$ to a high-dimensional space $\mathcal{H}$ in which the classification problem is now solvable by linear methods. Two questions naturally arise:

(i) Can we be sure the problem is linearly separable in $\mathcal{H}$?

(ii) How does one choose the function ϕ?

Cover's Theorem can help to answer question (i). The theorem, stated in Definition 3.3.1, depends on the notion of vectors being in *general position*. This means that, for a set of m vectors, every subset of m or fewer vectors is orthogonal.

Definition 3.3.1 (Cover's Theorem, Cover 1965)**.** Consider a set of m vectors in general position in $\mathbb{R}^n$. The number of ways to separate the points with a

hyperplane is $C(m, n)$, where

$$C(m, n) = 2 \sum_{i=0}^{n} \binom{m-1}{i}. \tag{3.53}$$

It is clear that $C(m, n)$ increases with the dimensionality n. However, for an arbitrary data set, it is unlikely that the instances will be in general position. Nevertheless, it provides intuition that moving to higher dimensions can improve our chances of finding a separating hyperplane.

3.3.1 Kernel Functions

In order to answer question (ii), let us first consider the functional form returned by the SVM. Indeed, for some $\boldsymbol{\alpha} \in \mathbb{R}^m$, the hyperplane f is of the form[3]

$$f(x) = \sum_{i=1}^{m} \boldsymbol{\alpha}_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b. \tag{3.54}$$

For a given instance $\mathbf{x}$, $f(\mathbf{x})$ depends only on the inner product between $\mathbf{x}$ and the training instances $\mathbf{x}_i$ for $i = 1, \ldots, m$.[4] Suppose instead that we have mapped all instances into $\mathcal{H}$ according to ϕ. Then f now takes the form

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle_{\mathcal{H}} + b. \tag{3.55}$$

[3] Here, we have absorbed the $\frac{1}{2}$ multiplier into the α_i for the sake of brevity.

[4] Technically, $f(\mathbf{x})$ depends on the inner product between $\mathbf{x}$ and the support vectors in the training set. For the sake of brevity, that detail is omitted in the proceeding discussion.

As before, $f(\mathbf{x})$ depends only on the inner product between $\phi(\mathbf{x})$ and the training instances $\phi(\mathbf{x}_i)$ for $i = 1, \ldots, m$. However, the inner product is now taken in the space $\mathcal{H}$. In order for this inner product to exist, we require that $\mathcal{H}$ admits a certain structure. More specifically, we require that $\mathcal{H}$ is an Hilbert Space.

Definition 3.3.2 (Hilbert Space, Schölkopf and Smola 2002)**.** Let $\mathcal{H}$ be an inner product space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ and associated norm $\| \cdot \|_{\mathcal{H}} : \mathcal{H} \to \mathbb{R}^+$ defined by

$$\| \mathbf{x} \|_{\mathcal{H}} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}_{\mathcal{H}}. \tag{3.56}$$

$\mathcal{H}$ is called a *Hilbert Space* if it is complete with respect to the norm $\| \cdot \|_{\mathcal{H}}$.

For the defintions of an inner product space and completeness, please see Kreyszig (1991). Technically, for the inner product to exist, we only require that $\mathcal{H}$ is an inner product space. However, as the following theorem shows, requiring $\mathcal{H}$ to be a Hilbert Space allows one to evaluate the inner product with a *kernel function*.

Definition 3.3.3 (Kernel Function, Steinwart and Christmann 2008)**.** Let $\mathcal{X}$ be a non-empty set. Then a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *kernel* on $\mathcal{X}$ if there exists a Hilbert space $\mathcal{H}$ and a map $\phi : \mathcal{X} \to \mathcal{H}$ such that for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ we have

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}. \tag{3.57}$$

We call ϕ a *feature map* and $\mathcal{H}$ a *feature space*.

As shown in Definition 3.3.3, choosing a specific feature map into a Hilbert space is equivalent to choosing a kernel function. However, the converse is not true, since infinitely many feature maps can be associated with a given kernel function (Minh et al., 2006). Nevertheless, for a given kernel function, there always exists a *canonical feature map* associated with the *reproducing kernel Hilbert space* (RKHS) of that kernel.

Definition 3.3.4 (Reproducing kernel Hilbert Space, Steinwart and Christmann 2008)**.** Let $\mathcal{X}$ be a non-empty set and $\mathcal{H}_k$ be a Hilbert space of functions f such that $f : \mathcal{X} \to \mathbb{R}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *reproducing kernel* of $\mathcal{H}_k$ if, for all $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}_k$, we have

(i) $k(\mathbf{x}, \cdot) \in \mathcal{H}_k$,

(ii) $f(\mathbf{x}) = \langle k(\mathbf{x}, \cdot), f \rangle_{\mathcal{H}_k}$ (reproducing property).

The space $\mathcal{H}_k$ is called a *reproducing kernel Hilbert space* over $\mathcal{X}$ if, for all $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}_k$, the evaluation functional $\mathcal{L}_\mathbf{x} : \mathcal{H}_k \to \mathbb{R}$ defined by

$$\mathcal{L}_\mathbf{x}(f) = f(\mathbf{x}), \tag{3.58}$$

is continuous.

To see that a reproducing kernel satisfies Definition 3.3.3, define the feature map $\phi_k : \mathcal{X} \to \mathcal{H}_k$ into an RKHS $\mathcal{H}_k$ such that

$$\phi_k(\mathbf{x}) = k(\mathbf{x}, \cdot), \text{ for } \mathbf{x} \in \mathcal{X}. \tag{3.59}$$

Since $k(\mathbf{x}, \cdot) \in \mathcal{H}_k$, we can apply the reproducing property of Definition 3.3.4 to yield

$$k(\mathbf{x}, \mathbf{x}') = \langle k(\mathbf{x}, \cdot), k(\mathbf{x}', \cdot) \rangle_{\mathcal{H}_k} = \langle \phi_k(\mathbf{x}), \phi_k(\mathbf{x}') \rangle_{\mathcal{H}_k}. \tag{3.60}$$

The feature map defined in Equation (3.59) is known as the canonical feature map. Furthermore, this definition defines a one-to-one correspondence between reproducing kernels and RKHSs (Steinwart and Christmann, 2008). Hence, going back to question (ii), it is natural to restrict the possible feature maps to canonical feature maps associated to a RKHS. The question is then equivalent to asking: How does one choose a kernel function? In general, the answer is problem-dependant. The choice of instance space $\mathcal{X}$ defines what kernels are applicable and, as we will see in later sections, kernels crafted for specific problems can often outperform their more general counterparts. However, we can restrict our search by considering only those functions $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which are *positive semi-definite*.

Definition 3.3.5 (Positive semi-definite property, Cristianini, Shawe-Taylor, et al. 2000)**.** Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called *positive semi-definite* if, for all $n \in \mathbb{N}$, $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n \in \mathbb{R}$ and $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, we have

$$\sum_{i,j=1}^{m} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \tag{3.61}$$

The function k is a kernel according to Definition 3.3.3 if and only if it is symmetric and positive semi-definite.

3.3.2 Learning Functions in a RKHS

As mentioned above, the use of a kernel function allows one to extend the support vector algorithms in order to learn non-linear functions. The general idea is to map the data into a feature space in which the problem is solved. The reproducing property of Definition 3.3.4 ensures we can efficiently operate in the high or even infinite-dimensional feature space without having to explicitly express the feature map. This is known as the *kernel trick*. In this section, we review the problem of supervised learning in a RKHS.

We approach the problem of supervised learning in a RKHS from the perspective of RRM. According to Equation (3.4), this is a problem of the form

$$\underset{f \in \mathcal{H}_k}{\text{minimise}} \quad \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i)) + g(\|f\|_{\mathcal{H}_k}) \tag{3.62}$$

where $\mathcal{H}_k$ is a RKHS associated with kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. This is a minimisation problem over the function space $\mathcal{H}_k$, which, in general, is not an easy problem to solve. $\mathcal{H}_k$ is not explicitly defined and can potentially be infinite-dimensional. However, Theorem 3.3.1, known as the Representer Theorem, characterises solutions to problems of the form of Equation (3.62).

Theorem 3.3.1 (Representer Theorem)**.** (Schölkopf et al., 2001) Let $\mathcal{X}$ and $\mathcal{Y}$ be a non-empty sets, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a reproducing kernel associated to a RKHS $\mathcal{H}_k$, $g : \mathbb{R} \to \mathbb{R}$ a strictly monotonically increasing function and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ a loss function. Given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$,

any solution $f^* \in \mathcal{H}_k$ to the following problem

$$\underset{f \in \mathcal{H}_k}{\text{minimise}} \quad \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i)) + g(\|f\|_{\mathcal{H}_k}) \tag{3.63}$$

admits a representation of the form

$$f^*(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, \mathbf{x}_i). \tag{3.64}$$

The remarkable Representer Theorem states that any solution to Equation (3.62) can be expressed as a linear combination of the kernel function evaluated at the training data. Hence, the problem of finding a function from the possibly infinite-dimensional space $\mathcal{H}_k$ is reduced to the finite-dimensional problem of finding the set of multipliers $\boldsymbol{\alpha}_i$ which minimise Equation (3.62). Furthermore, we have made no reference to the form of the loss function or the regularisation function. Indeed, many kernel-based learning algorithms are derived as manifestations of this theorem for differing choices of the loss function and regularisation function.

3.3.3 Kernel-SVM

In what follows, we present a derivation of the SVM from the perspective Equation (3.62). To do so, we define the loss function as the hinge loss and the

regularisation function as a positively-scaled quadratic. This leads to Equation (3.65)

$$\underset{f \in \mathcal{H}_k}{\text{minimise}} \quad \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}_k}^2, \tag{3.65}$$

where $\lambda > 0$. The Representer Theorem states that any solution $f^* \in \mathcal{H}_k$ to Equation (3.65) can be expresed as

$$f^*(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, \mathbf{x}_i). \tag{3.66}$$

Hence, a strategy to solve the problem is to plug in this representation and solve for the weights α_i. First, note that the squared norm of f^* can be written as

$$\|f^*\|_{\mathcal{H}_k}^2 = \langle f^*, f^* \rangle_{\mathcal{H}_k} = \left\langle \sum_{i=1}^{m} \boldsymbol{\alpha}_i \, k(\cdot, \mathbf{x}_i), \sum_{j=1}^{m} \boldsymbol{\alpha}_j \, k(\cdot, \mathbf{x}_j) \right\rangle_{\mathcal{H}_k}, \tag{3.67}$$

where the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_k}$ is taken in the RKHS $\mathcal{H}_k$. Since the inner product is linear, application of the reproducing property yields

$$\|f^*\|_{\mathcal{H}_k}^2 = \sum_{i,j=1}^{m} \boldsymbol{\alpha}_i \, \boldsymbol{\alpha}_j \, k(\mathbf{x}_i, \mathbf{x}_j). \tag{3.68}$$

Equation (3.65) can therefore be expressed as

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{minimise}} \quad \frac{1}{n} \sum_{i=1}^{m} \max\left(0, 1 - y_i \sum_{j=1}^{m} \boldsymbol{\alpha}_j \, k(\mathbf{x}_i, \mathbf{x}_j)\right) + \lambda \sum_{i,j=1}^{m} \boldsymbol{\alpha}_i \, \boldsymbol{\alpha}_j \, k(\mathbf{x}_i, \mathbf{x}_j) \tag{3.69}$$

We have already shown in Section 3.2.1 that optimisation of a functional involving the hinge-loss can be expressed as a constrained optimisation problem in terms of slack variables. Hence Equation (3.69) can be expressed as

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\xi}}{\text{minimise}} \; & \frac{1}{m\lambda}\sum_{i=1}^{m}\boldsymbol{\xi}_i + \sum_{i,j=1}^{m}\boldsymbol{\alpha}_i\,\boldsymbol{\alpha}_j\,k(\mathbf{x}_i,\mathbf{x}_j) \\ \text{subject to} \quad & y_i\sum_{j=1}^{m}\boldsymbol{\alpha}_j\,k(\mathbf{x}_i,\mathbf{x}_j) \geq 1-\boldsymbol{\xi}_i, \qquad i=1,\ldots,m, \\ & \boldsymbol{\xi}_i \geq 0, \qquad\qquad\qquad\qquad\quad i=1,\ldots,m. \end{aligned} \tag{3.70}$$

This problem is a constrained convex optimisation problem in terms of $\boldsymbol{\alpha} \in \mathbb{R}^m$ and $\boldsymbol{\xi} \in \mathbb{R}^m$. It is the primal form of the *kernel-SVM* problem. Similarly to the dual of the soft-margin SVM, the dual of the kernel-SVM is expressed in terms of one variable. To derive the dual, we first express Equation (3.70) in terms of matrices.

Let us define the matrix $\mathbf{K} \in \mathbb{R}^{m\times m}$ such that its i,j entries are equal to $k(\mathbf{x}_i,\mathbf{x}_j)$. This is known as the *kernel matrix*. Furthermore, define $\mathbf{Y} \in \mathbb{R}^{m\times m}$ as a diagonal matrix whose ith diagonal element entry is equal to y_i, $\mathbf{1}_m \in \mathbb{R}^m$ as the vector of all ones and $\mathbf{0}_m \in \mathbb{R}^m$ as the vector of all zeroes. Equation (3.70) can be compactly expressed as[5]

[5]The $\succeq$ symbol is an element-wise application of $\geq$.

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\xi}}{\text{minimise}} \quad & \frac{1}{m\lambda}\,\mathbf{1}_m^T\,\boldsymbol{\xi} + \boldsymbol{\alpha}^T\,\mathbf{K}\,\boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{Y}\,\mathbf{K}\,\boldsymbol{\alpha} \succeq \mathbf{1}_m - \boldsymbol{\xi}, \\ & \boldsymbol{\xi} \succeq \mathbf{0}_m\,. \end{aligned} \tag{3.71}$$

The Lagrangian of Equation (3.71) is

$$L(\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\beta},\boldsymbol{\omega}) = \frac{1}{m\lambda}\,\mathbf{1}_m^T\,\boldsymbol{\xi} + \boldsymbol{\alpha}^T\,\mathbf{K}\,\boldsymbol{\alpha} + \boldsymbol{\beta}^T(\mathbf{1}^m - \boldsymbol{\xi} - \mathbf{Y}\,\mathbf{K}\,\boldsymbol{\alpha}) - \boldsymbol{\omega}^T\,\boldsymbol{\xi}, \tag{3.72}$$

where $\boldsymbol{\beta} \in \mathbb{R}^m$ and $\boldsymbol{\omega} \in \mathbb{R}^m$ are the Lagrangian Multipliers. Similarly to the hard-margin SVM, strong duality holds via application of Slater's Condition. Since the problem is differentiable, the KKT conditions can be applied. The derivatives of the Lagrangian with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\xi}$ are

$$\begin{aligned} \frac{\partial L(\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\beta},\boldsymbol{\omega})}{\partial\,\boldsymbol{\alpha}} &= \mathbf{K}(2\,\boldsymbol{\alpha} - \mathbf{Y}\,\boldsymbol{\beta}), \\ \frac{\partial L(\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\beta},\boldsymbol{\omega})}{\partial\,\boldsymbol{\xi}} &= \frac{1}{m\lambda}\,\mathbf{1}_m - \boldsymbol{\beta} - \boldsymbol{\omega}\,. \end{aligned}$$

Equating both to $\mathbf{0}_m$ gives the following sufficient conditions

$$\begin{aligned} \boldsymbol{\alpha} &= \frac{1}{2}\,\mathbf{Y}\,\boldsymbol{\beta}, \\ \mathbf{0}_m &= \frac{1}{m\lambda}\,\mathbf{1}_m - \boldsymbol{\beta} - \boldsymbol{\omega}\,. \end{aligned}$$

Substituting these expressions back into the Lagrangian and applying the

dual feasibility property of Definition 3.2.8 gives Equation (3.74), the Kernel-SVM Dual.

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{maximise}} \quad & \mathbf{1}_m^T \boldsymbol{\beta} - \frac{1}{4} \boldsymbol{\beta}^T \mathbf{Y} \mathbf{K} \mathbf{Y} \boldsymbol{\beta} \\ \text{subject to} \quad & \mathbf{0}_m \preceq \boldsymbol{\beta} \preceq \frac{1}{m\lambda} \mathbf{1}_m \,. \end{aligned} \tag{3.73}$$

(3.74)

Whilst we omit the specific analysis, the complementary slackness property of the KKT conditions allows one to draw similar conclusions regarding support vectors to that of the soft-margin SVM. Prediction at a new point $\tilde{\mathbf{x}}$ is expressed in terms of $\boldsymbol{\beta}$ and $\mathbf{k}_{\tilde{\mathbf{x}}} = (k(\mathbf{x}_1, \tilde{\mathbf{x}}), \ldots, k(\mathbf{x}_m, \tilde{\mathbf{x}}))^T$ as

$$f(\tilde{\mathbf{x}}) = \frac{1}{2} \mathbf{k}_{\tilde{\mathbf{x}}}^T \mathbf{Y} \boldsymbol{\beta} \tag{3.75}$$

3.3.4 Kernel SVR

The SVR algorithm can also be derived from the perspective of RRM in a RKHS. In particular, using the ε-insensitive loss and a positively-scaled quadratic regulariser leads to the primal form of Kernel SVR, as given in Equation (3.76).

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\xi}^*}{\text{minimise}} \quad & \frac{1}{m\lambda}\,\mathbf{1}_m^T(\boldsymbol{\xi}+\boldsymbol{\xi}^*) + \boldsymbol{\alpha}^T\,\mathbf{K}\,\boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{K}\,\boldsymbol{\alpha} - \mathbf{y} \preceq \varepsilon\,\mathbf{1}_m + \boldsymbol{\xi}, \\ & \mathbf{y} - \mathbf{K}\,\boldsymbol{\alpha} \preceq \varepsilon\,\mathbf{1}_m + \boldsymbol{\xi}^*, \\ & \boldsymbol{\xi},\boldsymbol{\xi}^* \succeq \mathbf{0}_m\,. \end{aligned} \tag{3.76}$$

Here, $\boldsymbol{\xi},\boldsymbol{\xi}^* \in \mathbb{R}^m$ are the two sets of slack variables corresponding to the upper and lower boundaries of the epsilon tube and $\mathbf{y} \in \mathbb{R}^m$ is a vector whose ith element equals y_i. This is a convex optimisation problem with differentiable objective and constraint functions. Furthermore, one can trivially find a point such that Slater's condition holds. Hence, the KKT conditions can be applied. The dual formulation of the Kernel SVR problem is given in Equation (3.78).

$$\begin{aligned} \underset{\boldsymbol{\beta},\boldsymbol{\beta}^*}{\text{maximise}} \quad & \mathbf{y}^T(\boldsymbol{\beta}^*-\boldsymbol{\beta}) - \frac{1}{4}(\boldsymbol{\beta}^*-\boldsymbol{\beta})^T\,\mathbf{K}(\boldsymbol{\beta}^*-\boldsymbol{\beta}) - \varepsilon\,\mathbf{1}_m^T(\boldsymbol{\beta}^*+\boldsymbol{\beta}) \\ \text{subject to} \quad & \mathbf{0}_m \preceq \boldsymbol{\beta},\boldsymbol{\beta}^* \preceq \frac{1}{m\lambda}\,\mathbf{1}_m\,. \end{aligned} \tag{3.77}$$

(3.78)

Chapter 4

Kreĭn Space Methods for Structured Data

This chapter presents the main methodological contributions of the thesis. In particular, we derive the dual problem of an SVM in a Kreĭn space and present a novel, indefinite string kernel rooted in compression, the string compression kernel (SCK). The first section provides an overview of the fundamental theory of Kreĭn spaces and the notion of a Reproducing Kernel Kreĭn Space in Section 4.1. This theory is used in the following section to derive the dual problem of the Kreĭn-SVM. Similarly to a kernel-SVM, the derivation results in a quadratic program that can be readily solved with an off-the-shelf solver. We conclude the chapter with a discussion of the SCK. Using the notion of Kolmogorov Complexity (Cover, 1999; Kolmogorov, 1965) and its approximation via real-world compression software, we propose to measure the similarity of two strings by comparing their lenghts when compressed.

4.1 Kreĭn Space Theory

Let $\mathcal{K}$ be a vector space equipped with a bilinear form $\langle \cdot, \cdot \rangle_{\mathcal{K}} : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$. The bilinear form is called positive if $\langle f, f \rangle_{\mathcal{K}} \geq 0$ for all $f \in \mathcal{K}$. It is called symmetric if, for all $f, g \in \mathcal{K}$, $\langle f, g \rangle_{\mathcal{K}} = \langle g, f \rangle_{\mathcal{K}}$. The bilinear form is called indefinite if there exists $f, g \in \mathcal{K}$ such that $\langle f, f \rangle_{\mathcal{K}} > 0$ and $\langle g, g \rangle_{\mathcal{K}} < 0$. We call the form non-degenerate if, for $f \in \mathcal{K}$, $\langle f, g \rangle_{\mathcal{K}} = 0$ for all $g \in \mathcal{K}$ implies $f = 0$. Any non-degenerate, symmetric and positive bilinear form is called an inner product.

To formally define a Kreĭn space, we must first introduce the notion of a direct sum. This is given in Definition 4.1.1.

Definition 4.1.1 (Direct Sum, Kreyszig 1991)**.** A vector space $\mathcal{K}$ is said to be the direct sum of two subspaces $\mathcal{K}_1, \mathcal{K}_2$, written

$$\mathcal{K} = \mathcal{K}_1 \oplus \mathcal{K}_2, \tag{4.1}$$

if each $f \in \mathcal{K}$ has a unique representation

$$f = f_1 + f_2 \tag{4.2}$$

for $f_1 \in \mathcal{K}_1, f_2 \in \mathcal{K}_2$. To be more explicit, we often write

$$f = f_1 \oplus f_2. \tag{4.3}$$

As shown in Definition 4.1.2, a Kreĭn space is simply a vector space equipped

with a bilinear form that can be expressed as an orthogonal direct sum of two Hilbert spaces.

Definition 4.1.2 (Kreĭn Space, Ong et al. 2004)**.** Let $\mathcal{K}$ be a vector space equipped with a bilinear form $\langle \cdot, \cdot \rangle_{\mathcal{K}}$. The pair $(\mathcal{K}, \langle \cdot, \cdot \rangle_{\mathcal{K}})$ is called a Kreĭn space if there exist two linear manifolds $\mathcal{H}_{\pm}$ in $\mathcal{K}$ with inner products $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\pm}}$ such that

$$\mathcal{K} = \mathcal{H}_{+} \oplus \mathcal{H}_{-}. \tag{4.4}$$

The pairs $(\mathcal{H}_{\pm}, \langle \cdot, \cdot \rangle_{\mathcal{H}_{\pm}})$ define Hilbert spaces which are orthogonal to each other.

A consequence of Definition 4.1.2, given in Corollary 4.1.1, is that the bilinear form of a Kreĭn space can be expressed as the difference of inner products in the asscociated Hilbert spaces. Hence, it is a symmetric, non-degenerate and indefinite bilinear form.

Corollary 4.1.1. As in Definition 4.1.2, let $(\mathcal{K}, \langle \cdot, \cdot \rangle_{\mathcal{K}})$ be a Kreĭn space such that $\mathcal{K} = \mathcal{H}_{+} \oplus \mathcal{H}_{-}$ for Hilbert spaces $(\mathcal{H}_{\pm}, \langle \cdot, \cdot \rangle_{\mathcal{H}_{\pm}})$. For elements $f, g \in \mathcal{K}$, $f_{\pm}, g_{\pm} \in \mathcal{H}_{\pm}$ such that $f = f_{+} \oplus f_{-}$ and $g = g_{+} \oplus g_{-}$, the bilinear form $\langle \cdot, \cdot \rangle_{\mathcal{K}}$ can be expressed as

$$\langle f, g \rangle_{\mathcal{K}} = \langle f_{+}, g_{+} \rangle_{\mathcal{H}_{+}} - \langle f_{-}, g_{-} \rangle_{\mathcal{H}_{-}}. \tag{4.5}$$

The decomposition in Corollary 4.1.1 allows one to define a Hilbert space by replacing the difference of inner products with a sum. This space, known as the associated Hilbert space, is defined in Definition 4.1.3.

Definition 4.1.3 (Asscociated Hilbert Space, Ong et al. 2004)**.** Let $(\mathcal{K}, \langle\cdot,\cdot\rangle_{\mathcal{K}})$ be a Kreĭn space with decomposition into Hilbert spaces $(\mathcal{H}_{\pm}, \langle\cdot,\cdot\rangle_{\mathcal{H}_{\pm}})$. Define $(\mathcal{H}_{\mathcal{K}}, \langle\cdot,\cdot\rangle_{\mathcal{H}_{\mathcal{K}}})$ as the Hilbert space such that

(i) $\mathcal{H}_{\mathcal{K}} = \mathcal{H}_+ \oplus \mathcal{H}_-$

(ii) $\langle f, g\rangle_{\mathcal{H}_{\mathcal{K}}} = \langle f_+, g_+\rangle_{\mathcal{H}_+} + \langle f_-, g_-\rangle_{\mathcal{H}_-}, \quad (f_{\pm}, g_{\pm} \in \mathcal{H}_{\pm}).$

We say $(\mathcal{H}_{\mathcal{K}}, \langle\cdot,\cdot\rangle_{\mathcal{H}_{\mathcal{K}}})$ is the *associated Hilbert space* of the Kreĭn space $(\mathcal{K}, \langle\cdot,\cdot\rangle_{\mathcal{K}})$.

For a Kreĭn space $\mathcal{K}$, the decomposition $\mathcal{K} = \mathcal{H}_+ \oplus \mathcal{H}_-$ is not necessarily unique. Generally, a Kreĭn space can be associated to infinitely many Hilbert spaces. However, the topology introduced on $\mathcal{K}$ via the norm $\|f\|_{\mathcal{H}_{\mathcal{K}}} = \sqrt{\langle f, f\rangle}_{\mathcal{H}_{\mathcal{K}}}$ is independent of the specific decomposition and associated Hilbert space (Oglic and Gärtner, 2019). We refer to this topology as the *strong topology*.

Given an instance space $\mathcal{X}$, the evaluation functional $\mathcal{L}_x : \mathbb{R}^{\mathcal{X}} \to \mathbb{R}$ on $\mathcal{K}$ is an operator that evaluates each function $f \in \mathcal{K}$ at a point $x \in \mathcal{X}$. We call the Kreĭn space $\mathcal{K}$ a reproducing kernel Kreĭn space (RKKS) if the evaluation functional is continuous on $\mathcal{K}$ with respect to the strong topology (Alpay, 1991; Ong et al., 2004). The following theorem provides a useful characterisation of RKKSs and their relation to RKHSs.

Theorem 4.1.1 (Kreĭn Kernel, Alpay 1991)**.** Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a real-valued, symmetric function. Then, there is an associated RKKS if and only if there exists positive definite kernels k_+ and k_- such that

$$k = k_+ - k_-. \tag{4.6}$$

If k admits such a decomposition, k_+ and k_- can be chosen such that the corresponding RKHSs are disjoint.

A consequence of Theorem 4.1.1 allows us to define an analog of the reproducing property of an RKHS. This is given in Corollary 4.1.2.

Corollary 4.1.2 (Reproducing Kreĭn Kernel, Ong et al. 2004)**.** Let $(\mathcal{K}, \langle\cdot,\cdot\rangle_{\mathcal{K}})$ be a RKKS with decomposition into Hilbert spaces $(\mathcal{H}_\pm, \langle\cdot,\cdot\rangle_{\mathcal{H}_\pm})$. Then $(\mathcal{H}_\pm, \langle\cdot,\cdot\rangle_{\mathcal{H}_\pm})$ are RKHSs with kernels $k_\pm$. Furthermore, there is a unique symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for all $f \in \mathcal{K}$ and $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$,

(i) $k(\mathbf{x}, \cdot) \in \mathcal{K}$,

(ii) $\langle f, k(\mathbf{x}, \cdot)\rangle_{\mathcal{K}} = f(\mathbf{x})$ (reproducing property),

(iii) $k(\mathbf{x}, \mathbf{x}') = k_+(\mathbf{x}, \mathbf{x}') - k_-(\mathbf{x}, \mathbf{x}')$.

We call k the *reproducing Kreĭn kernel* of $\mathcal{K}$.

A RKKS can also be associated to a Hilbert space in the same manner as given in Definition 4.1.3. In this case, the associated space is a RKHS whose kernel is defined as the sum of the kernels from the decomposition spaces.

The notion of a Kreĭn space and also a RKKS generalises that of a Hilbert space and a RKHS, respectively. Indeed, a Hilbert space can be seen as a Kreĭn space such that $\mathcal{H}_- = 0$. Similarly, a RKKS can be seen as a RKHS in which $\mathcal{H}_- = 0$, or equivalently, $k_- = 0$.

4.2 Classification in a RKKS

Having reviewed the relevant theory of Kreĭn spaces and RKKSs, we now move on to function estimation. In this section, we derive a support vector classifier for a hypothesis belonging to a RKKS. We will focus on the problem of RRM in a RKKS. Recall that RRM involves minimising the empirical risk together with a regularisation term. As a loss function, we choose the squared-hinge loss. The algorithms using the hinge and squared-hinge loss functions are derived in a similar manner. However, the squared-hinge provides greater penalisation to instances residing on the incorrect side of the hyperplane. As a regularisation term, we select the norm induced via the the strong topology in which each decomposition component is separately scaled. That is, for a RKKS $(\mathcal{K}, \langle\cdot,\cdot\rangle_{\mathcal{K}})$ we regularise with the term

$$\lambda_+ \|f_+\|^2_{\mathcal{H}_+} + \lambda_- \|f_-\|^2_{\mathcal{H}_-}. \tag{4.7}$$

Providing distinct regularisation parameters $\lambda_\pm$ for each component allows for greater control over the hypothesis. Previous approaches have regularised with respect to the bilinear form $\langle f, f\rangle_{\mathcal{K}} = \|f_+\|^2_{\mathcal{H}_+} - \|f_+\|^2_{\mathcal{H}_-}$ of the RKKS (Ong et al., 2004). However, this does not define a norm. These approaches have been empirically shown to produce hypotheses that struggle to generalise to unseen instances (Oglic and Gärtner, 2018). Furthermore, for specific loss functions, regularising with respect to the strong topology leads to convex optimisation problems.

Given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \{-1, +1\}$, positive scalars

$\lambda_\pm > 0$ and a RKKS $\mathcal{K}$, we aim to solve the optimisation problem of Equation (4.8).

$$\underset{f\in\mathcal{K}}{\text{minimise}} \quad \frac{1}{2}\sum_{i=1}^{m}\max(0,1-y_i f(\mathbf{x}_i))^2 + \frac{\lambda_+}{2}\|f_+\|^2_{\mathcal{H}_+} + \frac{\lambda_-}{2}\|f_-\|^2_{\mathcal{H}_-}, \tag{4.8}$$

We refer to Equation (4.8) as the *Kreĭn-SVM* problem. It is a minimisation problem over a potentially infinite function space and is thus not an easy problem to solve. However, as shown in Theorem 4.2.1, a representer-like theorem holds for Equation (4.8).

Theorem 4.2.1. Let $f^* \in \mathcal{K}$ be an optimal solution to the Kreĭn-SVM problem of Equation (4.8) and let $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ be the reproducing Kreĭn kernel associated to $\mathcal{K}$. Then f^* admits a representation of the form

$$f^* = \sum_{i=1}^{m}\alpha_i k(\cdot,\mathbf{x}_i). \tag{4.9}$$

The proof of Theorem 4.2.1 is similar to that provided by Oglic and Gärtner (2018), with a slight modification for the different objective function. Similarly to the derivation of the kernel-SVM, the representer theorem allows us to express the solution to Equation (4.8) in terms of a minimisation problem over a finite domain. Firstly, note that for $f^* = \sum_{i=1}^m \alpha_i k(\cdot,\mathbf{x}_i)$ and $f^* = f^*_+ \oplus f^*_-$ we have $f^*_\pm = \sum_{i=1}^m \alpha_i k_\pm(\cdot,\mathbf{x}_i)$. Therefore, we obtain

$$\begin{aligned}\underset{\boldsymbol{\alpha}}{\text{minimise}} \quad & \frac{1}{2}\sum_{i=1}^{m}\max(0,1-y_i\sum_{j=1}^{m}\boldsymbol{\alpha}_i\, k(\mathbf{x}_i,\mathbf{x}_j))^2 \\ & + \frac{\lambda_+}{2}\sum_{i,j=1}^{m}\boldsymbol{\alpha}_i\,\boldsymbol{\alpha}_j\, k_+(\mathbf{x}_i,\mathbf{x}_j) + \frac{\lambda_-}{2}\sum_{i,j=1}^{m}\boldsymbol{\alpha}_i\,\boldsymbol{\alpha}_j\, k_-(\mathbf{x}_i,\mathbf{x}_j).\end{aligned} \tag{4.10}$$

This can be expressed in terms of slack variables as

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\xi}}{\text{minimise}} \quad & \frac{1}{2}\sum_{i=1}^{m}\boldsymbol{\xi}_i^2 + + \frac{\lambda_+}{2}\sum_{i,j=1}^{m}\boldsymbol{\alpha}_i\,\boldsymbol{\alpha}_j\,k_+(\mathbf{x}_i,\mathbf{x}_j) + \frac{\lambda_-}{2}\sum_{i,j=1}^{m}\boldsymbol{\alpha}_i\,\boldsymbol{\alpha}_j\,k_-(\mathbf{x}_i,\mathbf{x}_j) \\ \text{subject to} \quad & y_i\sum_{j=1}^{m}\boldsymbol{\alpha}_j\,k(\mathbf{x}_i,\mathbf{x}_j) \geq 1-\boldsymbol{\xi}_i, \quad i=1,\ldots,m. \end{aligned} \tag{4.11}$$

As with the kernel-SVM, it is simpler to proceed by first expressing Equation (4.11) in terms of matrices. Define $\mathbf{Y} \in \mathbb{R}^{m\times m}$, $\mathbf{1}_m \in \mathbb{R}^m$ and $\mathbf{0}_m \in \mathbb{R}^m$ as was done in Section 3.3.3. Define the matrix $\mathbf{K} \in \mathbb{R}^{m\times m}$ such that its i,j entries are equal to $k(\mathbf{x}_i,\mathbf{x}_j)$. Similarly, define the matrices $\mathbf{K}\pm \in \mathbb{R}^{m\times m}$ with i,j entries are equal to $k_\pm(\mathbf{x}_i,\mathbf{x}_j)$. It follows that $\mathbf{K} = \mathbf{K}_+ - \mathbf{K}_-$. In practice, the matrices $\mathbf{K}_\pm$ are computed from $\mathbf{K}$ via its eigendecomposition. In particular, let the eigendecomposition be $\mathbf{K} = \mathbf{V}\,\boldsymbol{\Sigma}\,\mathbf{V}^T$ where $\mathbf{V}$ is the matrix with eigenvectors as columns and $\boldsymbol{\Sigma}$ is the diagonal matrix with eigenvalues along its diagonal. Defining $\boldsymbol{\Sigma}_\pm$ as the diagonal matrices that contain the absolute values of the positive/negative eigenvalues of $\boldsymbol{\Sigma}$, we have that $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_+ - \boldsymbol{\Sigma}_-$ and hence $\mathbf{K}_\pm = \mathbf{V}\,\boldsymbol{\Sigma}_\pm\,\mathbf{V}^T$. Furthermore, for $\sigma_i(\mathbf{K})$ being the ith eigenvalue of $\mathbf{K}$, let $\boldsymbol{\Lambda}$ be the diagonal matrix whose ith diagonal entry $\boldsymbol{\Lambda}_{i,i}$ can be expressed as

$$\boldsymbol{\Lambda}_{i,i} = \begin{cases} \lambda_+, & \sigma_i(\mathbf{K}) \geq 0, \\ -\lambda_-, & \sigma_i(\mathbf{K}) < 0. \end{cases} \tag{4.12}$$

These definitions lead to the equivalent problem given in Equation (4.13).

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\xi}}{\text{minimise}} \quad & \frac{1}{2}\,\boldsymbol{\xi}^T\,\boldsymbol{\xi} + + \frac{1}{2}\,\boldsymbol{\alpha}^T\,\mathbf{H}\,\mathbf{K}\,\boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{Y}\,\mathbf{K}\,\boldsymbol{\alpha} \succeq \mathbf{1}_m - \boldsymbol{\xi}, \end{aligned} \tag{4.13}$$

where $\mathbf{H} = \mathbf{V}\,\boldsymbol{\Lambda}\,\mathbf{V}^T$. Upon inspection, this problem looks remarkably similar to that of the kernel-SVM. We do not require the positivity constraint on the $\boldsymbol{\xi}$ since the square of the values are incorporated into the objective function. However, apart from that, the only notable difference is the replacement of $\mathbf{H}\,\mathbf{K}$ in place of the standard kernel matrix. By construction, this product can be expressed as $\mathbf{H}\,\mathbf{K} = \lambda_+\,\mathbf{K}_+ + \lambda_-\,\mathbf{K}_-$. Hence, this problem reduces to a standard squared-hinge SVM when the kernel function is positive-definite i.e. $\mathbf{K}_- = 0$.

The problem presented in Equation (4.13) is a constrained convex optimisation problem over $\boldsymbol{\alpha} \in \mathbb{R}^m$ and $\boldsymbol{\xi} \in \mathbb{R}^m$. Viewing it as the primal problem, we can apply the theory of Langrangian Duality in order to derive the dual. The Lagrangian is defined as

$$L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\beta}) = \frac{1}{2}\,\boldsymbol{\xi}^T\,\boldsymbol{\xi} + + \frac{1}{2}\,\boldsymbol{\alpha}^T\,\mathbf{H}\,\mathbf{K}\,\boldsymbol{\alpha} + \boldsymbol{\beta}(\mathbf{1}_m - \boldsymbol{\xi} - \mathbf{Y}\,\mathbf{K}\,\boldsymbol{\alpha}). \tag{4.14}$$

where $\boldsymbol{\beta} \in \mathbb{R}^m$ is the Langrangian Multiplier. Similarly to the kernel-SVM problem, strong duality holds via application of Slater's Condition. Since the problem is differentiable, the KKT conditions can be applied. The derivatives

of the Langrangian with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\xi}$ are

$$\frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\beta}, \boldsymbol{\omega})}{\partial \boldsymbol{\alpha}} = \mathbf{H}\,\mathbf{K}\,\boldsymbol{\alpha} - \mathbf{K}\,\mathbf{Y}\,\boldsymbol{\beta},$$
$$\frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\beta}, \boldsymbol{\omega})}{\partial \boldsymbol{\beta}} = \boldsymbol{\xi} - \boldsymbol{\beta}\,.$$

Setting these equations equal to $\mathbf{0}_m$ gives the following conditions

$$\mathbf{H}\,\mathbf{K}\,\boldsymbol{\alpha} = \mathbf{K}\,\mathbf{Y}\,\beta,$$

$$\boldsymbol{\xi} = \boldsymbol{\beta},$$

or, equivalently,

$$\boldsymbol{\alpha} = \mathbf{H}^{-1}\,\mathbf{Y}\,\boldsymbol{\beta}\,. \tag{4.15}$$

Substituting these expressions back into the Langrangian and applying the dual feasibility property of Definition 3.2.8 gives Equation (4.16), the Kreĭn-SVM Dual.

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{maximise}} \quad & \mathbf{1}_m^T\,\boldsymbol{\beta} - \frac{1}{2}\,\boldsymbol{\beta}^T(\mathbf{Y}\,\mathbf{K}\,\mathbf{H}^{-1}\,\mathbf{Y} + \mathbf{I}_m)\,\boldsymbol{\beta} \\ \text{subject to} \quad & \boldsymbol{\beta} \succeq \mathbf{0}_m\,. \end{aligned} \tag{4.16}$$

As with the kernel-SVM, Equation (4.16) is a quadratic program and can be solved efficiently with an off-the-shelf solver. Computation of the inverse matrix may can be performed efficiently by recalling that $\mathbf{H} = \mathbf{V}\,\boldsymbol{\Lambda}\,\mathbf{V}^T$, and therefore $\mathbf{H}^{-1} = \mathbf{V}\,\boldsymbol{\Lambda}^{-1}\,\mathbf{V}^T$. Since $\boldsymbol{\Lambda}$ is a diagonal matrix, its inverse is the

matrix whose diagonal entries are the reciprocal of those of $\mathbf{\Lambda}$.

4.3 Compression-Based Similarity

Compression-based similarity methods have been widely used across many areas of informatics, including, for example, the comparison of molecules in cheminformatics (Melville et al., 2007) and as feature space embeddings in machine learning (Sculley and Brodley, 2006). Motivated by this, we propose to measure the similarity of strings by comparing their lengths when compressed, and define an intuitive string kernel that is not a positive-definite function. We call this kernel the *string compression kernel* (SCK). Before moving forward, we clarify our terminology.

Let Σ be a finite alphabet, $\Sigma^n \subseteq \Sigma$ be the set of all strings of length n from Σ and Σ^* the set of all strings from that alphabet. A string $s \in \Sigma^*$ of length $|s|$ is a sequence of characters that can be indexed as $s = s_1 \dots s_{|s|}$.

Intuitive Construction In order to understand the motivation for the SCK in more detail, consider a compression function $f : \Sigma^* \to \Sigma^*$ and two strings $s, s' \in \Sigma^*$. Furthermore, denote by ss' the concatenation of s and s', $|f(s)|$ the size of the string s when compressed by f and consider the function

$$d(s, s') = |f(s)| - |f(ss')|. \tag{4.17}$$

For an efficient compressor, one would expect $d(s, s)$ to be small in magnitude. Compressing a string concatenated with itself should require little

overhead compared to compressing the string directly. Hence, the size of the compressed strings should be similar. This is the idea that underpins the SCK. Furthermore, it can be expanded to similar and dissimilar strings. Consider three strings $t, u, v \in \Sigma^*$ such that u and v are similar strings whilst v and t are dissimilar strings [1]. The difference $d(v,u) - d(v,t)$ can be expressed as

$$d(v,u) - d(v,t) = |f(vt)| - |f(vu)|. \tag{4.18}$$

Given that v and u are similar, a compressor should be able to utilise the information provided by v in order to compress u more efficiently than that of t. Hence, one would expect $|f(vt)|$ to be greater in magnitude than $|f(vu)|$, implying that $d(v,u) - d(v,t)$ is positive. That is, $d(v,u) > d(v,t)$ when v is more similar to u than it is to t. Whilst not a formal argument, this theoretical scenario provides intuition into the SCK. In order to transform this idea into a valid Kreĭn kernel, we require a symmetric function. Hence, for strings s and s', the SCK kernel $k_{SCK} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ is simply the sum of $d(s,s')$ and $d(s',s)$, as shown below

$$k_{SCK}(s,s') = |f(s)| + |f(s')| - |f(ss')| - |f(s's)|. \tag{4.19}$$

Formal Construction The intuitive construction of the SCK can be formalised by considering the notion of Kolmogorov complexity, the definition of which is given below:

[1] In this hypothetical scenario, any notion of similarity is sufficient. The main idea is that of similar and dissimilar strings.

Definition 4.3.1 (Kolmogorov complexity, Li, Vitányi, et al. 2008)**.** Let $s \in \Sigma^*$ be a string and $K : \Sigma^* \to \mathbb{N}$ a function such that $K(s)$ is *the length of the shortest computer program, in a predefined language, that produces s as output.* We call $K(s)$ the *Kolmogorov complexity* of s.

The Kolmogorov complexity of a string or, more generally, an arbitrary digital object, is essentially the length of the ultimate compressed version of the object (Li et al., 2004). The notion can be extended into a conditional form, in which an auxiliary input is provided:

Definition 4.3.2 (Conditional Kolmogorov complexity, Li, Vitányi, et al. 2008)**.** Let $s, s' \in \Sigma^*$ be two strings and $K_C : \Sigma^* \times \Sigma^* \to \mathbb{N}$ a function such that $K_C(s \mid s')$ is *the length of the shortest computer program, in a predefined language, that produces s as output when s' is provided as an auxiliary input.* We call $K_C(s \mid s')$ the *Conditional Kolmogorov complexity* of s given s'.

We can also consider the joint complexity of two strings:

Definition 4.3.3 (Joint Kolmogorov complexity, Li, Vitányi, et al. 2008)**.** Let $s, s' \in \Sigma^*$ be two strings and $K_J : \Sigma^* \times \Sigma^* \to \mathbb{N}$ a function such that $K_J(s, s')$ is *the length of the shortest computer program, in a predefined language, that produces s and s' as output, as well as a method to distinguish them.* We call $K_J(s, s')$ the *Joint Kolmogorov Complexity* of s and s'.

Finally, for two strings $s, s' \in \Sigma^*$, the various notions of Kolmogorov complexity are related via the following formula:

$$K_C(s \mid s') = K_J(s, s') - K(s'). \tag{4.20}$$

Whilst intuitively simple, the preceding definitions are not computable functions. Hence, in order to incorporate them into a notion of string similarity, once must approximate the various notions of Kolmogorov complexity. It turns out that real-world compression software serves as a good proxy for the Kolmogorov complexity (Li et al., 2004). Indeed, given a string s and a compression function $f : \Sigma^* \to \Sigma^*$, the Kolmogorov complexity of s is well-approximated by $|f(s)|$. We have that $K(s)$ serves as the universal, lower bound that any real-world compressor can possibly achieve (Li et al., 2004).

This notion also extends to the joint complexity, in which for two strings $s, s' \in \Sigma^*$ and a compression function $f : \Sigma^* \to \Sigma^*$, the joint complexity of s and s' is well-approximated by $|f(ss')|$, where ss' denotes the concatenation of s and s'. Hence, by Equation (4.20), the conditional Kolmogorov complexity of s given s' can be approximated by $|f(ss')| - |f(s')|$. That is,

$$K_{\mathrm{C}}(s \mid s') \approx |f(ss')| - |f(s')|. \tag{4.21}$$

Therefore the SCK can be expressed as

$$k_{SCK}(s, s') \approx -\Big(K_{\mathrm{C}}(s \mid s') + K_{\mathrm{C}}(s' \mid s) \Big). \tag{4.22}$$

We note that the negative sign in Equation (4.22) is merely a constant and does not defer from the fact the the SCK approximates the sum of conditional Kolmogorov complexities.

4.4 Software and Programming Languages

The main goal of the software developed in this thesis is to produce reliable results. To ensure trustworthiness, the code must be thoroughly tested, well-documented, concise, and accessible. Therefore, the proof-of-concept reference implementations are created using the `Python` scripting language. This offers readability and brevity, enabling fast development times and reducing the number of potential errors. As the software is scientific, the standard `Python` scientific development stack of `numpy`, `pandas`, and `matplotlib` is used. We utilise the machine learning framework `scikit-learn` for useful preprocessing functionality and the convex optimisation framework `cvxopt` to fit the Kreĭn-SVM. Reference implementations of both the Kreĭn-SVM and SCK are provided at `https://github.com/Mrjoeybux/KreinAMP` and `https://github.com/Mrjoeybux/strukern`, respectively.

Chapter 5

Identification of Translation Initiation Sites

This chapter describes and evaluates a framework to develop models that identify TIS codons. We assess the effectiveness of our proposed approach on two datasets. The first is a synthetic string dataset in which the instances are distributed according to a k-order Markov model. The second is a real-world dataset of mRNA sequences and their respective TIS codons. The synthetic dataset is designed to suit our methodology well, and our results support this. The results on the real-world dataset are not as clear-cut, indicating that a k-order Markov model can not fully describe mRNA sequences. The rest of the chapter is organised as follows: First, we provide a background on the problem of TIS identification, giving a broad description of gene expression and how TIS codons fit into this. This is followed by a detailed description of our methodology, in which we discuss the datasets and kernel functions used in our experiments and our computational setup. Finally, we discuss the results of our experiments and propose some avenues for further work.

5.1 Background

Gene expression is a fundamental biological process that converts the genetic information encoded in DNA into functional proteins. It consists of two main stages: *transcription* and *translation*. Transcription, which occurs in the nucleus of a cell, copies the information encoded in DNA into messenger RNA (mRNA). Once created, the mRNA molecule leaves the nucleus and starts the translation process. This process consists of three main stages: initiation, elongation, and termination. The ribosome assembles on the mRNA during initiation and locates the TIS, where it begins synthesizing the polypeptide chain. Codons, a sequence of three consecutive nucleotides in the mRNA molecule, correspond to specific amino acids. Elongation involves the sequential reading of codons and the addition of corresponding amino acids to the growing polypeptide chain. Termination occurs when the ribosome encounters a stop codon, releasing the completed protein (Dever and Green, 2012).

The TIS is a specific codon in the mRNA sequence where the ribosome begins translation. The selection of the TIS is crucial as it establishes the correct open reading frame for mRNA decoding, ensuring that the protein is synthesized correctly. The most common TIS in eukaryotes is the AUG codon, which codes for the amino acid methionine(Jackson et al., 2010). However, eukaryote translation does not always start at the initial AUG codon. Thus, context information plays a role, making prediction of TIS non-trivial.

Accurate identification of TIS codons is crucial for understanding gene

expression and protein synthesis. Automating this process, particularly with machine learning and deep learning techniques, leads to many potential benefits, including enhanced gene annotation, improved understanding of diseases and therapeutics, and large-scale, cost-effective processing (Gleason et al., 2022; Kozak, 2002; Hyatt et al., 2010; Hyatt et al., 2012). Hence, automated TIS prediction is a valuable tool in modern genomic research.

5.2 Methodology

This section describes the methodology used to develop and evaluate our models. We start with a detailed discussion of the datasets. This is followed by a description of the positive-definite kernel function we have used as a baseline. We conclude the section with a description of our computational setup.

5.2.1 The Datasets

Synthetic String

Many text compression algorithms are either statistical or dictionary based. A dictionary based compressor works by substituting common fragments of text for keys (of smaller size), according to a given dictionary. In comparison, a statistical based compressor works by developing a statistical model of the text. Most consist of two stages, a modelling stage and a coding stage. During the modelling stage, the model assigns a probability distribution over the next symbol, given the previously processed text. During the coding stage,

a coder computes a codeword (a binary number) for the next symbol. The length of the codeword is related to the assigned distribution, with higher probability elements corresponding to shorter codewords (Salomon, 2007).

The prediction by partial matching (PPM) compression algorithm (Cleary and Witten, 1984) is a statistical compression algorithm whose modelling stage assigns probability distributions based on a number of Markov models of varying order. With this in mind, we generated a number of synthetic classification datasets to test our compression kernel (using PPM). We created these datasets in the hope that they would be well suited to our compression kernel yet may be more difficult for the locality-improved kernel. Each dataset consisted of strings distributed according to two k-order Markov models (see Algorithm 1 for details) i.e. the positive and negative instances followed different distributions. We created four datasets corresponding to values of $k \in \{1, 3, 5, 7\}$. Each dataset consisted of 500 positive instances and 500 negative instances, in which each instance was a string of length 300 belonging to the alphabet {"A", "T", "C", "G"}.

Translation Initiation Sites

In order to develop TIS classifiers, we utilise a dataset of eukaryotic mRNA sequences from the *Arabidopsis thaliana* plant. This dataset, which has been widely studied over the years (Pedersen and Nielsen, 1997; Zien et al., 2000), consists of 514 sequences, each annotated with its corresponding TIS codon. In its raw form, the dataset is unsuitable for the classification task at hand. However, Pedersen and Nielsen (1997) describe a method to transform it into

Algorithm 1 An algorithm to generate string data according to a k-order markov model.

Input: Σ, an alphabet. N, number of instances to generate. L, the length of each instance. k, the order of the markov model. $\alpha \in \mathbb{R}_{>0}^{|\Sigma|}$, parameters of the Dirichlet distribution.
Output: $\mathcal{D}$, the data set of strings.
Set $\mathcal{D} = \varnothing$.
for $s \in \Sigma^k$ **do**
  Sample $p \sim \mathrm{Dir}(\alpha)$.
  Set $P(\Sigma|s) = p$.
end for
while $|\mathcal{D}| < N$ **do**
  Choose uniformly at random $z \in \Sigma^k$. Set $x = z$.
  while $|x| < L$ **do**
    Sample $l \sim P(\Sigma|x_{(|x|-k)} \dots x_{|x|})$
    $x = xl$
  end while
  $\mathcal{D} = \mathcal{D} \cup \{x\}$
end while

a classification data set. The process converts a sequence into a set of subsequences, each of which is assigned a binary label indicating whether or not the subsequence contains the actual TIS codon. The subsequences are extracted from the original sequence as a length l window centred at every potential TIS codon. Algorithm 2 provides pseudocode for this procedure. When transformed using Algorithm 2, the dataset contains 2021 instances. The class ratio, i.e., the ratio of the number of TIS sequence fragments to non-TIS sequence fragments, is 0.242.

5.2.2 Kernel Functions

This section briefly reviews the locality-improved kernel, a positive-definite kernel function which we have used as a baseline to compare against the SCK from Section 4.3. This section uses the same notation and terminology

Algorithm 2 Pseudocode describing the transformation of the TIS data sets.

Input: D, a data set of mRNA sequences. l, a window size.
Output: D_t, the transformed data set.
Set $D_t = \varnothing$
for $x \in D$ **do**
  Set $I_x = \{i : x[i : i+2] = \text{'ATG'}\}$.
  for $i \in I_x$ **do**
    Set $x' = x[i-l : i+2+l]$
    if i is the TIS index **then**
      Set $y = 1$
    else
      Set $y = 0$
    end if
    $D_t = D_t \cup \{(x', y)\}$
  end for
end for

as Section 4.3. For ease of exposition, we repeat it below:

Let Σ be a finite alphabet, $\Sigma^n \subseteq \Sigma$ be the set of all strings of length n from Σ and Σ^* the set of all strings from that alphabet. A string $s \in \Sigma^*$ of length $|s|$ is a sequence of characters that can be indexed as $s = s_1 \dots s_{|s|}$.

Locality-Improved kernel The locality-improved kernel $k_{LI} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ is a positive-definite string kernel, first published by Zien et al. (2000). It emphasises local correlations by analysing fixed length substrings of the original strings. To define it, we first introduce the matching kernel. For two characters $c_1, c_2 \in \Sigma$, define the matching kernel $k_\delta : \Sigma \times \Sigma \to \{0, 1\}$

$$k_\delta(c_1, c_2) = \begin{cases} 1, & \text{if} \quad c_1 = c_2. \\ 0, & \text{otherwise.} \end{cases}, \tag{5.1}$$

Let $s, s' \in \Sigma^n$ be two strings, $l \in \mathbb{N}$ be the substring length parameter and $d_1 \in \mathbb{N}$ the substring polynomial power. The substring polynomial kernel is defined as

$$k_i(s, s') = \left(\sum_{j=-l}^{l} k_\delta(s_{i+j}, s'_{i+j}) \right)^{d_1}. \tag{5.2}$$

The locality-improved kernel, given below, is then another polynomial kernel of power $d_2 \in \mathbb{N}$ computed on all substring polynomial kernels.

$$k_{LI}(s, s') = \left(\sum_{i=l}^{n-l} k_i(s, s') \right)^{d_2}. \tag{5.3}$$

5.2.3 Computational Setup

In our experiments, we performed nested-cross validation with 10 inner and 10 outer folds. We ensured consistency in the results by using the same outer cross-validation splits across all models. We followed the scheme outlined in Algorithm 2 to create a usable classification data set, using a fixed window size of 203. In order to avoid data-leakage in the TIS experiments i.e., the process in which unwarranted information is shared between train and test sets, we defined our cross-validation splits on the original mRNA sequences and transformed these when required. We used the continuous optimisation scheme described by Chapelle et al. (2002) to select the best combination of regularisation parameters for the KSVM algorithm. The three hyperparameters of the locality-improved kernel were optimised with grid search. Values of 1 to 5 were tested for the d_1 and d_2 parameters, whilst values of 1 to 7 were tested for the l parameter. The `zlib`, `snappy`, and `PPM` (Gailly and Adler, 2020;

Google, 2020; Cleary and Witten, 1984) compression functions were used in conjunction with the SCK. Where applicable, the compression level of a compression function was optimised using grid search.

5.3 Experimental Evaluation

This section presents our experimental evaluation of the Kreĭn-SVM equipped with the SCK. We first discuss the results on the synthetic string dataset, in which we assess the efficacy of the SCK, equipped with PPM compression function, in classifying strings distributed accoridng to a k-order Markov Model. This is followed by results on the TIS dataset, in which we evaluate the effectiveness of the SCK in identifying TIS codons.

5.3.1 Synthetic string dataset

In our first set of experiments, we used the SCK (with PPM compression function) and locality-improved kernel to perform classification on our synthetic datasets. The results from this set of experiments can be seen in Table 5.1. For $k \in \{1, 3\}$, there is no difference between between performance of the models. For $k = 5$, KSVM-PPM is still perfectly discriminating between instances whilst SVM-LI suffers a small loss in performance. The results for $k = 7$ are the most noticeable, with SVM-LI suffering from a large degradation in performance compared to lesser values of k. On the other hand, KSVM-PPM still performs very well. These results indicate that the

locality-improved kernel fails to accurately discriminate between instances distributed according to a k-order Markov models when $k = 7$.

k	Model	Train AUC	Test AUC	Train Accuracy	Test Accuracy
1	SVM-LI	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)	0.991 (0.007)
	KSVM-PPM	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)
3	SVM-LI	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)	0.998 (0.004)
	KSVM-PPM	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)
5	SVM-LI	1.0 (0.000)	0.941 (0.006)	1.0 (0.000)	0.866 (0.013)
	KSVM-PPM	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)
7	SVM-LI	1.0 (0.000)	0.666 (0.017)	1.0 (0.000)	0.623 (0.013)
	KSVM-PPM	1.0 (0.000)	0.982 (0.005)	1.0 (0.000)	0.925 (0.018)

TABLE 5.1: The average accuracy and average area under the ROC curve (AUC) on the synthetic string dataset for varying values of k. KSVM-PPM denotes the Kreĭn squared-hinge SVM algorithm with PPM compression kernel, SVM-LI denotes the SVM with locality-improved kernel (standard deviations are given in brackets).

Arabidopsis

In our second set of experiments, we used the string compression and locality-improved kernels to perform classification on the *Arabidopsis* mRNA data set. The results for this set of experiments, as well as a baseline SVM using the locality-improved kernel, can be seen in Table 5.2. We chose this as a baseline as it produces an accurate classifier on similar data sets; for example, Zien et al. (2000) report an overall error rate of 11.9% on a similar data set consisting of vertebrate sequences. Inspecting Table 5.2, in terms of accuracy, it is clear that the baseline provides superior performance over the proposed string compression kernels. However, the ability of a classifier to discriminate between instances is better represented by the AUC. Our results indicate that

the Kreĭn-SVM models outperform the baseline in this respect. Inspecting only the Kreĭn-SVM models, it is clear that the `PPM` compression algorithm provides the best results. The `snappy` compression algorithm is designed to sacrifice compression rates for speed, and the `zlib` compression algorithm is designed as an all-round compression algorithm for any data type. In comparison, `PPM` is designed for compression of text data, and achieves high compression rates across a number test data sets. With this in mind, we attribute the superior performance of `PPM` to the fact that it is designed for textual data.

MODEL	ACCURACY	AUC
SVM-LI	**0.875** (0.022)	0.821 (0.027)
KSVM-PPM	0.731 (0.004)	**0.887** (0.004)
KSVM-ZLIB	0.678 (0.003)	0.867 (0.005)
KSVM-SNAPPY	0.707 (0.004)	0.875 (0.007)

TABLE 5.2: The average accuracy and average area under the ROC curve (AUC) for a number of models. KSVM-N denotes the Kreĭn squared-hinge SVM algorithm with string compression kernel, where N is the compression function. SVM-LI denotes the SVM with locality-improved kernel (standard deviations are given in brackets).

5.4 Discussion

This chapter has proposed and evaluated a methodology to perform TIS identification. More specifically, we have used the Kreĭn-SVM and SCK from Chapter 4 to develop models capable of identifying the location of TIS codons. We initially validated our methodology on a synthetic dataset of strings, distributed according to a k-order Markov model. This particular distribution is well-suited to the PPM compression function since, internally, it uses a

k-order Markov model to encode the sequence of characters. Hence, we suspected that the proposed methodology, using the PPM compression function, would excel in this setting. Indeed, our experimental evaluation confirms this. What's more, the baseline model, which was specifically designed for the task of TIS identification, exhibits a large degradation in performance as k increases. This indicates that, perhaps, the TIS data cannot be fully described by a k-order Markov model. This hypothesis is further reinforced by our experiments on the real-world TIS data. In this case, the baseline is the most accurate model. Comparing only against the SCK models, PPM clearly leads to the best performing model. Whilst TIS data cannot be fully described via a k-order Markov model, it still serves as a good proxy for the compression function.

The results of this chapter are significant to the fields of machine learning and bioinformatics. We have presented a methodology for classifying sequence data using readily available compression software. This is greatly beneficial since we suspect our methodology will translate to other forms of sequence data. It could serve as a benchmark to rapidly prototype sequence classification models, including biological sequences. Although our models were less accurate than the baseline in identifying TIS codons, we have located several starting points for further work. Firstly, the observation that our methodology excels in classifying sequences distributed according to a k-order Markov model should be explored further. Ideally, a practitioner

wanting to develop a sequence classification model could test for the Markovian property ahead of time, e.g. using the framework of hypothesis testing (Pethel and Hahs, 2014). A significant result would indicate that our methodology is warranted and likely applicable. Another avenue for further work would be to evaluate compression functions explicitly designed for biological sequence data (Rajarajeswari and Apparao, 2011; Deorowicz, 2020; Pratas et al., 2020). Our evaluation of general-purpose compression functions looked promising, so it is natural to consider whether mRNA-specific compression functions may improve model performance. Finally, our experiments only considered a single kernel function. However, combining multiple potential kernel functions is common, e.g., through their product, sum or linear combination (Gönen and Alpaydın, 2011). It is often the case that combinations of kernel functions result in more accurate models than those of their constituent components. Indeed, when used with the SCK, each compression function implicitly defines a different data representation. It is worth considering combining them, as combining the implicit representations could lead to a more robust representation. As with all kernel methods, the resulting model is heavily influenced by the choice of kernel function. Hence, a more robust representation could lead to increased modelling performance.

Chapter 6

Prediction of C-N Cross Coupling Reaction Yield

This chapter proposes and evaluates a methodology to predict the yield across a set of Buchwald-Hartwig C-N cross-coupling reactions. We assess the effectiveness of our approach in two settings. The first helps us understand how our models perform on known molecules. More specifically, the models predict the yield of reactions consisting of molecules present in the training dataset. The results in this setting are promising, with our approach performing very similarly to the baseline. The second setting is an out-of-sample evaluation in which certain additives are removed entirely from the training set, and we aim to understand how our models generalise to unknown areas of chemical space. The results in this setting demonstrate that both our approach and the baseline struggle to generalise. The rest of the chapter is organised as follows: First, we provide a background on the Buchwald-Hartwig reaction and machine learning for reaction prediction. This is followed by a detailed description of our methodology, in which we discuss the

dataset, learning algorithm and kernel functions used in our experiments, and our computational setup. Finally, we discuss the results of our experiments and propose some avenues for further work.

6.1 Background

The Buchwald-Hartwig amination is a pivotal cross-coupling reaction, catalyzed by palladium complexes, that facilitates the formation of C-N bonds between aryl halides and amines. Since its inception in the mid-1990s by Stephen L. Buchwald and John F. Hartwig (Guram and Buchwald, 1994; Louie and Hartwig, 1995), this reaction has evolved significantly, becoming a cornerstone in academic and industrial chemistry for the synthesis of aryl amines. The reaction's versatility and efficiency have made it a preferred method for the synthesis of pharmaceuticals, agrochemicals, and materials science applications (Dorel et al., 2019; Forero-Cortés and Haydl, 2019).

The reaction mechanism typically involves the oxidative addition of an aryl halide to a palladium(0) complex, followed by the coordination and deprotonation of the amine, and finally, reductive elimination to form the C-N bond. Over the years, the scope of the Buchwald-Hartwig amination has expanded through the development of various ligands and catalytic systems, enhancing its efficiency and substrate compatibility (Dorel et al., 2019). However, it can still present challenges when applied to complex drug-like molecules (Kutchukian et al., 2016). One such challenge is that traditional solvents like toluene, xylene, and 1,4-dioxane are commonly used, but they

pose environmental and safety concerns. Recent advancements have explored the use of water and solvent-free conditions to address these issues. These greener alternatives not only reduce the environmental impact but also often lead to improved reaction efficiencies and yields (Wagner et al., 2014; Fleckenstein and Plenio, 2007; Topchiy et al., 2014).

Automatically predicting the yield of chemical reactions, including Buchwald-Hartwig aminations, is a critical task in synthetic chemistry. Machine learning models have shown promise in this area by leveraging large datasets to predict reaction outcomes (Zhao et al., 2021; Kwon et al., 2022; Ahneman et al., 2018; Haywood et al., 2021). Accurate quantification of reaction yield poses many potential benefits, including, for example, the streamlining of laboratory experiments and prior optimisation of reaction conditions. However, care must be taken to ensure a generalisable and robust model. The chemistry of Buchwald-Hartwig aminations can be difficult (or low yielding) on compounds featuring chemical moieties typically used in medicinal chemistry. Hence, chemical knowledge plays an important role in the synthesis of drug-like molecules and must be incorporated when learning chemical relationships.

6.2 Methodology

This section describes the methodology used to develop and evaluate our models. We start by discussing the dataset. This is followed by a brief description of the learning algorithm and positive-definite and indefinite kernel

functions we have used. We conclude the section with a description of our computational setup.

6.2.1 The Dataset

The dataset used in this chapter, curated by Ahneman et al. (2018), contains 3955 C-N cross coupling chemical reactions. Each instance is an example of a Buchwald-Hartwig C-N cross coupling reaction of the form $\big($additive, ligand, aryl halide, base, yield$\big)$, and is a unique combination of the four chemical species in the reaction. In particular, the reactions varied in 23 isoxazole additives, 15 aryl/heteroaryl halides, 3 bases, and 4 Buchwald ligands. Ahneman et al. perform both in-sample and out-of-sample experiments. The in-sample uses a random 70/30 split of the whole data set, whereas the out-of-sample experiment completely removes additives 16 - 23 from the training set. We use the same out-of-sample partition in our experiments.

6.2.2 The Algorithm

This section briefly discusses the learning algorithm we have used to predict reaction yield. It is a Kreĭn regression algorithm, first proposed by Oglic and Gärtner (2018), in which the variance of the solution is constrained to an *a priori* specified value. The algorithm is called the Variance-Constrained Least Squares Method (VCLSM).

Given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$, RRM in a RKKS $\mathcal{K}$ with the squared-loss leads to a convex optimisation problem of the form

$$\underset{f\in\mathcal{K}}{\text{minimise}} \quad \frac{1}{m}\sum_{i=1}^{m}(f(\mathbf{x}_i)-y_i)^2+\lambda_+\|f_+\|^2_{\mathcal{H}_+}+\lambda_-\|f_-\|^2_{\mathcal{H}_-}. \tag{6.1}$$

As mentioned, we require the variance of f to be equal to a prespecified parameter $r^2 \in \mathbb{R}$. Hence, we require that

$$\frac{1}{m}\sum_{i=1}^{m}\left(f(\mathbf{x}_i)-\frac{1}{m}\sum_{j=1}^{m}f(\mathbf{x}_j)\right)^2=r^2. \tag{6.2}$$

Combining this with Equation (6.1) leads to the VCLSM problem, given in Equation (6.3).

$$\begin{aligned}\underset{f\in\mathcal{K}}{\text{minimise}} \quad & \frac{1}{m}\sum_{i=1}^{m}(f(\mathbf{x}_i)-y_i)^2+\lambda_+\|f_+\|^2_{\mathcal{H}_+}+\lambda_-\|f_-\|^2_{\mathcal{H}_-}\\ \text{subject to} \quad & \frac{1}{m}\sum_{i=1}^{m}\left(f(\mathbf{x}_i)-\frac{1}{m}\sum_{j=1}^{m}f(\mathbf{x}_j)\right)^2=r^2.\end{aligned} \tag{6.3}$$

Similarly to the Kreĭn-SVM, the representer theorem can be applied to the VCLSM problem (Oglic and Gärtner, 2018). Hence, it can be expressed as the following finite-dimensional minimisation problem.

$$\begin{aligned}\underset{\boldsymbol{\alpha}}{\text{minimise}} \quad & \frac{1}{m}\|\,\mathbf{K}\,\boldsymbol{\alpha}-\mathbf{y}\,\|_2^2+m\,\boldsymbol{\alpha}^T\,\mathbf{H}\,\mathbf{K}\,\boldsymbol{\alpha}\\ \text{subject to} \quad & \boldsymbol{\alpha}^T\,\mathbf{K}^2\,\boldsymbol{\alpha}=mr^2.\end{aligned} \tag{6.4}$$

Here, $\mathbf{y} \in \mathbb{R}^m$ is a vector whose ith element is equal to y_i. Equation (6.4) is a non-convex minimisation problem that minimises a quadratic form over a hyperellipsoid of radius r. The solution to which is quite involved and beyond the scope of this thesis. We refer the interested reader to (Oglic and

Gärtner, 2018), in which the authors present an algorithm to find a globally optimal solution.

6.2.3 Kernel Functions

This section discusses the two kernel functions used in our study. As each reaction is of the form (additive, ligand, aryl halide, base), we computed kernels for reactions in the following manner. Let a_i, l_i, h_i, b_i represent the additive, ligand, aryl halide and base present in the ith reaction, respectively. Then, for two reactions $r_1 = (a_1, l_1, h_1, b_1)$, $r_2 = (a_2, l_2, h_2, b_2)$ and a base kernel $k(x, x')$, the reaction kernel $k_{\text{rxn}}(r_1, r_2)$ is defined as

$$k_{\text{rxn}}(r_1, r_2) = k(a_1, a_2)k(l_1, l_2)k(h_1, h_2)k(b_1, b_2), \tag{6.5}$$

and is equivalent to the Hadamard product of the respective kernel matrices. For the remainder of this section, when referring to a graph kernel $k(x, x')$, we mean the reaction kernel with $k(x, x')$ as the base kernel. Before moving forward with the definitions of the kernel functions, we clarify our terminology.

A graph $\mathcal{G}$ is defined as the triplet (V, E, l), where V is the set of vertices, E is the set of edges (for this work, we only consider undirected edges) and $l : V \to \Sigma$ is a labelling function that assigns labels from the alphabet Σ to the nodes of the graph. We define the neighbourhood $\mathcal{N}(v)$ of node v as the set of nodes to which v is connected by an edge, so $\mathcal{N}(v) = \{v' : (v, v') \in E\}$

Weisfeiler-Lehman Kernel The Weisfeiler-Lehman kernel (Shervashidze et al., 2011) defines a feature space embedding for extracting structural information from a graph. It has been proven to be useful in other applications (Oglic et al., 2018). Inherent to the kernel is the relabelling scheme, which iteratively assigns new labels to the nodes of the considered graphs. For a definition of this scheme, please refer to Shervashidze et al. (2011).

Given two graphs $\mathcal{G}, \mathcal{G}'$, let Σ_i be the set of shared node labels gathered from the ith iteration of the relabelling scheme (with Σ_0 the set of original node labels). Define a map $c_i : \{\mathcal{G}, \mathcal{G}'\} \times \Sigma_i \to \mathbb{N}$ such that $c_i(\mathcal{G}, \sigma_{ij})$ is the number of occurences of the letter σ_{ij} in the graph $\mathcal{G}$. Given an iteration parameter h, the Weisfeiler-Lehman kernel $k_{\mathrm{WL}}(\mathcal{G}, \mathcal{G})$ is defined as

$$k_{\mathrm{WL}}(\mathcal{G}, \mathcal{G}') = \langle \phi_{\mathrm{WL}}^{(h)}(\mathcal{G}), \phi_{\mathrm{WL}}^{(h)}(\mathcal{G}') \rangle, \tag{6.6}$$

where

$$\phi_{\mathrm{WL}}^{(h)}(\mathcal{G}) = (c_0(\mathcal{G}, \sigma_{01}), \ldots, c_0(\mathcal{G}, \sigma_{0|\sigma_0|}),$$
$$\ldots, c_h(\mathcal{G}, \sigma_{h1}), \ldots, c_h(\mathcal{G}, \sigma_{h|\sigma_h|})),$$

and

$$\phi_{\mathrm{WL}}^{(h)}(\mathcal{G}') = (c_0(\mathcal{G}', \sigma_{01}), \ldots, c_0(\mathcal{G}', \sigma_{0|\sigma_0|})$$
$$\ldots, c_h(\mathcal{G}', \sigma_{h1}), \ldots, c_h(\mathcal{G}', \sigma_{h|\sigma_h|})).$$

Optimal Molecule Assignment Kernel The optimal assignment kernel (Fröhlich et al., 2005) is an indefinite kernel (Vert, 2008) defined on structured objects that can be decomposed into parts, including, for example, graphs composed as a set of nodes and a set of edges. Let $\mathcal{X}$ be a domain of structured

objects and let $X, Z \in \mathcal{X}$ be objects from that domain such that X and Z consist of $|X|$ and $|Z|$ parts, respectively. We denote the parts of X and Z as $X = \{x_1, \ldots, x_{|X|}\}$ and $Z = \{z_1, \ldots, z_{|Z|}\}$. Furthermore, let $\mathcal{X}'$ denote the domain of the parts i.e., $x_i, z_j \in \mathcal{X}'$ for all $1 \leq i \leq |X|$ and $1 \leq j \leq |Z|$. Given a valid Hilbert kernel $k_1 : \mathcal{X}' \times \mathcal{X}' \to \mathbb{R}$ and π, a permutation of the first $|X|$ or $|Z|$ natural numbers (this will be clear from the context), the optimal assignment kernel $k_A : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined as

$$k_A(X, Z) = \begin{cases} \text{maximise}_\pi \sum_{i=1}^{|X|} k_1(x_i, z_{\pi(i)}), & \text{if} \quad |Z| \geq |X| \\ \text{maximise}_\pi \sum_{i=1}^{|Z|} k_1(x_{\pi(i)}, z_i), & \text{otherwise.} \end{cases} \tag{6.7}$$

For the case of two molecules m, m' composed of atoms i.e., $m = \{a_1, \ldots, a_{|m|}\}$ and $m' = \{a'_1, \ldots, a'_{|m|}\}$, one needs to define k_1 on the space of atoms. Fröhlich et al. (2005) do so by first defining $k_{atom} : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ and $k_{bond} : \mathcal{B} \times \mathcal{B} \to \mathbb{R}$ as kernels that compare atom and bond features, where $\mathcal{A}$ and $\mathcal{B}$ are the spaces of atoms and bonds, respectively. k_{atom} and k_{bond} are taken to be RBF kernels, defined as

$$k_{atom/bond}(x_i, x'_i) = \exp\left(\frac{\|x_i - x'_i\|^2}{2\sigma^2}\right), \tag{6.8}$$

where x_i are the features of the ith atom or bond. They also define a kernel $R_0 : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ which compares all direct neighbours of atoms a and a'. Denote by $|N(a)|$ the number of neighbours of atom a, $n_i(a) \in \mathcal{N}(a)$ the ith neighbour of atom a and $b_i(a)$ the bond connecting atom a with its ith

neighbour. Assuming $|N(a')| \geq |N(a)|$, $R_0(a, a')$ is defined as

$$R_0(a, a') = \frac{1}{|N(a')|} \underset{\pi}{\text{maximise}} \sum_{i=1}^{|N(a)|} \left(k_{atom}\big(n_i(a), n_{\pi(i)}(a')\big) k_{bond}\big(b_i(a), b_{\pi(i)}(a')\big) \right). \tag{6.9}$$

By considering neighbours at a distance l away from an atom, another kernel $R_l : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ is defined recursively as

$$R_l(a, a') = \frac{1}{|N(a)||N(a')|} \sum_{i,j} R_{l-1}(n_i(a), n_j(a')). \tag{6.10}$$

With a parameter L controlling the maximum distance to consider and a decay parameter γ which reduces the influence of neighbours that are further away, k_1 from Equation (6.7) can be defined as

$$k_1(a, a') = k_{atom}(a, a') + R_0(a, a') + \sum_{l=1}^{L} \gamma(l) R_l(a, a'), \tag{6.11}$$

where $\gamma(l) = \gamma^l$.

6.2.4 Computational Setup

In our experiments, we performed nested cross-validation across the whole dataset as well as an out-of-sample evaluation. The nested cross-validation experiments used 10 inner and 10 outer folds. During the out-of-sample experiments, we re-trained our models using ten-fold cross-validation on a specific training set, and then evaluated these models on the hold-out test set. This test set, which is the same as that of Ahneman et al. (2018), was chosen

as it contained a number of additives not present in the training set, in particular, additives 16 - 23 (Ahneman et al., 2018). We ensured consistency in the results by using the same outer cross-validation splits across all models. For both sets of experiments, all regularisation parameters of the VCLSM models were optimised using a continuous optimisation scheme described by Oglic and Gärtner (2018). All kernel hyperparameters were optimised over the inner folds using grid search. Values of 1 to 4 were tested for the L parameter of the OMA, the tested values for the σ and γ parameters were $\{0.1, 1, 10\}$ and $\{0.1, 0.5, 0.9\}$, respectively. Of the tested kernel hyperparameters, we present results for a select few combinations in order to display a range of performances, including those which achieved the largest coefficient of determination.

6.3 Experimental Evaluation

This section discusses the results of our experimental evaluation. Utilising the OMA kernel, we performed two sets of experiments on the C-N data set: *i*) nested cross-validation across the whole data set; *ii*) an out-of-sample evaluation that withheld a specific set of additives from the training set. Experiment *i*) is designed to gauge the performance of models within known chemical space, whereas experiment *ii*) is designed to evaluate how the models generalise to unseen areas of chemical space.

6.3.1 Nested Cross-Validation

The results of our nested cross-validation experiments, for both the proposed approach, as well as the baseline, are given in Table 6.1, experiment i). We report the average root mean-squared error (RMSE) and coefficient of determination (r^2). The units of the RMSE are percentages since it derives its units from those of the dependent variable; in this case, the percentage yield. The results demonstrate that both the baseline and VCLSM OMA models exhibit similar performance. The baseline achieved a slightly smaller RMSE than the best performing VCLSM OMA model, at 9.97% compared to 10%. However, the r^2 for both of these models was equal. Interestingly, the tested hyperparameters of the OMA kernel had little effect on predictive performance. Across all hyperparameter combinations, the average RMSE and r^2 are 10.2 ± 0.192 and 0.859 ± 0.007, respectively. These results demonstrate that the proposed approach, as well as the baseline, work well when operating in known chemical space. Whilst not as accurate as the random forest proposed by Ahneman et al. (2018) (which achieves an RMSE of 7.8% and an r^2 of 0.92), both models produce accurate regressors that are capable of learning the relationship between a reaction and its yield.

6.3.2 Out-of-sample Evaluation

The out-of-sample experiments were performed to assess the generalisability of the proposed models. Table 6.1, experiment ii) displays the results (averaged over all test additives) and indicates that all models performed worse

Experiment	Model	Hyperparameters	RMSE(%)	r^2
i)	VCLSM OMA	$\sigma = 0.1, L = 1, \gamma = 0.5$	10.1 (0.539)	0.862 (0.014)
	VCLSM OMA	$\sigma = 1, L = 1, \gamma = 0.9$	10.0 (0.482)	**0.865** (0.013)
	VCLSM OMA	$\sigma = 10, L = 4, \gamma = 0.1$	10.7 (2.05)	0.841 (0.071)
	SVR WL	$h = 1$	17.0 (0.927)	0.610 (0.044)
	SVR WL	$h = 7$	10.4 (0.717)	0.853 (0.023)
	SVR WL	$h = 15$	**9.97** (0.653)	**0.865** (0.019)
ii)	VCLSM OMA	$\sigma = 0.1, L = 1, \gamma = 0.5$	15.8 (1.64)	0.539 (0.144)
	VCLSM OMA	$\sigma = 1, L = 1, \gamma = 0.9$	19.4 (1.67)	0.305 (0.211)
	VCLSM OMA	$\sigma = 10, L = 4, \gamma = 0.1$	22.1 (2.1)	0.099 (0.278)
	SVR WL	$h = 1$	18.8 (1.74)	0.423 (0.114)
	SVR WL	$h = 7$	**15.7** (3.02)	**0.583** (0.177)
	SVR WL	$h = 15$	16.6 (2.99)	0.541 (0.165)

TABLE 6.1: The average RMSE and r^2 values achieved during experiments *i*) and *ii*) on the C-N data set. SVR WL denotes the (SVR) algorithm with Weisfeiler-Lehman kernel, VCLSM OMA denotes the the VCLSM algorithm using the OMA kernel (standard deviations are given in brackets). σ, L and γ are the hyperparameters of the OMA kernel and h is the hyperparameter of the Weisfeiler-Lehman kernel; see Section 5.2.2 for details.

when generalising to unseen examples. There is a clear increase in RMSE and decrease in r^2 when comparing between experiments *i*) and *ii*). Once again, the baseline SVR WL model outperforms the VCLSM OMA model. However, both models are outperformed by the random forest proposed by Ahneman et al. (2018) (which achieves an average RMSE of 11.3% and an average r^2 of 0.83). The overall inferior performance is indicative of the general difficulty statistical models encounter when generalising to unseen data. The notion of activity cliffs make this especially true when working with chemical data (Stumpfe et al., 2019). Analysis of the predictions for individual additives indicates that both models systematically performed worse on additives 18 and 19 and perform the best on additive 16. The same result is observed in the work of Ahneman et al. (2018). Figure 6.1 provides a heatmap detailing

the pairwise similarities between all additives. There are clear patterns between the set of training and test additives. Additives 1 - 16 are generally more similar to each other than they are to additives 17 - 23. Furthermore, additives 18 and 19 display high similarity to each other and low similarity to all other additives in the set. We suspect this is why both models performed worst on these additives.

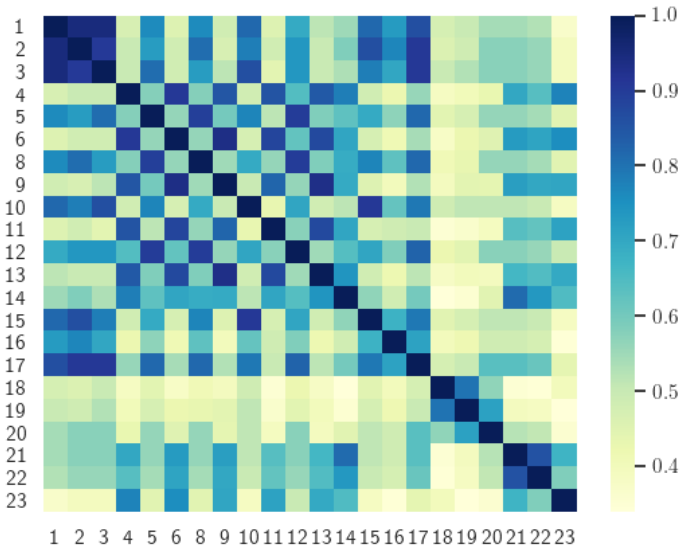


FIGURE 6.1: The pairwise similarities between all additives in the C-N data set. Similarities were calculated using the tanimoto coeffcient, with molecules represented using MACCS keys (Durant et al., 2002).

6.4 Discussion

This chapter has developed and evaluated models to predict chemical reaction yield. More specifically, we have used the VCLSM algorithm, in conjunction with the OMA kernel, to develop models that predict the yield of a set of Buchwald-Hartwig amination reactions. We initially performed a

set of nested cross-validation experiments to understand the applicability of our approach when deployed in known chemical space. Our methodology performs very similarly to the baseline in this setting, with both approaches achieving a small RMSE and large r^2. We conclude from this set of experiments that our approach is capable of high performance when operating in known chemical space. However, it is more desirable that the models generalise to unseen areas of chemical space. To test this, we followed the approach of Ahneman et al. (2018) to perform an out-of-sample evaluation. In particular, additives 16 - 23 were removed from the training set entirely, therefore simulating the setting in which an unknown molecule is presented to the model. In this setting, all models performed worse than the in-sample case. However, this is to be expected since generalising to unseen additives is much more challenging. Furthermore, observing that our proposed approach performed similarly to the baseline is encouraging. Nevertheless, more work is needed to improve the generalisability of our approach.

The results of this chapter are significant to the field of chemoinformatics. We have established that Kreĭn-based approaches using a kernel for graph-structured data performs similarly to positive-definite kernel methods using the same representation. With Kreĭn-based methods being a relatively new approach, this leads to many potential areas for future work. One approach that could directly benefit our current methodology is incorporating more information, beyond structural, into the model. Whilst they have shown promise, the kernels we have considered, both positive-definite and indefinite, operate solely on the structure of the molecules in a reaction. On the

other hand, the model of Ahneman et al. (2018) uses the random forest algorithm in conjunction with atomic, molecular and vibrational features. It would be a relatively straightforward task to extend the OMA kernel to account for more molecular information, e.g. via the sum of itself and a Gaussian kernel defined over the molecular features. We suspect that the superior generalisability of the model of Ahneman et al. is likely due to the information contained within the features. Hence, incorporating these features into the OMA kernel could improve the generalisability of our approach. It should be noted, however, that many of the features used by Ahneman et al. are quantum chemical and, therefore, time-consuming and computationally demanding to obtain.

Our approach could also benefit from a method to constrain the predictions. In our experiments, we noticed that some yield predictions were outside the range of admissible values (less than 0% or greater than 100%). Imposing upon the models the prior knowledge that the yield is constrained to lie between 0% and 100% would remove dubious predictions and improve model performance. This would, however, require a modification of the learning algorithm, which is much more involved than our previous recommendation of incorporating chemical features into the model. A possible avenue to accomplish this would be a Kreĭn space extension of beta regression. Beta regression, which falls under the family of generalised linear models, is a statistical technique used to predict proportions, i.e. values lying in the interval $(0, 1)$ (Ferrari and Cribari-Neto, 2004; Agresti, 2015). The standard formulation assumes a linear relation between the dependent and

independent variables. However, similarly to Section 4.2, one could instead assume the relationship is described as a function within an RKKS and derive a learning algorithm from this perspective.

Chapter 7

Classification of Antimicrobial Peptides

This chapter proposes and evaluates a methodology to classify antimicrobial peptides. We initially assess the effectiveness of our method in identifying peptides exhibiting general antimicrobial activity. That is, those which are active against any microbe. A detailed experimental evaluation highlights the promise of our approach in this setting. We further tested our methodology in experiments to identify species-specific activity. In this case, our models fell short of one of the in-house baselines, outperforming the other in-house baseline and a preexisting web-tool. Additionally, the general and species-specific models are made freely available as web applications at http://comp.chem.nottingham.ac.uk/KreinAMP. The rest of the chapter is organised as follows: First, we provide a background on AMPs and their automated identification. This is followed by a detailed description of our methodology, in which we discuss the datasets and kernel functions used in our experiments and our computational setup. Finally, we discuss the results

of our experiments and propose some avenues for further work.

7.1 Background

AMPs, also known as host defense peptides, are a class of evolutionary conserved molecules that form an important component in the innate immune system (Mookherjee et al., 2020; Hancock et al., 2016; Ting et al., 2022). These molecules are usually made of 12 to 50 amino acid residues, and they typically possess certain properties, including cationicity, 30-50% hydrophobicity, and amphiphilicity. They exhibit good antimicrobial activity against a broad range of bacteria, viruses, fungi, and parasites. In addition, they have an inherent low risk of developing antimicrobial resistance (AMR), largely attributed to their underlying rapid membrane permeabilising activity (Mookherjee et al., 2020; Ting et al., 2021a; Mayandi et al., 2020). Such broad-spectrum and rapid antimicrobial activity has prompted researchers to consider AMPs as a potential remedy to the growing problem of AMR, which is a major global health threat (Murray et al., 2022; Ali et al., 2022). Nonetheless, there has so far been a lack of success in translating AMP-based therapy to clinical use, due to challenges such as complex structure-activity relationship (SAR), drug toxicity, instability in host and inefective environment, and low financial incentives (Fjell et al., 2012; Ting et al., 2020). Owing to the complex SAR and the costly and time-consuming process of wet-lab experiments associated with AMP investigations, many researchers have proposed computational approaches, including molecular dynamics (MD) simulations and

machine learning (ML) algorithms, to accelerate the discovery and development of potential AMPs for clinical use (Das et al., 2021; Yount et al., 2019; Ting et al., 2021b; Capecchi et al., 2021; Li et al., 2022; Aronica et al., 2021; Pinacho-Castellanos et al., 2021).

Several studies have highlighted the promise of ML algorithms in predicting the antimicrobial activity, dissecting the complex SAR, and informing the drug design of AMPs (Das et al., 2021; Yount et al., 2019; Ting et al., 2021b). A wide range of ML algorithms have been utilised, including random forests (Thomas et al., 2010), support vector machines (SVMs) (Thomas et al., 2010; Lata et al., 2010; Torrent et al., 2011; Lee et al., 2016; Meher et al., 2017) and artificial neural networks (Lata et al., 2010; Thomas et al., 2010; Torrent et al., 2011; Yan et al., 2020; Veltri et al., 2018). Many of these algorithms are used in combination with a carefully selected set of peptide features, which can be divided into two categories: compositional and physicochemical. The amino acid composition is the simplest example of a compositional feature, which is a vector containing counts of each amino acid in a given peptide. There are various extensions, such as the reduced amino acid composition (Feng et al., 2013) and the pseudo amino acid composition (Chou, 2001). When computing the reduced amino acid composition, a peptide is represented in a reduced alphabet in which similar amino acids are grouped together. The pseudo amino acid composition accounts for composition as well as sequence-order information, as this is not considered in the standard amino acid composition. The set of physicochemical features include peptide properties such as the charge, hydrophobicity and isoelectric point (Rose et

al., 1985; Thomas et al., 2010; Meher et al., 2017). These features are typically average values of the respective properties calculated over the length of the peptide.

7.2 Methodology

This section describes the methodology used to produce our AMP classifiers. We start with a discussion of the datasets, distinguishing those used to develop general AMP classifiers and those used to develop species-specific AMP classifiers. This is followed by a detailed description of the sequence alignment measures we have used as kernel functions in the Kreĭn-SVM algorithm. We conclude the section with a description of our computational setup.

7.2.1 The Datasets

General Antimicrobial Datasets

We selected two AMP classification data sets from the literature, which we refer to as AMPScan (Veltri et al., 2018) and DeepAMP (Yan et al., 2020), in order to test the ability of approach to predict general antimicrobial activity. Detailed discussions on the creation of these data sets can be found in the original studies. The AMPScan and DeepAMP data sets contain 3556 and 3246 instances, respectively. Each data set also contains a 50:50 ratio of AMPs to non-AMPs, allowing us to avoid issues that result from class imbalance. Associated to each data set is a specific test set, and reporting results

on this set allows comparison with the authors' proposed models. Despite being of similar size, one major differentiating factor between the two data sets is the length of peptides. Figure 7.1 displays the empirical distribution of peptide lengths for both data sets, partitioned by peptide classification. In both cases, the distributions corresponding to AMPs and non-AMPs are very similar. However, the distributions across data sets are clearly very different. The Kolmogorov-Smirnov two-sample test (Hodges Jr, 1958) provides evidence to reject the null hypothesis that the peptide length distributions of the AMPScan and DeepAMP data sets are identical. DeepAMP contains generally shorter peptides than AMPScan. Indeed, the DeepAMP data set was curated since short-length AMPs have been shown to exhibit enhanced activity, lower toxicity and higher stability as opposed to their longer counterparts (Kim et al., 2014; Ramesh et al., 2016). More importantly, synthesis is cheaper for the short AMPs than the full-length AMPs, which increases the potential for clinical translation and commercialisation (Ting et al., 2020). Figure 7.2 displays the empirical amino acid distributions for both data sets, indicating their similarity.

Species-Specific Training Datasets

Vishnepolsky et al. (2018) have shown that, given an appropriate training data set, predictive models of peptide activity against specific species can be constructed. This involves training a separate model for each species of interest. Their model, predicting activity against *E. coli* ATCC 25922, achieved a balanced accuracy of 0.79, which was greater than a number of common

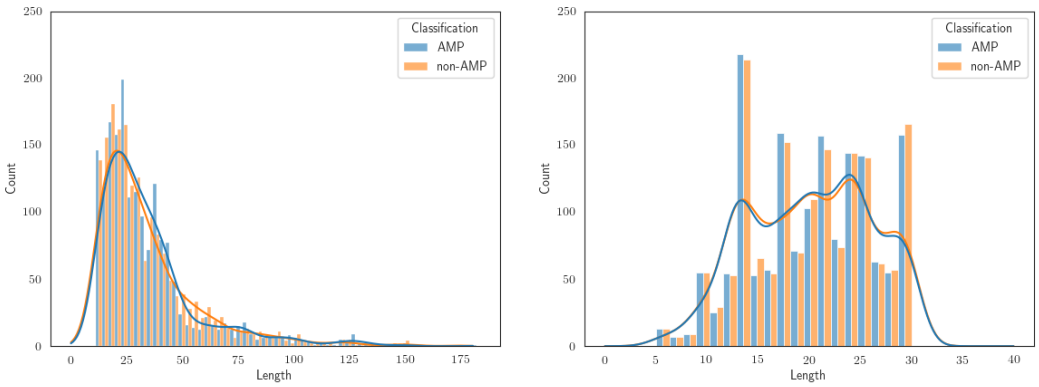


FIGURE 7.1: The distribution of peptide lengths for (a) the AMPScan and (b) DeepAMP data sets.

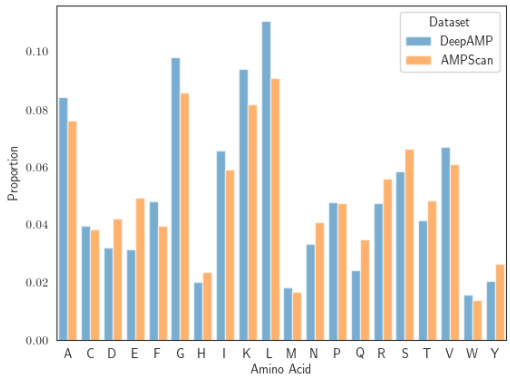


FIGURE 7.2: The distribution of amino acids for the AMPScan and DeepAMP data sets.

AMP prediction tools (Vishnepolsky et al., 2018). Furthermore, models to predict activity against *S. aureus* ATCC 25923 and *P. aeruginosa* ATCC 27853 were made publicly available as web-tools.

We follow the methodology of Vishnepolsky et al. to construct useful training data sets for our problem. We utilised the Database of Antimicrobial Activity and Structure of Peptides (DBAASP) as a source of data. DBAASP contains peptide activity measurements against a wide-range of species (Pirtskhalava et al., 2021), including those of interest to us. We extracted from DBAASP all peptides with activity measured against *S. aureus* ATCC 29213,

S. aureus ATCC 25923 or *P. aeruginosa* ATCC 27853 subject to the following conditions: *i*) peptide length in the range [6, 18], *ii*) without intrachain bonds, *iii*) without non-standard amino acids and *iv*) MIC measured in μg/mL or μM. Condition *i*) was imposed as that is the range of peptide lengths in our external test set. Conditions *ii*) and *iii*) were imposed following the recommendation of the Vishnepolsky et al. and condition *iv*) was imposed as conversion from μM to μg/mL is possible by estimating the molecular weight of a given sequence. Since no web-tool to predict activity against *S. aureus* ATCC 29213 was available, we couldn't directly compare our results. Instead, we collected data for peptides active against *S. aureus* ATCC 25923. This allowed us to compare our models with the state of the art provided by Vishnepolsky et al.

Three separate data sets of peptides with activity measured against *S. aureus* ATCC 29213, *S. aureus* ATCC 25923 and *P. aeruginosa* ATCC 27853 were created using the data collected from DBAASP. We refer to these data sets as SA_{29213}, SA_{25923}, and PA_{27853}, respectively. The peptides in the respective data sets were labelled according to their activity against the specific strain. Each data set is constructed from highly active peptides (MIC $\leq$ 25μg/mL) and inactive peptides (MIC $\geq$ 100μg/mL). A peptide with 25μg/mL $<$ MIC $<$ 100μg/mL would not be included in our training data set. This large interval allows us to account for experimental errors, which in turn increases the confidence in our class labels. In the case that a peptide was associated to multiple activity measurements, the median value was taken to represent its activity. As shown in Table 7.1, the three training data sets are

Data set	Size	Class Ratio
SA_{29213}	463	0.644
SA_{25923}	808	0.646
PA_{27853}	686	0.547

TABLE 7.1: Descriptive statistics of the SA_{29213}, SA_{25923} and PA_{27853} data sets

all relatively small and contain slightly more active peptides than inactive peptides.

Species-Specific Testing Datasets

A previously established data set of 16 short-length peptides (18 amino acids or shorter in length), with minimum inhibitory concentration (MIC) measured against *S. aureus* ATCC 29213 and *P. aeruginosa* ATCC 27853, was used to test the ability of the developed models in predicting species-specific antimicrobial activity (Ting et al., 2021a). These peptides were commercially synthesised by Mimotopes (Mulgrave Victoria, Australia) via solid phase Fmoc synthesis and were purified by reverse phase high performance liquid chromatography (RP-HPLC) to >95% purity. The efficacy of these peptides was already experimentally validated using established MIC assay with broth microdilution method approved by the Clinical and Laboratory Standards Institute. Full detail of the previously conducted microbiological experiment can be found in the previous study (Ting et al., 2021a). To make this data set suitable for classification, we label a peptide as active if its MIC $< 100\mu g/mL$ and inactive otherwise.

7.2.2 Kernel Functions

Classical sequence alignment algorithms, such as the Smith-Waterman (Smith and Waterman, 1981) and Needleman-Wunsch (Needleman and Wunsch, 1970) algorithms, are computationally intensive and do not scale well to large problems. Many papers have advocated the use of alignment-free methods to determine sequence similarity (Zielezinski et al., 2019; Kantorovitz et al., 2007; Zielezinski et al., 2017; Kuksa and Pavlovic, 2009; Pinacho-Castellanos et al., 2021). The success of these endeavours notwithstanding, sequence alignment functions are effective notions of biological-sequence similarity that can reflect ancestral, structural or functional similarity and therefore should not be overlooked. Several studies have utilised sequence alignment functions for AMP prediction. For example, Wang et al. (2011) and Ng et al. (2015) utilised the BLAST algorithm (Altschul et al., 1990) in a classification model by comparing the BLAST similarity scores of a query peptide to all those in the training set. Whilst these approaches led to accurate models, the BLAST algorithm is a heuristic method that finds only approximate optimal alignments. This approximation leads to generally faster results than what could be obtained by the Smith-Waterman algorithm, and it is one of the main reasons practitioners choose to use it. However, on the relatively small data sets in the aforementioned studies, it is interesting to consider whether the same approaches using the optimal alignment score would improve the models. In this work, we have decided to focus only on exact alignments. Whilst heuristic alignment algorithms such as BLAST lead to generally faster results, this benefit only materialises when working with

much larger datasets than what we have considered. Furthermore, due to the sensitivity of the resulting model to the choice of kernel function, the potential loss in performance caused by approximately optimal, as opposed to optimal, alignments does not outweigh the benefit of more efficient computation.

We now proceed to define the sequence similarities used throughout this work. This section closely follows the works of Setubal and Meidanis (1997) and Yujian and Bo (2007). First, we clarify our terminology.

Notation and Terminology

Let Σ be a finite alphabet, $\Sigma^n \subseteq \Sigma$ be the set of all strings of length n from Σ and Σ^* the set of all strings from that alphabet. A string $s \in \Sigma^*$ of length n is a sequence of characters that can be indexed as $s = s_1 \dots s_n$. Given a string $s \in \Sigma^n$, we say that $u \in \Sigma^m$ is a subsequence of s if there exists a set of indices $I = \{i_1, \dots, i_m\}$ with $1 \leq i_1 \leq \dots \leq i_m \leq n$, such that $u_j = s_{i_j}$ for $j = 1, \dots, m$. We write $u = s[I]$ for short. We say that $v \in \Sigma^l$ is a substring of s if v is a subsequence of s with index set $J = \{j_1, \dots, j_l\}$ such that $j_{r+1} = j_r + 1$ for $r = 1, \dots, l-1$. That is, v is a subsequence consisting of consecutive characters of s.

Global Alignments

The goal of a sequence alignment is to establish a correspondence between the characters in two sequences. In the context of bioinformatics, a pairwise alignment can indicate ancestral, structural or functional similarities between

the pair of sequences. In this section, we provide a formal review of global sequence alignment.

Definition 7.2.1 (Global Alignment)**.** Let Σ be an alphabet and let $s \in \Sigma^n$ and $t \in \Sigma^m$ be two strings over Σ. Define $\Sigma_g = \Sigma \cup \{" - "\}$ as the extension of Σ with the gap character " $-$ ". The tuple $\alpha(s,t) = (s',t')$ is a global alignment of sequences s and t if and only if

1. $s', t' \in \Sigma_g^*$
2. $|s'| = |t'| = l$, such that $\max(n,m) \leq l \leq m+n$,
3. The subsequence of s' obtained by removing all gap characters is equal to s,
4. The subsequence of t' obtained by removing all gap characters is equal to t,
5. $\{i|s'_i = " - "\} \cap \{i|t'_i = " - "\} = \varnothing$.

Definition 7.2.1 provides a formal definition of global alignment. Whilst many possible alignments exist for two strings, the goal of sequence alignment is to find an alignment that optimises some criterion. A scoring function, presented in Definition 7.2.2, can be used to quantify the "appropriateness" of an alignment. An optimal global alignment is then one which is optimal with respect to a given scoring function, as shown in Definition 7.2.3.

Definition 7.2.2 (Scoring Functions)**.** Let Σ be an alphabet, $\Sigma_g = \Sigma \cup \{" - "\}$ be the extension of Σ with the gap character " $-$ " and $p : \Sigma_g \times \Sigma_g \to \mathbb{R}$ be

a function defined over the elements of Σ_g. We say p is a *similarity* scoring function over Σ_g if, for all $x, y \in \Sigma$, we have

1. $p(x, x) > 0$,
2. $p(x, x) > p(x, y)$,
3. $p(x, y) = p(y, x)$,
4. $p(x, \text{“}-\text{”}) \leq 0$,
5. $p(\text{“}-\text{”}, \text{“}-\text{”}) = -\infty$.

Similarly, we say p is a *distance* scoring function over Σ_g if, for all $x, y, z \in \Sigma$, we have

1. $p(x, x) = 0$,
2. $p(x, y) > 0$,
3. $p(x, y) = p(y, x)$,
4. $p(x, \text{“}-\text{”}) > 0$,
5. $p(x, z) \leq p(x, y) + p(y, z)$,
6. $p(\text{“}-\text{”}, \text{“}-\text{”}) = \infty$.

Definition 7.2.3 (Optimal Global Alignment)**.** Let Σ be an alphabet, $\Sigma_g = \Sigma \cup \{\text{“}-\text{”}\}$ be the extension of Σ with the gap character “ $-$ ” and consider two strings $s \in \Sigma^n$, $t \in \Sigma^m$. Let $\alpha(s, t) = (s', t')$ be a valid global alignment of s and t (valid in the sense that it satisfies the conditions of Definition 7.2.1), $p : \Sigma_g \times \Sigma_g \to \mathbb{R}$ be a scoring function over Σ_g and $\mathcal{A}(s, t)$ be the space of all

valid alignments of s and t. The score $\mathcal{S}_p(\alpha(s,t))$ of $\alpha(s,t)$ with respect to the scoring function p is defined as

$$\mathcal{S}_p(\alpha(s,t)) = \sum_{i=1}^{l} p(s'_i, t'_i). \tag{7.1}$$

If p is a similarity scoring function, then the optimal global alignment $\alpha^*(s,t)$ with respect to p is defined as

$$\alpha^*(s,t) = \underset{\alpha(s,t)\in\mathcal{A}(s,t)}{\arg\max}\ \mathcal{S}_p(\alpha(s,t)). \tag{7.2}$$

Similarly, if p is a distance scoring function then the optimal global alignment $\alpha^*(s,t)$ with respect to p is defined as

$$\alpha^*(s,t) = \underset{\alpha(s,t)\in\mathcal{A}(s,t)}{\arg\min}\ \mathcal{S}_p(\alpha(s,t)). \tag{7.3}$$

Since an alignment is optimal with respect to a given scoring function, it is natural to consider which scoring function to use in order to obtain the most meaningful alignments. In the context of biological sequences, researchers have been considering this problem for many years. A number of families of scoring matrices have been designed to encode useful notions of similarity. In this work, we only considered the BLOSUM62 scoring matrix (Henikoff and Henikoff, 1992), as it is a standard choice when performing sequence alignment. However, one could consider the position-specific scoring matrix (PSSM) generated from the PSI-BLAST algorithm (Altschul et al., 1997). PSI-BLAST generates a PSSM, or profile, by scanning a protein database. The

PSSM can be used to construct a scoring matrix, which is ordinarily used for further scanning of the database. This process is repeated, wherein each iteration the PSSM is updated using the results from the previous scan. This iterative method allows for refinement of the PSSM to detect distant relationships between proteins (Bhagwat and Aravind, 2008). The resulting PSSM could be provided as input to, say, the Smith-Waterman algorithm, leading to a custom scoring matrix tailored to the distant evolutionary relationships of a given protein. Our choice of the BLOSUM62 scoring matrix is mainly due to simplicity. The generation of custom PSSMs could be very beneficial for our task, but we aim to demonstrate that accurate models can be built using only relatively simple scoring matrices.

Levenshtein Distance

The string edit distance defines a useful notion of distance between a pair of strings. It is informally defined as the minimum number of edit operations required to transform one string into another. The Levenshtein distance is a variant of the string edit distance that allows the operations of substitution, deletion and insertion of characters, and these are defined in Definition 7.2.4.

Definition 7.2.4 (Elementary Edit Operations)**.** Let Σ be an alphabet. For two characters $a, b \in \Sigma$, we denote by $a \to b$ the elementary edit operation that substitutes a with b. Denoting by ε the null character (the empty string), we can define the elementary edit operations of insertion and deletion as $\varepsilon \to b$ and $a \to \varepsilon$, respectively.

In order to define more complex transformations, one can consider the consecutive application of a sequence of elementary edit operations. Of special interest to the Levenshtein distance are those sequences of operations that transform one string into another. Such a sequence is known as an edit path and its length is defined as the number of operations in the sequence. The Levenshtein distance between two strings is defined as the length of the minimum length edit path, as seen in Definition 7.2.5.

Definition 7.2.5 (Levenshtein Distance)**.** Let Σ be an alphabet and consider two strings $s \in \Sigma^n$ and $t \in \Sigma^m$ from Σ. An edit path from s to t is denoted by $P_{s,t}$ and represents a sequence of elementary edit operations that transforms s into t. Denote by $|P_{s,t}|$ the number of operations contained in $P_{s,t}$ and by $\mathcal{P}_{s,t}$ the space of all edit paths from s to t. The Levenshtein distance $d_L(s,t)$ between s and t is defined as

$$d_L(s,t) = \min_{P_{s,t} \in \mathcal{P}_{s,t}} |P_{s,t}|. \tag{7.4}$$

Definition 7.2.5 shares some interesting similarities with Definition 7.2.3. Both problems solve a combinatorial optimisation problem and, indeed, the Levenshtein distance can be realised as a special case of global alignment. More specifically, consider the distance scoring function $p : \Sigma_g \times \Sigma_g \to \{0,1\}$ defined as

$$p(x,y) = \begin{cases} 0, & \text{if} \quad x = y \\ 1, & \text{otherwise.} \end{cases} \tag{7.5}$$

For two strings $s \in \Sigma^n$ and $t \in \Sigma^m$, let their optimal global alignment with respect to p be equal to $\alpha^*(s,t) = \{s', t'\}$. Define the set U as

$$U = \{i \mid s'_i \neq t'_i\}. \tag{7.6}$$

Then the score $\mathcal{S}_p(\alpha^*(s,t))$ of $\alpha^*(s,t)$ with respect to p is equal to the cardinality of U. This is exactly equal to the Levenshtein distance between s and t.

Local Alignments

A global alignment produces an alignment which spans the whole length of a pair of strings. It is based on the assumption that the strings are related in their entirety. This assumption can be restrictive, since it is often the case that certain substrings exhibit high similarity whilst others do not. A local alignment produces an alignment that finds those high similarity substrings. That is, it finds the highest scoring global alignment from all possible substrings of the pair of strings. We formalise this notion in Definition 7.2.6.

Definition 7.2.6 (Optimal Local Alignment)**.** Let Σ be an alphabet, $\Sigma_g = \Sigma \cup \{\text{“}-\text{”}\}$ be the extension of Σ with the gap character “$-$” and $p : \Sigma_g \times \Sigma_g \to \mathbb{R}$ be a similarity scoring function. For a string $s \in \Sigma^n$, let $\mathcal{I}_s$ be the space of all index sets such that for any $I_s \in \mathcal{I}_s$, $s[I_s]$ is a valid substring of s. Similarly, for a string $t \in \Sigma^m$, let $\mathcal{I}_t$ be the space of all index sets such that for any $I_t \in \mathcal{I}_t$, $t[I_t]$ is a valid substring of t. For any $I_s \in \mathcal{I}_s$ and $I_t \in \mathcal{I}_t$, denote by $\alpha^*(s[I_s], s[I_t])$ the optimal global alignment of $s[I_s]$ and $t[I_t]$ with respect to p.

The optimal local alignment $\alpha_L^*(s,t)$ of s and t with respect to p is defined as

$$\alpha_L^*(s,t) = \underset{I_s \in \mathcal{I}_s, I_t \in \mathcal{I}_t}{\arg\max} \mathcal{S}_p(\alpha^*(s[I_s], t[I_t])). \tag{7.7}$$

7.2.3 Computational Setup

This section discusses the setup of our computational evaluation. To assess the usage of learning with sequence alignment functions, we performed a series of computational experiments on a number of AMP data sets. In each of our evaluations, we tested both the local alignment score (LA) and the Levenshtein distance (LEV) in conjunction with the Kreĭn-SVM algorithm. We compare against two baselines: an SVM with amino acid composition kernel and an SVM using the gapped k-mer kernel. The former is a positive-definite kernel function; peptides are represented via their amino acid composition and the kernel is defined as the inner product under this representation. The latter is also a positive-definite kernel function. It has produced accurate models in a number of biological-sequence classification tasks (Wang et al., 2016; Ghandi et al., 2014; Blakely et al., 2020) and hence makes for a useful baseline. When applicable, we also compared our models with AMP identification tools from the literature. The `parasail` package (Daily, 2016) was used to compute local alignment scores. We only considered normalised variants of the local alignment score and Levenshtein distance, with the normalisation performed according to Schölkopf et al. (2004) and Yujian and Bo (2007), respectively. We report the accuracy, the area under the receiver operating characteristic curve (AUC) and Matthews correlation coefficient (MCC)

to compare models. The accuracy and MCC are defined in Equation (7.8), where TP, TN, FP and FN are the number of true-positives, true-negatives, false-positives and false-negatives, respectively.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{7.8}$$

$$\mathrm{MCC} = \frac{(\mathrm{TP} \times \mathrm{TN}) - (\mathrm{FN} \times \mathrm{FP})}{(\mathrm{TP} + \mathrm{FN}) \times (\mathrm{TN} + \mathrm{FP}) \times (\mathrm{TP} + \mathrm{FP}) \times (\mathrm{TN} + \mathrm{FN})} \tag{7.9}$$

The AUC is defined as the probability that a classifier will score a randomly chosen positive instance higher than a randomly chosen negative instance (Fawcett, 2006). In order to allow for a fair comparison, all models used the same training and test splits. The optimal hyperparameters were selected by performing an exhaustive grid search over the training set, using 10-fold cross validation. The λ hyperparameter of the SVM algorithm, as well as the λ_+ and λ_- hyperparameters of the Kreĭn-SVM algorithm, were selected from $\{0.01, 0.1, 1, 10, 100\}$. The Levenshtein distance and amino acid composition kernel have no hyperparameters to control; we used the default values for the hyperparameters of the local alignment score. The gapped k-mer kernel has two hyperparameters g and m and is quite susceptible to their values. The optimal value of g was selected from $\{1, 2, 3, 4, 5\}$ and the optimal value of m was selected from $\{1, 2, 3, \ldots, 10\}$. It is required that $g > m$, so only valid combinations of the two were considered. In our nested cross-validation experiments, we used 10 inner and 10 outer folds and the reported results are averaged over the outer fold test sets.

7.3 Results

In Section 7.3.1, we discuss the ability of our models to identify general antimicrobial activity. We observe that our proposed models consistently outperform the baselines and, in some cases, the models proposed in the literature. One shortcoming of any computational model that identifies general antimicrobial activity is that it cannot be used to identify activity against specific species. We address this shortcoming in Section 7.3.2, by training our models to identify activity against S. *aureus* ATCC 29213 and P. *aeruginosa* ATCC 27853 on an experimentally validated data set of 16 peptides, for which the proposed approach produces accurate models.

7.3.1 Identifying General Antimicrobial Activity

Nested Cross-Validation

The performance of the AMP classifiers on the considered data sets is reported in Table 7.2. The results are averaged over the multiple test sets generated by nested-cross validation. On both data sets, the proposed models achieve a greater average accuracy, AUC and MCC than the baselines, with the local alignment score achieving the best values in all cases. The Welch t-test (Welch, 1947), with $p = 0.05$, is used to compare the test set AUC of our proposed models against the baseline SVM with Gapped k-mer kernel across the outer folds of nested cross-validation. Adjusting for the testing of multiple hypotheses with the Bonferroni correction, we observe a significant difference between the mean AUC of the local alignment score and that of the

baseline SVM with Gapped k-mer kernel on the AMPScan data set. All other comparisons, including those on the DeepAMP data set, are not significant. The performance of all models is greater on the AMPScan data set than on the DeepAMP data set.

	AMPScan			DeepAMP		
Model	Accuracy	AUC	MCC	Accuracy	AUC	MCC
LA-KSVM	0.920 (0.017)	0.969 (0.006)	0.842 (0.033)	0.760 (0.025)	0.821 (0.028)	0.522 (0.051)
LEV-KSVM	0.910 (0.021)	0.966 (0.010)	0.821 (0.042)	0.756 (0.032)	0.819 (0.029)	0.513 (0.063)
GKM-SVM	0.899 (0.015)	0.957 (0.007)	0.799 (0.030)	0.751 (0.032)	0.817 (0.029)	0.506 (0.066)
AAC-SVM	0.865 (0.023)	0.930 (0.009)	0.732 (0.044)	0.723 (0.031)	0.784 (0.035)	0.447 (0.061)

TABLE 7.2: Quality of the predictions on the AMPScan and DeepAMP data sets. The average accuracy. AUC and MCC are reported (standard deviation in parentheses), computed over the outer fold test sets of the nested cross-validation procedure. Results are presented for the Kreĭn-SVM with local alignment score (LA-KSVM), the Kreĭn-SVM with Levenshtein distance (LEV-KSVM), the SVM with Gapped k-mer kernel (GKM-SVM) and the SVM with amino acid composition kernel (AAC-SVM)

Predefined Test Set

Table 7.3 reports the results of all models on the predefined test set associated with each data set. For the sake of completeness, we also include the performance of the neural network-based classifiers proposed by the authors of each data set. As for the nested cross-validation results, we observe that all models perform better on the AMPScan data set than the DeepAMP data set. Considering the former, the local alignment score achieves the largest accuracy, AUC and MCC, and is followed closely by the literature model. On the DeepAMP data set, the performance is similar among all methods. The local alignment score achieves the best AUC but the Levenshtein distance achieves the best accuracy and MCC. However, the Levenshtein distance outperforms

the literature model against all metrics. On both data sets, the baselines are the least predictive models. It is encouraging to observe that the sequence alignment functions can produce classifiers that match, and also outperform, the neural network-based classifiers.

	AMPScan			DeepAMP		
Model	Accuracy	AUC	MCC	Accuracy	AUC	MCC
LA-KSVM	0.911	0.967	0.823	0.761	0.863	0.523
LEV-KSVM	0.904	0.960	0.809	0.798	0.860	0.596
GKM-SVM	0.900	0.954	0.801	0.782	0.838	0.564
AAC-SVM	0.870	0.929	0.742	0.771	0.853	0.543
Literature	0.910	0.965	0.820	0.771	0.853	0.543

TABLE 7.3: Quality of the predictions on the AMPScan and DeepAMP data sets. The accuracy, AUC and MCC are reported, computed on the predefined test sets. Results are presented for the Kreĭn-SVM with local alignment score (LA-KSVM), the Kreĭn-SVM with Levenshtein distance (LEV-KSVM), the SVM with Gapped *k*-mer kernel (GKM-SVM) and the SVM with amino acid composition kernel (AAC-SVM). Results from the respective neural network-based classifiers(Veltri et al., 2018; Yan et al., 2020) proposed by the authors of each data set are also presented, denoted by Literature

7.3.2 Identifying Species-Specific Activity

In this section, we highlight the ability of our models to identify AMPs that are active against specific species, particularly S. *aureus* ATCC 29213 and P. *aeruginosa* ATCC 27853. Table 7.4 displays the accuracy on the external test set for models trained on the AMPScan and DeepAMP data sets. Once again, we also include the performance of the neural-network-based classifiers proposed by the authors of the AMPScan and DeepAMP data sets. We observe that the performance of all models is very poor. We noticed in our investigations that the majority of models predicted *active* for a large proportion of the

peptides. This general poor performance is to be expected. We attribute it to the fact that these models have been trained to recognise if a peptide exhibits antimicrobial activity against any type of species. It is therefore unreasonable to assume that they are able to discriminate activity against a specific species.

Model	Data set	S. *aureus*	P. *aeruginosa*
LA-KSVM	AMPScan	0.312	0.312
	DeepAMP	0.250	0.250
LEV-KSVM	AMPScan	0.312	0.312
	DeepAMP	0.312	0.312
GKM-SVM	AMPScan	0.312	0.312
	DeepAMP	0.375	0.375
AAC-SVM	AMPScan	0.312	0.312
	DeepAMP	0.312	0.312
Literature	AMPScan	0.250	0.250
	DeepAMP	0.438	0.438

TABLE 7.4: Predictive accuracy of the Kreĭn-SVM with local alignment score (LA-KSVM), the Kreĭn-SVM with Levenshtein distance (LEV-KSVM) and the SVM with Gapped *k*-mer kernel (GKM-SVM) on the species-specific test sets of 16 peptides. Results from the respective neural network-based classifiers(Veltri et al., 2018; Yan et al., 2020) proposed by the authors of each data set are also presented, denoted by Literature. The data set column indicates which data set a model was trained on. The heading S. *aureus* indicates the model was predicting activity against S. *aureus* ATCC 29213 and the heading P. *aeruginosa* indicates the model was predicting activity against P. *aeruginosa* ATCC 27853

Table 7.5 displays the accuracy on the external test set for models trained on the species-specific data sets. We also present the performance of the web-tools provided by Vishnepolsky et al. (2018) As mentioned in Section 7.2.1, at the time of publication, there is no web-tool to predict activity against S. *aureus* ATCC 29213. Hence, we also provide results for models trained on SA_{25923} and tasked with predicting activity against S. *aureus* ATCC 29213.

There is clearly a general improvement over the models trained on the AMP-Scan and DeepAMP data sets, indicating that the models have much greater discriminative power. On the SA_{29213} data set, the baseline SVM with amino acid composition kernel is the most predictive with respect to all the metrics. All remaining models achieve the same accuracy and MCC. Considering the SA_{25923} data set, the Levenshtein distance produces a model with the same accuracy and MCC as the web-tool but with a larger AUC. It also achieves the same accuracy and AUC as the baseline SVM with amino acid composition kernel, but with a larger MCC. It is interesting to note that the models trained on SA_{25923} can still make accurate predictions on S. *aureus* ATCC 29213. Whilst these are two different strains, the findings suggest that the antimicrobial susceptibility to the AMPs is similar for both strains, implying similar mechanisms work in the same species. Considering the PA_{27853} data set, the baseline SVM with amino acid composition kernel performs the best against all metrics. We find that the local alignment score, Levenshtein distance and baseline SVM with Gapped *k*-mer kernel produce equally accurate models, all of which are more accurate than the web-tool. However, the AUC of the local alignment score and Levenshtein distance are considerably higher than that of the baseline SVM with Gapped *k*-mer kernel. Whilst it is difficult to make any strong conclusions on such a small data set, it is still encouraging to observe that our models achieve similar accuracy to both the baseline models and web-tools.

Model	Evaluation Metric	Training Data set		
		SA_{29213}	SA_{25923}	PA_{27853}
LA-KSVM	Accuracy	0.688	0.688	0.750
	AUC	0.873	0.909	0.891
	MCC	0.522	0.405	0.595
LEV-KSVM	Accuracy	0.688	0.875	0.750
	AUC	0.927	**0.982**	0.855
	MCC	0.522	0.764	0.595
GKM-SVM	Accuracy	0.688	0.688	0.750
	AUC	0.945	0.891	0.655
	MCC	0.522	0.405	0.389
AAC-SVM	Accuracy	0.875	0.875	0.875
	AUC	**0.964**	**0.982**	**0.964**
	MCC	0.709	0.709	0.764
DBAASP Web-tool	Accuracy	-	0.875	0.688
	AUC	-	0.945	0.718
	MCC	-	0.764	0.522

TABLE 7.5: Quality of the predictions of the Kreĭn-SVM with local alignment score (LA-KSVM), the Kreĭn-SVM with Levenshtein distance (LEV-KSVM), the SVM with Gapped *k*-mer kernel (GKM-SVM) and the SVM with amino acid composition kernel (AAC-SVM) on the test sets of 16 peptides. Results from the DBAASP Web-tools are also presented(Vishnepolsky et al., 2018; Pirtskhalava et al., 2021). The headings in the third to fifth columns indicate which data set the models were trained on. The models trained on both SA_{29213} and SA_{25923} were tasked with predicting activity against S. *aureus* ATCC 29213. The models trained on PA_{27853} were tasked with predicting activity against P. *aeruginosa* ATCC 27853. Numbers highlighted in bold indicate the largest AUC achieved on the respective data set

7.4 Discussion

This chapter has proposed and evaluated a framework to classify AMPs. More specifically, we have used the Kreĭn-SVM from chapter 4, in conjunction with the local-alignment score and Levenshtein distance, to develop models capable of classifying general and species-specific antimicrobial activity. We performed two sets of experiments to evaluate the effectiveness of

our methodology to identify general antimicrobial activity. The first, a nested cross-validation experiment, was designed as a robust test of model performance. The multiple test sets of nested cross-validation allowed us to obtain a mean and standard deviation of each performance metric. It also enabled testing differences in model performance via hypothesis testing. Across two datasets, this set of experiments confirmed the validity of our approach. Indeed, our methodology achieved a greater average accuracy, AUC and MCC than the in-house baselines on both datasets. Additionally, on the AMPScan data set, we observed a significant difference between the mean AUC of the local alignment score and that of the baseline SVM with Gapped k-mer kernel. The second set of experiments allowed us to compare our approach to models from the literature, as a predefined test set was associated with each dataset. We observed similar results in this case, with our proposed approach outperforming the in-house and literature baselines. With such promising performance in predicting general antimicrobial activity, we further evaluated our models by predicting antimicrobial activity in specific species. This was accomplished by compiling a dataset of peptides with activity measured S. aureus ATCC 29213, S. aureus ATCC 25923 or P. aeruginosa ATCC 27853. The models trained on this dataset were evaluated on a small dataset of peptides with experimentally validated activities. In this case, our models fell short of one of the in-house baselines but outperformed the other in-house baseline and preexisting web-tool. However, with the testing dataset consisting of only 16 peptides, it is difficult to draw firm conclusions.

The current chapter presents significant contributions to machine learning and bioinformatics research. We have presented a methodology for classifying peptides using common sequence alignment algorithms. Whilst we have focused on the specific application of identifying antimicrobial activity in peptides, the approach we have presented is much more general. The main benefit of our approach is the implicit representation of peptides provided by the alignment measures. As this representation is agnostic of the specific task, our methodology is likely applicable to other peptide classification problems. Therefore, we have presented an approach for bioinformaticians and machine learning researchers to develop peptide classification models. Furthermore, releasing our models as web-tools allows those working in AMP research to rapidly screen a prospective set of peptides for antimicrobial activity, thus allowing practitioners to focus their efforts on only the most promising candidates. This significant contribution will help streamline laboratory experiments and aid in AMP discovery.

The success of our approach notwithstanding, we would be remiss without discussing areas of improvement. The size of the testing dataset in our species-specific experiments is a limiting factor that inhibits our ability to fully understand the effectiveness of our approach. The study could be substantially enhanced by using a larger sample size, perhaps with a wider range of species. This would allow us to draw stronger conclusions and investigate the benefits and weaknesses our approach in more detail. Another possible improvement, applicable to all our experiments, would be to equip the models with a notion of confidence. In particular, the current models predict

whether a given peptide exhibits antimicrobial properties, but they do not indicate the confidence of their prediction. This problem could be tackled in several ways, the simplest of which would be to measure the distance of an instance to the classification hyperplane. Recall from Chapter 3 that the fitting of an SVM classifier corresponds to finding a maximum-margin hyperplane that separates the two classes. That is, of all possible hyperplanes that separate the two classes, the SVM finds the one which maximises the total distance from both classes. Hence, the further away from the hyperplane an instance lies, the more confident we are in its class. Combined with a method to normalise the distances, this would produce a number in the interval $(0, 1)$ that indicates the confidence of a prediction. The procedure we have alluded to is remarkably similar to how the logistic regression algorithm operates. Indeed, it finds a hyperplane separating the two classes. Furthermore, using a sigmoid function, the distance of an instance to the hyperplane is normalised into a probabilistic prediction. Hence, another approach to embedding our models with a notion of confidence would be to use a logistic regression algorithm that operates in a Kreĭn space. Whilst potentially more beneficial, this is a more involved endeavour as it would require the derivation of a new algorithm. However, in a similar manner to how we have derived the Kreĭn-SVM by minimising the hinge loss over a RKKS, one could minimise the logistic loss over a RKKS to derive a Kreĭn logistic regression algorithm.

Chapter 8

Conclusions and Future Directions

The work done in this thesis explores the applicability of Kreĭn space methods for problems involving structured data, particularly string and graph data originating from bioinformatics and chemoinformatics. In doing so, we have made theoretical and empirical contributions to the fields of machine learning, bioinformatics and chemoinformatics. In this chapter, we summarise our main contributions and identify further research areas.

8.1 Summary of Contributions

In **Chapter 4** we introduced the major theoretical contribution of the thesis, a complete derivation and solution of the Kreĭn-SVM dual problem. The idea of the Kreĭn-SVM was first proposed by Oglic and Gärtner (2019), but only in its primal form. The primal is an optimisation problem in two variables (the coefficients and the slack variables), which makes fitting the model cumbersome. On the other hand, the dual problem is an optimisation problem in one variable that can be solved efficiently with an off-the-shelf solver. Furthermore, since the Kreĭn-SVM satisfies strong duality, one can solve the

problem in its dual without any loss of information. The derivation of the Kreĭn-SVM dual, as well as its reference implementation, represents a significant contribution to the field of machine learning, which we hope inspires the creation and adoption of other Kreĭn-based learning algorithms.

Chapter 4 provided the other theoretical contribution of the thesis. We introduced the SCK, a novel, indefinite string kernel that measures the similarity of strings by comparing their lengths when compressed. We provided an intuitive and formal construction of the SCK. The intuitive construction helped explain our reasoning for the proposal of the SCK, whilst the formal construction used the notion of Kolmogorov complexity to present a theoretical justification for its existence. We also provided a reference implementation of the SCK, making this an important contribution to the field of machine learning.

In **Chapter 5**, we introduced the first empirical contribution of the thesis. Specifically, we proposed and evaluated models to identify TIS codons from mRNA sequence data using the Kreĭn-SVM and SCK of Chapter 4. We initially validated our methodology on a synthetic dataset of strings distributed according to a k-order Markov model. This dataset was suspected to be well-suited to the PPM compression function, and the experimental evaluation supported this. We then evaluated our approach on a real-world TIS dataset, finding that our approach yielded accurate models but did not outperform the state-of-the-art baseline. Nevertheless, this chapter still contributes significantly to machine learning and bioinformatics research. The experiments on the synthetic dataset provide a deep insight into the scenarios in which

the SCK with the PPM compression function is applicable. Furthermore, the general methodology we have presented is agnostic of the specific task of TIS identification, and we suspect it would apply to other biological sequence classification tasks.

Chapter 6 presented another empirical contribution of the thesis, in which we proposed and evaluated models to predict the yield across a set of Buchwald-Hartwig C-N cross-coupling reactions. Our approach, which used the molecular graph representation of molecules, excelled when operating in known chemical space, but struggled to generalise to areas of unknown chemical space. However, similar conclusions were drawn from a positive-definite baseline using the same representation. This indicated that the molecular graph representation may not be sufficient for accurate generalisation in yield prediction. Nevertheless, the findings in this chapter represent a significant contribution to the field of chemoinformatics, as our analysis provides a starting point from which more sophisticated models can be created.

Chapter 7 presented the major empirical contribution of the thesis, which described the development and assessment of Kreĭn-SVM models to identify peptides exhibiting antimicrobial activity. We used two data sets from the literature to develop models capable of identifying general antimicrobial activity in a peptide. In a comparison against two in-house and one literature baseline, we observed that our proposed methodology performed best across all performance metrics. Using the same methodology, we extended our analysis by creating models to identify peptides active against

specific species. These were evaluated on an external data set of 16 peptides with experimentally validated activity. In this case, our approach led to accurate models but was outperformed by the baseline. A final contribution of the chapter is the release of our peptide models, both general and species-specific, as web-tools. This itself represents a significant contribution to bioinformatics research, as it will help those investigating AMPs to streamline their laboratory research. The methodology presented is also of significant value, since it is broadly applicable to all manner of peptide classification tasks.

8.2 Future Directions

Having reviewed our direct contributions, we now discuss the potential research topics which might emerge from this thesis. We distinguish between methodological and application-specific improvements.

8.2.1 Methodological Improvements

This thesis has focused on the application of Kreĭn space methods to the molecular graph and biological sequence representations. However, many other forms of structured data exist, and expanding the scope of considered data types beyond these representations would help us understand the broad applicability of Kreĭn space methods. Within the life sciences, trees are used to represent ancestral relationships of organisms and directed acyclic graphs (DAGs) can be used to represent the steps required to synthesise a molecule

(Bradshaw et al., 2020; Kapli et al., 2020). Beyond the life sciences, time-series data is often the natural description of phenomena in finance, images and videos play a central role in computer vision applications, and raw text data is the main object of study in natural language processing (Tsay, 2005; Voulodimos et al., 2018; Chowdhary and Chowdhary, 2020). Moreover, expressive indefinite kernels for these data types naturally exist. For example, edit distances and alignment measures are defined for strings, graphs, trees and time-series (Müller, 2007; Varón and Wheeler, 2012; Ganassali, 2022; Smith and Waterman, 1981; Levenshtein, 1966; Marteau, 2008; Bille, 2005; Gao et al., 2010). Additionally, the SCK of Chapter 4 applies to all digital data types, including, for instance, images and videos. As such, there is a large body of problems for which Kreĭn space methods for structured data can be applied, and doing so would provide further understanding of the benefits and weaknesses of the approach.

In Chapters 6 and 7, we alluded to the idea of developing more Kreĭn-based learning algorithms, suggesting a Kreĭn extension of beta regression to ensure the validity of yield predictions and a Kreĭn extension of logistic regression to equip our AMP classifiers with a notion of confidence. While these proposals were specific to the considered applications, they would also apply to various machine learning problems. The literature on positive-definite kernel methods is dense, with many well-known linear algorithms having an equivalent kernelised version. On the other hand, Kreĭn-based learning algorithms are a relatively new idea, and the algorithmic developments have yet to catch up with their positive-definite counterparts. We hope

the ideas presented in this thesis encourage those working in methodological development to try deriving a Kreĭn-based generalisation of a well-known kernel method. Some possible algorithms that could result from this line of work include Kreĭn-based generalised linear models (Cawley et al., 2007; Agresti, 2015), Kreĭn-based Gaussian processes or, more generally, Bayesian Kreĭn methods (Ayhan and Chu, 2012; Rasmussen and Williams, 2006) and Kreĭn-based multi-task learning (Alvarez et al., 2012; Dinuzzo et al., 2011; Ciliberto et al., 2015). We envisage an extensive library of Kreĭn-based learning algorithms, in which a practitioner could select the most appropriate kernel function and learning algorithm for their task. Moreover, a Kreĭn-based method will naturally reduce to its positive-definite counterpart when presented with a positive-definite kernel function. It may not be the case that an indefinite kernel function is the most appropriate choice, but providing the possibility of using one is certainly more beneficial than outright rejecting it.

8.2.2 Application-Specific Improvements

The models developed in this thesis have generally used small-to-medium-sized datasets, with the largest being the Buchwald-Hartwig cross-coupling dataset, containing 3955 instances. In this case, our methodology is highly applicable. Indeed, kernel methods are known to excel on small-to-medium-sized datasets. However, their application to large datasets can become computationally prohibitive due to their memory requirements. In their standard implementation, it is necessary to compute and store the whole kernel matrix as a stage in fitting a kernel method, an operation which grows as $O(N^2)$

for a data set of size N. Many modern biological and chemical datasets are orders of magnitude larger than we have considered, making our approach infeasible for large-scale analysis. However, this drawback of kernel methods has long been understood. In Sections 2.1 and 2.2.2, we mentioned the numerous approximation methods designed to alleviate this computational burden and their extensions to indefinite kernels. The most applicable example, the Nyström subsampling technique (Oglic and Gärtner, 2019), could be utilised to scale our methodology to much larger datasets of interest. For instance, our AMP classification methodology could be applied to the whole of DBAASP, which currently contains over 22,000 peptides and is continually growing (Pirtskhalava et al., 2021). Furthermore, we could apply our procedure to produce yield prediction models to the USPTO reaction dataset, which contains over 1.8 million organic chemical reactions extracted from US patents and patent applications (Lowe, 2012). Finally, our approach to perform TIS identification could be applied to the Gao15 dataset (Zhang et al., 2017), which contains over 100,000 mRNA sequences. The application of our methodology to much larger-scale datasets would provide further insight into its effectiveness, allowing us to draw much stronger conclusions than what is possible on the datasets we have considered.

In Chapter 5, we suggested that combining the representation of multiple compression functions may prove to be beneficial. We believe this is true not only for our TIS identification models but, more generally, that all of our applications could benefit from multiple representations. Peptides and proteins can be considered from three different levels of abstraction (Buxbaum

et al., 2007). The primary structure, which is the most abstract, considers only the sequence of amino acids and is the structure we have utilised. The secondary structure is the second abstraction layer and describes a protein via its local spatial conformation. Due to interactions between the atoms in amino acids, proteins tend to arrange themselves into common shapes, the two most common being α-helices and β-sheets. The least abstract description is the tertiary structure, which describes the overall three-dimensional structure of a protein. There is also the quaternary structure, which only applies to proteins consisting of multiple peptide chains. DNA and mRNA molecules can also be described by primary, secondary, tertiary or quaternary structure (Lodish, 2008). Furthermore, the notion of layers of abstraction also exists when describing molecules. Indeed, whilst we have worked with the molecular graph representation, we could have also considered the three-dimensional conformation of a molecule. In the applications considered, we have access to a whole hierarchy of levels to describe the instances but have only considered the most abstract. The reasons for doing so are mainly that of efficiency and practicality. Moving down the layers of abstraction requires more complex methods to process data and is also more computationally demanding. Indeed, one would require a kernel function operating on every considered representation. Nevertheless, it would be interesting to investigate how using less abstract descriptions, independently or in combination with more abstract descriptions, would influence our models.

Bibliography

A. Agresti (2015). *Foundations of linear and generalized linear models*. John Wiley & Sons.

D. Ahneman, J. Estrada, S. Lin, S. Dreher, and A. Doyle (2018). "Predicting reaction performance in CN cross-coupling using machine learning". *Science* 360, pp. 186–190.

T. Akutsu and H. Nagamochi (2011). "Kernel methods for chemical compounds: From classification to design". *IEICE TRANSACTIONS on Information and Systems* 94, pp. 1846–1853.

W. Ali, A. Elsahn, D. S. Ting, H. S. Dua, and I. Mohammed (2022). "Host Defence Peptides: A Potent Alternative to Combat Antimicrobial Resistance in the Era of the COVID-19 Pandemic". *Antibiotics* 11, p. 475.

D. Alpay (1991). "Some remarks on reproducing kernel Kreıĭn spaces". *Rocky Mt. J. Math.*, pp. 1189–1205.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman (1990). "Basic local alignment search tool". *J. Mol. Biol.* 215, pp. 403–410.

S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs". *Nucleic acids research* 25, pp. 3389–3402.

M. A. Alvarez, L. Rosasco, N. D. Lawrence, et al. (2012). "Kernels for vector-valued functions: A review". *Foundations and Trends® in Machine Learning* 4, pp. 195–266.

P. G. Aronica, L. M. Reid, N. Desai, J. Li, S. J. Fox, S. Yadahalli, J. W. Essex, and C. S. Verma (2021). "Computational methods and tools in antimicrobial peptide research". *J. Chem. Inf. Model.* 61, pp. 3172–3196.

M. S. Ayhan and C.-H. H. Chu (2012). *Towards indefinite Gaussian processes*. Tech. rep. Technical report, University of Louisiana at Lafayette.

F. Bach (2013). "Sharp analysis of low-rank kernel matrix approximations". In: *Conference on learning theory*. PMLR, pp. 185–209.

M. Bhagwat and L. Aravind (2008). "Psi-blast tutorial". *Comparative genomics*, pp. 177–186.

P. Bille (2005). "A survey on tree edit distance and related problems". *Theor. Comput. Sci.* 337.

E. J. Bjerrum (2017). "Smiles enumeration as data augmentation for neural network modeling of molecules". *arXiv preprint arXiv:1703.07076*.

D. Blakely, E. Collins, R. Singh, A. Norton, J. Lanchantin, and Y. Qi (2020). "FastSK: fast sequence analysis with gapped string kernels". *Bioinformatics* 36, pp. i857–i865.

J. Bognár (1974). *Indefinite inner product spaces*. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer.

K. M. Borgwardt (2011). "Kernel methods in bioinformatics". In: *Handbook of statistical bioinformatics*. Springer, pp. 317–334.

S. Boyd and L. Vandenberghe (2004). *Convex optimization*. Cambridge university press.

J. Bradshaw, B. Paige, M. J. Kusner, M. Segler, and J. M. Hernández-Lobato (2020). "Barking up the right tree: an approach to search over molecule synthesis dags". *Advances in neural information processing systems* 33, pp. 6852–6866.

E. Buxbaum et al. (2007). *Fundamentals of protein structure and function*. Vol. 31. Springer.

D.-S. Cao, J.-C. Zhao, Y.-N. Yang, C.-X. Zhao, J. Yan, S. Liu, Q.-N. Hu, Q.-S. Xu, and Y.-Z. Liang (2012). "In silico toxicity prediction by support vector machine and SMILES representation-based string kernel". *SAR and QSAR in Environmental Research* 23, pp. 141–153.

A. Capecchi, X. Cai, H. Personne, T. Köhler, C. van Delden, and J.-L. Reymond (2021). "Machine learning designs non-hemolytic antimicrobial peptides". *Chem. Sci.* 12, pp. 9221–9232.

G. C. Cawley, G. J. Janacek, and N. L. Talbot (2007). "Generalised kernel machines". In: *2007 International Joint Conference on Neural Networks*. IEEE, pp. 1720–1725.

P. Cayley (1874). "LVII. On the mathematical theory of isomers". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 47, pp. 444–447.

A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé, and G. Pujadas (2015). "Molecular fingerprint similarity search in virtual screening". *Methods* 71, pp. 58–63.

A. Ceroni, F. Costa, and P. Frasconi (2007). "Classification of small molecules by two-and three-dimensional decomposition kernels". *Bioinformatics* 23, pp. 2038–2045.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee (2002). "Choosing multiple parameters for support vector machines". *Mach. Learn.* 46, pp. 131–159.

Y. Chen, M. R. Gupta, and B. Recht (2009). "Learning kernels from indefinite similarities". In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 145–152.

K.-C. Chou (2001). "Prediction of protein cellular attributes using pseudo-amino acid composition". *Proteins: Struct. Funct. Genet.* 43, pp. 246–255.

K. Chowdhary and K. Chowdhary (2020). "Natural language processing". *Fundamentals of artificial intelligence*, pp. 603–649.

C. Ciliberto, Y. Mroueh, T. Poggio, and L. Rosasco (2015). "Convex learning of multiple tasks and their structure". In: *International Conference on Machine Learning*. PMLR, pp. 1548–1557.

J. Cleary and I. Witten (1984). "Data compression using adaptive coding and partial string matching". *IEEE Trans. Commun.* 32, pp. 396–402.

T. M. Cover (1999). *Elements of information theory*. John Wiley & Sons.

T. M. Cover (1965). "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition". *IEEE transactions on electronic computers*, pp. 326–334.

N. Cristianini, J. Shawe-Taylor, et al. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

J. Daily (2016). "Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments". *BMC Bioinform.* 17, pp. 1–11.

P. Das, T. Sercu, K. Wadhawan, I. Padhi, S. Gehrmann, F. Cipcigan, V. Chenthamarakshan, H. Strobelt, C. Dos Santos, P.-Y. Chen, et al. (2021). "Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations". *Nat. Biomed. Eng.* 5, pp. 613–623.

S. Deorowicz (2020). "FQSqueezer: k-mer-based compression of sequencing data". *Scientific reports* 10, p. 578.

T. E. Dever and R. Green (2012). "The elongation, termination, and recycling phases of translation in eukaryotes". *Cold Spring Harbor perspectives in biology* 4, a013706.

I. S. Dhillon, Y. Guan, and B. Kulis (2004). "Kernel k-means: spectral clustering and normalized cuts". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 551–556.

F. Dinuzzo, C. S. Ong, G. Pillonetto, and P. V. Gehler (2011). "Learning output kernels with block coordinate descent". In: *Proceedings of the 28th international conference on machine learning (icml-11)*, pp. 49–56.

R. Dorel, C. P. Grugel, and A. M. Haydl (2019). "The Buchwald–Hartwig amination after 25 years". *Angewandte Chemie International Edition* 58, pp. 17118–17129.

R. P. Duin and E. Pekalska (2005). *Dissimilarity representation for pattern recognition, the: foundations and applications*. Vol. 64. World scientific.

J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse (2002). "Reoptimization of MDL Keys for Use in Drug Discovery". *J. Chem. Inf. Comput. Sci.* 42, pp. 1273–1280.

T. Fawcett (2006). "An introduction to ROC analysis". *Pattern Recognit. Lett.* 27, pp. 861–874.

P.-M. Feng, W. Chen, H. Lin, and K.-C. Chou (2013). "iHSP-PseRAAAC: Identifying the heat shock protein families using pseudo reduced amino acid alphabet composition". *Anal. Biochem.* 442, pp. 118–125.

S. Ferrari and F. Cribari-Neto (2004). "Beta regression for modelling rates and proportions". *Journal of applied statistics* 31, pp. 799–815.

C. D. Fjell, J. A. Hiss, R. E. Hancock, and G. Schneider (2012). "Designing antimicrobial peptides: form follows function". *Nat. Rev. Drug Discov.* 11, pp. 37–51.

C. A. Fleckenstein and H. Plenio (2007). "9-Fluorenylphosphines for the Pd-Catalyzed Sonogashira, Suzuki, and Buchwald–Hartwig Coupling Reactions in Organic Solvents and Water". *Chemistry–A European Journal* 13, pp. 2701–2716.

P. A. Forero-Cortés and A. M. Haydl (2019). "The 25th anniversary of the Buchwald–Hartwig amination: development, applications, and outlook". *Organic Process Research & Development* 23, pp. 1478–1483.

H. Fröhlich, J. K. Wegner, F. Sieker, and A. Zell (2005). "Optimal assignment kernels for attributed molecular graphs". In: *Proceedings of the Twenty-Second International Conference on Machine learning*, pp. 225–232.

J.-l. Gailly and M. Adler (2020). *zlib compression algorithm*. URL: https://www.zlib.net.

L. Ganassali (2022). "The graph alignment problem: fundamental limits and efficient algorithms". PhD thesis. PSL Research University; Ecole normale supérieure.

X. Gao, B. Xiao, D. Tao, and X. Li (2010). "A survey of graph edit distance". *Pattern Anal. Appl.* 13, pp. 113–129.

T. Gärtner (2003). "A survey of kernels for structured data". *ACM SIGKDD explorations newsletter* 5, pp. 49–58.

M. Ghandi, D. Lee, M. Mohammad-Noori, and M. A. Beer (2014). "Enhanced regulatory sequence prediction using gapped k-mer features". *PLoS Comput. Biol.* 10, e1003711.

B. Ghojogh, F. Karray, and M. Crowley (2019). "Fisher and kernel Fisher discriminant analysis: Tutorial". *arXiv preprint arXiv:1906.09436*.

S. Giguere, F. Laviolette, M. Marchand, D. Tremblay, S. Moineau, X. Liang, É. Biron, and J. Corbeil (2015). "Machine learning assisted design of highly active peptides for drug discovery". *PLoS computational biology* 11, e1004074.

S. Giguere, M. Marchand, F. Laviolette, A. Drouin, and J. Corbeil (2013). "Learning a peptide-protein binding affinity predictor with kernel ridge regression". *BMC bioinformatics* 14, pp. 1–16.

A. C. Gleason, G. Ghadge, J. Chen, Y. Sonobe, and R. P. Roos (2022). "Machine learning predicts translation initiation sites in neurologic diseases with nucleotide repeat expansions". *PLoS One* 17, e0256411.

M. Gönen and E. Alpaydın (2011). "Multiple kernel learning algorithms". *The Journal of Machine Learning Research* 12, pp. 2211–2268.

Google (2020). *Snappy compression algorithm*. URL: http://google.github.io/snappy/.

A. Gupta, A. Bansal, and V. Khanduja (2017). "Modern lossless compression techniques: Review, comparison and analysis". In: *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, pp. 1–8.

A. S. Guram and S. L. Buchwald (1994). "Palladium-catalyzed aromatic aminations with in situ generated aminostannanes". *Journal of the American Chemical Society* 116, pp. 7901–7902.

B. Haasdonk (2005). "Feature space interpretation of SVMs with indefinite kernels". *IEEE Transactions on pattern analysis and machine intelligence* 27, pp. 482–492.

B. Haasdonk and E. Pekalska (2008). "Indefinite kernel fisher discriminant". In: *2008 19th International Conference on Pattern Recognition*. IEEE, pp. 1–4.

R. E. Hancock, E. F. Haney, and E. E. Gill (2016). "The immunology of host defence peptides: beyond antimicrobial activity". *Nat. Rev. Immunol.* 16, pp. 321–334.

A. L. Haywood, J. Redshaw, M. W. Hanson-Heine, A. Taylor, A. Brown, A. M. Mason, T. Gärtner, and J. D. Hirst (2021). "Kernel methods for predicting yields of chemical reactions". *Journal of Chemical Information and Modeling* 62, pp. 2077–2092.

X. He, K. Zhao, and X. Chu (2021). "AutoML: A survey of the state-of-the-art". *Knowledge-based systems* 212, p. 106622.

S. Henikoff and J. G. Henikoff (1992). "Amino acid substitution matrices from protein blocks". *Proc. Natl. Acad. Sci. U.S.A.* 89, pp. 10915–10919.

J. W. Hershey, N. Sonenberg, and M. B. Mathews (2012). "Principles of translational control: an overview". *Cold Spring Harbor perspectives in biology* 4, a011528.

N. J. Higham (1988). "Computing a nearest symmetric positive semidefinite matrix". *Linear Algebra Appl.* 103, pp. 103–118.

J. D. Hirst, S. Boobier, J. Coughlan, J. Streets, P. L. Jacob, O. Pugh, E. Özcan, and S. Woodward (2023). "ML meets MLn: machine learning in ligand promoted homogeneous catalysis". *Artificial Intelligence Chemistry* 1, p. 100006.

J. Hodges Jr (1958). "The significance probability of the Smirnov two-sample test". *Arkiv för matematik* 3, pp. 469–486.

B. Hoffmann, M. Zaslavskiy, J.-P. Vert, and V. Stoven (2010). "A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3D: application to ligand prediction". *BMC bioinformatics* 11, pp. 1–16.

J. P. Hughes, S. Rees, S. B. Kalindjian, and K. L. Philpott (2011). "Principles of early drug discovery". *British journal of pharmacology* 162, pp. 1239–1249.

D. Hyatt, G.-L. Chen, P. F. LoCascio, M. L. Land, F. W. Larimer, and L. J. Hauser (2010). "Prodigal: prokaryotic gene recognition and translation initiation site identification". *BMC bioinformatics* 11, pp. 1–11.

D. Hyatt, P. F. LoCascio, L. J. Hauser, and E. C. Uberbacher (2012). "Gene and translation initiation site prediction in metagenomic sequences". *Bioinformatics* 28, pp. 2223–2230.

R. J. Jackson, C. U. Hellen, and T. V. Pestova (2010). "The mechanism of eukaryotic translation initiation and principles of its regulation". *Nature reviews Molecular cell biology* 11, pp. 113–127.

M. R. Kantorovitz, G. E. Robinson, and S. Sinha (2007). "A statistical method for alignment-free comparison of regulatory sequences". *Bioinformatics* 23, pp. i249–i255.

P. Kapli, Z. Yang, and M. J. Telford (2020). "Phylogenetic tree building in the genomic age". *Nature Reviews Genetics* 21, pp. 428–444.

M. Kapur and S. L. Ackerman (2018). "mRNA translation gone awry: translation fidelity and neurological disease". *Trends in Genetics* 34, pp. 218–231.

S. K. Karmaker, M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni (2021). "Automl to date and beyond: Challenges and opportunities". *ACM Computing Surveys (CSUR)* 54, pp. 1–36.

H. Kim, J. H. Jang, S. C. Kim, and J. H. Cho (2014). "De novo generation of short antimicrobial peptides with enhanced stability and cell specificity". *J. Antimicrob. Chemother.* 69, pp. 121–132.

A. Kocsor, A. Kertész-Farkas, L. Kaján, and S. Pongor (2006). "Application of compression-based distance measures to protein sequence classification: a methodological study". *Bioinformatics* 22, pp. 407–412.

A. N. Kolmogorov (1965). "Three approaches to the quantitative definition ofinformation'". *Problems of information transmission* 1, pp. 1–7.

M. Kozak (2002). "Emerging links between initiation of translation and human diseases". *Mammalian Genome* 13, p. 401.

E. Kreyszig (1991). *Introductory functional analysis with applications*. Vol. 17. John Wiley & Sons.

P. Kuksa and V. Pavlovic (2009). "Efficient alignment-free DNA barcode analytics". *BMC Bioinform.* 10, pp. 1–18.

P. S. Kutchukian, J. F. Dropinski, K. D. Dykstra, B. Li, D. A. DiRocco, E. C. Streckfuss, L.-C. Campeau, T. Cernak, P. Vachal, I. W. Davies, et al. (2016). "Chemistry informer libraries: a chemoinformatics enabled approach to evaluate and advance synthetic methods". *Chem. Sci.* 7, pp. 2604–2613.

Y. Kwon, D. Lee, Y.-S. Choi, and S. Kang (2022). "Uncertainty-aware prediction of chemical reaction yields with graph neural networks". *Journal of Cheminformatics* 14, pp. 1–10.

S. Lata, N. K. Mishra, and G. P. Raghava (2010). "AntiBP2: improved version of antibacterial peptide prediction". *BMC Bioinform.* 11, pp. 1–7.

E. Y. Lee, B. M. Fulan, G. C. Wong, and A. L. Ferguson (2016). "Mapping membrane activity in undiscovered peptide sequence space using machine learning". *Proc. Natl. Acad. Sci. U.S.A.* 113, pp. 13588–13593.

A. Leman and B. Weisfeiler (1968). "A reduction of a graph to a canonical form and an algebra arising during this reduction". *Nauchno-Technicheskaya Informatsiya* 2, pp. 12–16.

A. Lempel and J. Ziv (1976). "On the complexity of finite sequences". *IEEE Transactions on information theory* 22, pp. 75–81.

C. Leslie, E. Eskin, and W. S. Noble (2001). "The spectrum kernel: A string kernel for SVM protein classification". In: *Biocomputing 2002*. World Scientific, pp. 564–575.

C. Leslie, E. Eskin, J. Weston, and W. S. Noble (2003). "Mismatch string kernels for SVM protein classification". *Advances in neural information processing systems*, pp. 1441–1448.

C. Leslie, R. Kuang, and K. Bennett (2004). "Fast string kernels using inexact matching for protein sequences." *Journal of Machine Learning Research* 5.

V. I. Levenshtein (1966). "Binary codes capable of correcting deletions, insertions, and reversals". In: *Sov. Phys. Dok.* Vol. 10. Soviet Union, pp. 707–710.

C. Li, D. Sutherland, S. A. Hammond, C. Yang, F. Taho, L. Bergman, S. Houston, R. L. Warren, T. Wong, L. Hoang, et al. (2022). "AMPlify: attentive deep learning model for discovery of novel antimicrobial peptides effective against WHO priority pathogens". *BMC Genom.* 23, pp. 1–15.

M. Li, X. Chen, X. Li, B. Ma, and P. M. Vitányi (2004). "The similarity metric". *IEEE transactions on Information Theory* 50, pp. 3250–3264.

M. Li, P. Vitányi, et al. (2008). *An introduction to Kolmogorov complexity and its applications*. Vol. 3. Springer.

L. Liao and W. S. Noble (2003). "Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships". *Journal of computational biology* 10, pp. 857–868.

F. Liu, X. Huang, Y. Chen, and J. Suykens (2021). "Fast learning in reproducing kernel Kreĭn spaces via signed measures". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 388–396.

H. F. Lodish (2008). *Molecular cell biology*. Macmillan.

G. Loosli, S. Canu, and C. S. Ong (2015). "Learning SVM in Kreıĭn spaces". *IEEE Trans. Pattern Anal. Mach. Intell.* 38, pp. 1204–1216.

J. Louie and J. F. Hartwig (1995). "Palladium-catalyzed synthesis of arylamines from aryl halides. Mechanistic studies lead to coupling in the absence of tin reagents". *Tetrahedron Letters* 36, pp. 3609–3612.

D. M. Lowe (2012). "Extraction of chemical structures and reactions from the literature". PhD thesis.

R. Luss and A. d'Aspremont (2007). "Support vector machine classification with indefinite kernels". *Advances in neural information processing systems* 20.

F. Madeira, M. Pearce, A. R. Tivey, P. Basutkar, J. Lee, O. Edbali, N. Madhusoodanan, A. Kolesnikov, and R. Lopez (2022). "Search and sequence analysis tools services from EMBL-EBI in 2022". *Nucleic acids research* 50, W276–W279.

J. Magano and J. R. Dunetz (2011). "Large-scale applications of transition metal-catalyzed couplings for the synthesis of pharmaceuticals". *Chemical Reviews* 111, pp. 2177–2250.

P.-F. Marteau (2008). "Time warp edit distance with stiffness adjustment for time series matching". *IEEE transactions on pattern analysis and machine intelligence* 31, pp. 306–318.

V. Mayandi, Q. Xi, E. T. Leng Goh, S. K. Koh, T. Y. Jie Toh, V. A. Barathi, M. H. Urf Turabe Fazil, M. L. Somaraju Chalasani, J. Varadarajan, D. S. J. Ting, et al. (2020). "Rational substitution of ε-lysine for α-Lysine enhances the cell and membrane selectivity of pore-forming melittin". *J. Med. Chem.* 63, pp. 3522–3537.

P. K. Meher, T. K. Sahu, V. Saini, and A. R. Rao (2017). "Predicting antimicrobial peptides with improved accuracy by incorporating the compositional, physico-chemical and structural features into Chou's general PseAAC". *Sci. Rep.* 7, pp. 1–12.

J. L. Melville, J. F. Riley, and J. D. Hirst (2007). "Similarity by compression". *J. Chem. Inf. Model.* 47, pp. 25–33.

H. Q. Minh, P. Niyogi, and Y. Yao (2006). "Mercer's theorem, feature maps, and smoothing". In: *Learning Theory: 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006. Proceedings 19*. Springer, pp. 154–168.

N. Mookherjee, M. A. Anderson, H. P. Haagsman, and D. J. Davidson (2020). "Antimicrobial host defence peptides: functions and clinical potential". *Nat. Rev. Drug Discov.* 19, pp. 311–332.

H. L. Morgan (1965). "The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service." *Journal of Chemical Documentation* 5, pp. 107–113.

M. Müller (2007). "Dynamic time warping". *Information Retrieval for Music and Motion*, pp. 69–84.

N. V. T. S. Munagala, P. K. Amanchi, K. Balasubramanian, A. Panicker, and N. Nagaraj (2022). "Compression-Complexity Measures for Analysis and Classification of Coronaviruses". *Entropy* 25, p. 81.

C. J. Murray, K. S. Ikuta, F. Sharara, L. Swetschinski, G. R. Aguilar, A. Gray, C. Han, C. Bisignano, P. Rao, E. Wool, et al. (2022). "Global burden of bacterial antimicrobial resistance in 2019: a systematic analysis". *Lancet*.

N. Nagaraj, K. Balasubramanian, and S. Dey (2013). "A new complexity measure for time series analysis and classification". *The European Physical Journal Special Topics* 222, pp. 847–860.

S. B. Needleman and C. D. Wunsch (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". *J. Mol. Biol.* 48, pp. 443–453.

X. Y. Ng, B. A. Rosdi, and S. Shahrudin (2015). "Prediction of antimicrobial peptides based on sequence alignment and support vector machine-pairwise algorithm utilizing LZ-complexity". *BioMed Res. Int.*, p. 212715.

D. Oglic and T. Gärtner (2018). "Learning in Reproducing Kernel Kreıĭn Spaces". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80, pp. 3859–3867.

— (2017). "Nyström method with kernel k-means++ samples as landmarks". In: *International Conference on Machine Learning*. PMLR, pp. 2652–2660.

— (2019). "Scalable Learning in Reproducing Kernel Kreıĭn Spaces". In: *International Conference on Machine Learning*, pp. 4912–4921.

D. Oglic, S. A. Oatley, S. J. Macdonald, T. Mcinally, R. Garnett, J. D. Hirst, and T. Gärtner (2018). "Active Search for Computer-aided Drug Design". *Mol. Inform.* 37, p. 1700130.

C. S. Ong, X. Mary, S. Canu, and A. J. Smola (2004). "Learning with non-positive kernels". In: *Proceedings of the Twenty-First International Conference on Machine learning*, p. 81.

A. Passerini (2013). "Kernel methods for structured data". In: *Handbook on Neural Information Processing*. Springer, pp. 283–333.

A. G. Pedersen and H. Nielsen (1997). "Neural network prediction of translation initiation sites in eukaryotes: perspectives for EST and genome analysis." In: *ISMB*. Vol. 5, pp. 226–233.

E. Pekalska and B. Haasdonk (2008). "Kernel discriminant analysis for positive definite and indefinite kernels". *IEEE transactions on pattern analysis and machine intelligence* 31, pp. 1017–1032.

E. Pękalska, R. P. Duin, S. Günter, and H. Bunke (2004). "On not making dissimilarities Euclidean". In: *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18-20, 2004. Proceedings*. Springer, pp. 1145–1154.

S. Pethel and D. Hahs (2014). "Exact significance test for Markov order". *Physica D: Nonlinear Phenomena* 269, pp. 42–47.

S. A. Pinacho-Castellanos, C. R. García-Jacas, M. K. Gilson, and C. A. Brizuela (2021). "Alignment-Free Antimicrobial Peptide Predictors: Improving Performance by a Thorough Analysis of the Largest Available Data Set". *J. Chem. Inf. Model.* 61, pp. 3141–3157.

M. Pirtskhalava, A. A. Amstrong, M. Grigolava, M. Chubinidze, E. Alimbarashvili, B. Vishnepolsky, A. Gabrielian, A. Rosenthal, D. E. Hurt, and M. Tartakovsky (2021). "DBAASP v3: Database of antimicrobial/cytotoxic activity and structure of peptides as a resource for development of new therapeutics". *Nucleic Acids Res.* 49, pp. D288–D297.

J. Platt (1998). "Sequential minimal optimization: A fast algorithm for training support vector machines".

D. Pratas, M. Hosseini, and A. J. Pinho (2020). "GeCo2: An optimized tool for lossless compression and analysis of DNA sequences". In: *Practical Applications of Computational Biology and Bioinformatics, 13th International Conference*. Springer, pp. 137–145.

J. Qiu, M. Hue, A. Ben-Hur, J.-P. Vert, and W. S. Noble (2007). "A structural alignment kernel for protein structures". *Bioinformatics* 23, pp. 1090–1098.

A. Rahimi and B. Recht (2007). "Random features for large-scale kernel machines". *Advances in neural information processing systems* 20.

P. Rajarajeswari and A. Apparao (2011). "DNABIT compress–genome compression algorithm". *Bioinformation* 5, p. 350.

L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi (2005). "Graph kernels for chemical informatics". *Neural networks* 18, pp. 1093–1110.

S. Ramesh, T. Govender, H. G. Kruger, B. G. de la Torre, and F. Albericio (2016). "Short AntiMicrobial Peptides (SAMPs) as a class of extraordinary promising therapeutic agents". *J. Pept. Sci.* 22, pp. 438–451.

C. E. Rasmussen and C. K. I. Williams (2006). *Gaussian processes for machine learning*. *Adaptive Computation and Machine Learning*. MIT Press.

G. Rätsch and S. Sonnenburg (2004). "Accurate Splice Site Detection for Caenorhabditis elegans". *Kernel methods in computational biology* 277.

D. Rogers and M. Hahn (2010). "Extended-connectivity fingerprints". *Journal of chemical information and modeling* 50, pp. 742–754.

G. D. Rose, A. R. Geselowitz, G. J. Lesser, R. H. Lee, and M. H. Zehfus (1985). "Hydrophobicity of amino acid residues in globular proteins". *Science* 229, pp. 834–838.

A. Rudi, R. Camoriano, and L. Rosasco (2015). "Less is more: Nyström computational regularization". *Advances in Neural Information Processing Systems* 28.

A. Rudi, L. Carratino, and L. Rosasco (2017). "Falkon: An optimal large scale kernel method". *Advances in neural information processing systems* 30.

A. Rudi and L. Rosasco (2017). "Generalization properties of learning with random features". *Advances in neural information processing systems* 30.

P. Ruiz-Castillo and S. L. Buchwald (2016). "Applications of palladium-catalyzed C-N cross-coupling reactions". *Chem. Rev.* 116, pp. 12564–12649.

D. Salomon (2007). *Data compression: the complete reference*. 4th ed. London: Springer.

C. Saunders and A. Demco (2007). "Kernels for strings and graphs". In: *Perspectives of Neural-Symbolic Integration*. Springer, pp. 7–22.

D. J. Schaid (2010). "Genomic similarity and kernel methods II: methods for genomic information". *Human heredity* 70, pp. 132–140.

B. Schölkopf, R. Herbrich, and A. J. Smola (2001). "A generalized representer theorem". In: *International Conference on Computational Learning Theory*. Springer, pp. 416–426.

B. Schölkopf, A. Smola, and K.-R. Müller (1997). "Kernel principal component analysis". In: *International conference on artificial neural networks*. Springer, pp. 583–588.

B. Schölkopf and A. J. Smola (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press.

B. Schölkopf, K. Tsuda, and J.-P. Vert (2004). *Kernel Methods in Computational Biology*. MIT press.

R. Schwartz (1978). "Matrices for detecting distant relationships". *Atlas of protein sequence and structure*, pp. 353–359.

D. Sculley and C. E. Brodley (2006). "Compression and machine learning: A new perspective on feature space vectors". In: *Data Compression Conference*. IEEE, pp. 332–341.

J. C. Setubal and J. Meidanis (1997). *Introduction to Computational Molecular Biology*. PWS Pub. Boston.

A. R. Shah, C. S. Oehmen, and B.-J. Webb-Robertson (2008). "SVM-HUSTLE—an iterative semi-supervised machine learning approach for pairwise protein remote homology detection". *Bioinformatics* 24, pp. 783–790.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas (2015). "Taking the human out of the loop: A review of Bayesian optimization". *Proceedings of the IEEE* 104, pp. 148–175.

S. Shalev-Shwartz and S. Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt (2011). "Weisfeiler-Lehman graph kernels". *J. Mach. Learn. Res.* 12, pp. 2539–2561.

A. Shrikumar, E. Prakash, and A. Kundaje (2019). "GkmExplain: fast and accurate interpretation of nonlinear gapped k-mer SVMs". *Bioinformatics* 35, pp. i173–i182.

G. Sliwoski, S. Kothiwale, J. Meiler, and E. W. Lowe (2014). "Computational methods in drug discovery". *Pharmacological reviews* 66, pp. 334–395.

T. F. Smith and M. S. Waterman (1981). "Identification of common molecular subsequences". *J. Mol. Biol.* 147, pp. 195–197.

N. Sonenberg and A. G. Hinnebusch (2009). "Regulation of translation initiation in eukaryotes: mechanisms and biological targets". *Cell* 136, pp. 731–745.

I. Steinwart and A. Christmann (2008). *Support vector machines*. Springer Science & Business Media.

D. Stumpfe, H. Hu, and J. Bajorath (2019). "Evolving concept of activity cliffs". *Acs Omega* 4, pp. 14360–14368.

S. J. Swamidass, J. Chen, J. Bruand, P. Phung, L. Ralaivola, and P. Baldi (2005). "Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity". *Bioinformatics* 21, pp. i359–i368.

N. Takahashi, J. Guo, and T. Nishi (2008). "Global convergence of SMO algorithm for support vector regression". *IEEE Transactions on Neural Networks* 19, pp. 971–982.

T. T. Tanimoto (1958). "Elementary mathematical theory of classification and prediction".

S. Thomas, S. Karnik, R. S. Barai, V. K. Jayaraman, and S. Idicula-Thomas (2010). "CAMP: a useful resource for research on antimicrobial peptides". *Nucleic Acids Res.* 38, pp. D774–D780.

D. S. J. Ting, R. W. Beuerman, H. S. Dua, R. Lakshminarayanan, and I. Mohammed (2020). "Strategies in translating the therapeutic potentials of host defense peptides". *Front. Immunol.* 11, p. 983.

D. S. J. Ting, E. T. L. Goh, V. Mayandi, J. M. Busoy, T. T. Aung, M. H. Periayah, M. Nubile, L. Mastropasqua, D. G. Said, H. M. Htoon, et al. (2021a). "Hybrid derivative of cathelicidin and human beta defensin-2 against Gram-positive bacteria: A novel approach for the treatment of bacterial keratitis". *Sci. Rep.* 11, pp. 1–14.

D. S. J. Ting, J. Li, C. S. Verma, E. T. Goh, M. Nubile, L. Mastropasqua, D. G. Said, R. W. Beuerman, R. Lakshminarayanan, I. Mohammed, et al. (2021b). "Evaluation of host defense peptide (CaD23)-Antibiotic interaction and mechanism of action: Insights from experimental and molecular dynamics simulations studies". *Front. Pharmacol.*, p. 2793.

D. S. J. Ting, I. Mohammed, L. Rajamani, R. W. Beuerman, and H. S. Dua (2022). "Host Defense Peptides at the Ocular Surface: Roles in Health and Major Diseases, and Therapeutic Potentials". *Front. Med.*, p. 1667.

M. A. Topchiy, A. F. Asachenko, and M. S. Nechaev (2014). "Solvent-free Buchwald–Hartwig reaction of aryl and heteroaryl halides with secondary amines". *European Journal of Organic Chemistry* 2014, pp. 3319–3322.

C. Torborg and M. Beller (2009). "Recent applications of palladium-catalyzed coupling reactions in the pharmaceutical, agrochemical, and fine chemical industries". *Advanced Synthesis & Catalysis* 351, pp. 3027–3043.

M. Torrent, D. Andreu, V. M. Nogués, and E. Boix (2011). "Connecting peptide physicochemical and antimicrobial properties by a rational prediction model". *PloS One* 6, e16968.

R. S. Tsay (2005). *Analysis of financial time series*. John wiley & sons.

K. Tsuda (1999). "Support vector classification with asymmetric kernel function". In: *European Symposium on Artificial Neural Networks*, pp. 183–188.

K. Ullrich, J. Mack, and P. Welke (2016). "Ligand affinity prediction with multi-pattern kernels". In: *Discovery Science: 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings 19*. Springer, pp. 474–489.

L. Van Der Maaten, E. O. Postma, H. J. van den Herik, et al. (2009). "Dimensionality reduction: A comparative review". *Journal of Machine Learning Research* 10, p. 13.

V. N. Vapnik (1999). "An overview of statistical learning theory". *IEEE transactions on neural networks* 10, pp. 988–999.

A. Varón and W. C. Wheeler (2012). "The tree alignment problem". *BMC bioinformatics* 13, pp. 1–14.

D. Veltri, U. Kamath, and A. Shehu (2018). "Deep learning improves antimicrobial peptide recognition". *Bioinformatics* 34, pp. 2740–2747.

J.-P. Vert (2008). "The optimal assignment kernel is not positive definite". *None*.

J.-P. Vert, H. Saigo, and T. Akutsu (2004). "Local alignment kernels for biological sequences". *None*, pp. 131–154.

B. Vishnepolsky, A. Gabrielian, A. Rosenthal, D. E. Hurt, M. Tartakovsky, G. Managadze, M. Grigolava, G. I. Makhatadze, and M. Pirtskhalava (2018). "Predictive model of linear antimicrobial peptides active against gram-negative bacteria". *J. Chem. Inf. Model.* 58, pp. 1141–1151.

E. Vitaku, D. T. Smith, and J. T. Njardarson (2014). "Analysis of the structural diversity, substitution patterns, and frequency of nitrogen heterocycles among US FDA approved pharmaceuticals: miniperspective". *Journal of medicinal chemistry* 57, pp. 10257–10274.

D. V. Volgin (2014). "Chapter 17 - Gene Expression: Analysis and Quantitation". In: *Animal Biotechnology*. Ed. by A. S. Verma and A. Singh. San Diego: Academic Press, pp. 307–325.

A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis (2018). "Deep learning for computer vision: A brief review". *Computational intelligence and neuroscience* 2018, p. 7068349.

P. Wagner, M. Bollenbach, C. Doebelin, F. Bihel, J.-J. Bourguignon, C. Salomé, and M. Schmitt (2014). "t-BuXPhos: a highly efficient ligand for Buchwald–Hartwig coupling in water". *Green chemistry* 16, pp. 4170–4178.

P. Wang, L. Hu, G. Liu, N. Jiang, X. Chen, J. Xu, W. Zheng, L. Li, M. Tan, Z. Chen, et al. (2011). "Prediction of antimicrobial peptides based on sequence alignment and feature selection methods". *PloS One* 6, e18476.

R. Wang, Y. Xu, and B. Liu (2016). "Recombination spot identification Based on gapped k-mers". *Sci. Rep.* 6, pp. 1–10.

D. Weininger (1988). "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". *Journal of chemical information and computer sciences* 28, pp. 31–36.

B. L. Welch (1947). "The generalization of student's' problem when several different population variances are involved". *Biometrika* 34, pp. 28–35.

D. S. Wigh, J. M. Goodman, and A. A. Lapkin (2022). "A review of molecular representation in the age of machine learning". *Wiley Interdisciplinary Reviews: Computational Molecular Science* 12, e1603.

G. Wu, E. Y. Chang, and Z. Zhang (2005). "An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines". In: *Proceedings of the Twenty-Second International Conference on Machine Learning*. Vol. 8.

J. Yan, P. Bhadra, A. Li, P. Sethiya, L. Qin, H. K. Tai, K. H. Wong, and S. W. Siu (2020). "Deep-AmPEP30: improve short antimicrobial peptides prediction with deep learning". *Mol. Ther. Nucleic Acids* 20, pp. 882–894.

N. Y. Yount, D. C. Weaver, E. Y. Lee, M. W. Lee, H. Wang, L. C. Chan, G. C. Wong, and M. R. Yeaman (2019). "Unifying structural signature of eukaryotic α-helical host defense peptides". *Proc. Natl. Acad. Sci. U.S.A.* 116, pp. 6944–6953.

T. Yu and H. Zhu (2020). "Hyper-parameter optimization: A review of algorithms and applications". *arXiv preprint arXiv:2003.05689*.

L. Yujian and L. Bo (2007). "A normalized Levenshtein distance metric". *IEEE Trans. Pattern Anal. Mach. Intell.* 29, pp. 1091–1095.

S. Zafeiriou (2012). "Subspace learning in Kreıĭn spaces: Complete kernel fisher discriminant analysis with indefinite kernels". In: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV 12*. Springer, pp. 488–501.

C. Zhan, B. Y. S. Li, Q. Wen, G. Ying, and T. Hao (2019). "Indefinite Kernels in One-Class Support Vector Machine and its Application on Virtual Screening". In: *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, pp. 1149–1155.

S. Zhang, H. Hu, T. Jiang, L. Zhang, and J. Zeng (2017). "TITER: predicting translation initiation sites by deep learning". *Bioinformatics* 33, pp. i234–i242.

Y. Zhao, X. Liu, H. Lu, X. Zhu, T. Wang, G. Luo, R. Zheng, and Y. Luo (2021). "An optimized deep convolutional neural network for yield prediction of Buchwald-Hartwig amination". *Chemical Physics* 550, p. 111296.

A. Zielezinski, H. Z. Girgis, G. Bernard, C.-A. Leimeister, K. Tang, T. Dencker, A. K. Lau, S. Röhling, J. J. Choi, M. S. Waterman, et al. (2019). "Benchmarking of alignment-free sequence comparison methods". *Genome Biol.* 20, pp. 1–18.

A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowski (2017). "Alignment-free sequence comparison: benefits, applications, and tools". *Genome Biol.* 18, pp. 1–17.

A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller (2000). "Engineering support vector machine kernels that recognize translation initiation sites". *Bioinformatics* 16, pp. 799–807.