INTEGRATION OF TOPOLOGICAL FINGERPRINTS AND MACHINE LEARNING FOR THE PREDICTION OF CHEMICAL MUTAGENICITY

By

Yin Cao

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Mathematics — Doctor of Philosophy

2017

ABSTRACT

INTEGRATION OF TOPOLOGICAL FINGERPRINTS AND MACHINE LEARNING FOR THE PREDICTION OF CHEMICAL MUTAGENICITY

By

Yin Cao

Toxicity refers to the interaction between chemical molecules that leads to adverse effects in biological systems, and mutagenicity is one of its most important endpoints. Prediction of chemical mutagenicity is essential to ensuring the safety of drugs, foods, etc. *In silico* modeling of chemical mutagenicity, as a replacement of *in-vivo* bioassays, is increasingly encouraged, due to its efficiency, effectiveness, lower cost and less reliance on animal tests.

The quality of a good molecular representation is usually the key to building an accurate and robust *in silico* model, in that each representation provides a different way for the machine to look at the molecular structure. While most molecular descriptors were introduced based on the physio-chemical and biological activities of chemical molecules, in this study, we propose a new topological representation for chemical molecules, the combinatorial topological fingerprints (CTFs) based on persistent homology, knowing that persistent homology is a suitable tool to extract global topological information from a discrete sample of points. The combination of the proposed CTFs and machine learning algorithms could give rise to efficient and powerful *in silico* models for mutagenic toxicity prediction. Experimental results on a developmental toxicity dataset have also shown the predictive power of the proposed CTFs and its competitive advantages of characterizing and representing chemical molecules over existing fingerprints.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advisor, Dr.Guowei Wei for his guidance, patience, motivation, encouragement, immense knowledge, and hearty support in all the phases of my doctoral program at Michigan State University.

I would like to thank my committee members, Dr. Yingda Cheng, Dr. Chichia Chiu, and Dr. Yiying Tong for taking time to serve on my dissertation committee and for their insightful comments and encouragement.

I am grateful to all my group members and friends, David Bramer, Zixuan Cang, Lin Mu, Kristopher Opron, Bao Wang, Menglun Wang, Kedi Wu, Kelin Xia, Zhixiong Zhao for the valuable discussions and their friendship.

With a special mention to Liqian Cai, Jiayin Jin, Ji Li, Yu Liang, Qiliang Wu, Shijian Zhuang and many others for their trust and support.

Last but by no means the least, my heart-felt appreciations go to my family, especially my mother, and also to my girlfriend Shengyin Si, for their unconditional love and support throughout writing this thesis and my life in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

KEY TO ABBREVIATIONS

PH	Persistent Homlogy
TDA	Topological Data Analysis
DNN	Deep Neural Network
CTF	Combinatorial Topological Fingerprint
SVM	Support Vector Machine
RF	Random Forest
ET	Extra Trees
GBT	Gradient Boosting Trees
GD	Gradient Descent
SGD	Stochastic Gradient Descent
BBR	Binned Barcodes Representation
MOE	Molecular Operating Environment

Chapter 1

Introduction

To date, hundreds of *in silico* predictive models have been published in the literature for chemical genotoxicity, of which mutagenicity being the most commonly modeled endpoint. Mutagenicity refers to a chemical or physical agent's ability to induce permanent transmissible changes in the amount or structure of the genetic material, usually DNA, in cells or organisms. Mutagenic activities of chemicals are examined by Ames test, which is a widely employed bioassay performed on bacterial strains of Salmonella typhimurium where each bacterial strain is sensitive to certain chemical mutagen(s) [1]. The mutagenic potential of a chemical is a highly important safety liability to determine in toxicological risk management, whose goal is to protect humans from exposure to mutagenic chemicals that may potentially cause cancer.

The use of in silico prediction systems has become increasingly prevalent due to the large number of chemicals of potential concern and tremendous cost (in time, money, animals, etc.) of rodent mutagenicity bioassays. In silico methods train predictive models based on structures with known mutagenic or non-mutagenic activities determined from mutagenicity bioassays, and could examine large numbers of queried structures and predict their mutagenic activities at high speed. It has a unique advantage of being able to estimate chemicals for mutagenicity even before they are synthesized. A lot of effort has been made to improve the application of in silico methods for mutagenicity prediction to assist industry and regulators and to enhance protection of public health. With modern technology and generation of new data, in silico methods have shown promise for a much deeper understanding of the mutagenic mechanism that involves complex, cellular processes which are only partially understood.

For over 25 years, it has been known that molecular structures correlate with mutagenic activity outcomes. The electrophilic theory introduced by James and Elizabeth Miller [2] has promoted the use of (Q)SAR in the prediction of genotoxicity and mutagenicity. Basically, the electrophilic theory states that, in general, mutagenic chemicals have the unifying feature that they are either electrophiles or can be activated to electrophilic reactive intermediates (pro-electrophiles). Two approaches for modeling genotoxicity and mutagenicity has developed: a) knowledge-based SAR models that try to identify electrophilic functional groups or substructures (structural alerts or toxicophores); b) in silico QSAR models that based on molecular descriptors that can be quantitatively related to genotoxic and mutagenic outcome of queried chemicals.

A lot of expert systems [3] based on SAR models have developed providing pre-built rule-based and SA lists, such as Derek Nexus [4, 5, 6], Toxtree [7], Oncologic Cancer Expert System [8], etc. In general, a SAR model is easy to interpret and implement, it is used in drug design to determine how drugs should be altered to reduce their adverse or undesirable effects. But a SAR model is a coarse-grained approach, it can only provide a qualitative but not a quantitative relationship between structure and activity, which may not be sufficient and could cause a large number of false negatives (i.e., mutagenic chemicals predicted as non-mutagenic) due to incompleteness of rules and SAs list in practice [9, 4, 3]. While QSAR models could provide a more precise means of predicting genotoxicity and mutagenicity, however mainly for congeneric sets of chemicals (local QSAR). There are two main steps to develop a QSAR model, generating a list of molecular descriptors and building a model to fit the data. QSAR models are made available in tools such as OECD QSAR Toolbox [10], Derek Nexus [4, 5, 6], HazardExpert [4, 11], CAESAR [12, 13], T.E.S.T. [14], etc.

Statistical and machine learning methods have gained a lot of popularity over the past few years, such as linear models that based on linear and multiple linear regression and discriminant models; non-linear models like artificial neural networks [15, 16, 17, 18], support vector machines [19, 20, 21, 18], decision trees [22], clustering, Naive Bayes, and K-nearest neighbor [23]. Overviews of cheminformatics and (Q)SAR modeling as well as historical perspectives can be found in Gasteiger and Engel [24], Tropha et al [25], Gillet and Leach [26], and Cherkasov et al [25].

The Toxicity Testing in the 21st Century (Tox21) joins efforts among Environmental Protection Agency (EPA), National Institute of Health (NIH), including National Center for Advancing Translational Sciences and the National Toxicology Program (NTP) at the National Institute of Environmental Health Sciences, and the Food and Drug Administration. The goal of the challenge is to develop toxicity prediction models for toxicology assessment accurately, cost-effectively and efficiently. Since the inception of the program in 2008, approximately 10000 chemicals and drugs in about 50 quantitative high-throughput screening (HTS) bioassays have been tested. The Tox21 Data Challenge held by NIH in 2014 has been the largest effort of the scientific community to compare and evaluate computational models in toxicology assessment. Approximately 12000 environmental chemicals and drugs are provided in SDF and SMILE format, with 12 different toxic effects categorized in two classes (Nuclear Receptor Panel and Stress Response Panel). Each chemical compound has multiple possible toxic effects (some toxic effects might be missing).

The grand challenge winner (DeepTox) [27] employed Deep Neural Network (DNN) and Multi-task learning, and won the total of 9 of the 15 challenges. It used the adjusted logistic loss function designed to cope with multi-task learning with missing labels. The chemical descriptors they used included several thousands of static descriptors such as atom counts, surface areas, presence or absence of 2500 toxicophores (structural alerts), weights, Van der Waals volume, partial charge, MACCS fingerprints and many other topological and physical properties, and hundreds of millions of dynamic descriptors obtained by "JCompoundMapper". The final model is a probabilistic consensus model that contains DNN and other complementary models such as Support Vector Machines, Random Forest and Elastic Net. Team AMAZIZ won the second place in the challenge [28]. The Associative Neural Network (ASNN) was employed to construct all models. ASNN is also a deep learning method (multilayered perceptron neural network) that utilizes ensemble learning. The descriptors were selected from ten descriptor packages from OCHEM, such as GSFlag, Chemaxon, Estate indices, CDK, and Dragon 6, 3D descriptors were obtained from optimized 3D structures by CORINA software. The final consensus model is the combination of over 12000 models, which obtained the best balanced accuracy across 12 analyzed targets. Team dmlab got the third place in the challenge bt using the multi-Tree ensemble method, which is the best non DNN method among all participants. The descriptors used in their work include 1444 2D descriptors and 881 binary PubChem fingerprints generated by the PaDel package, as well as 117 descriptive attributes, 118 Gasteiger charges, and 1024 Avalon fingerprints. Random forest classifier, extra trees classifier, gradient boosting trees classifier, and support vector machines were examined thoroughly, among which random forest and extra trees classifiers produced the most reliable predictive results. The great success of deep learning methods, especially multi-task DNN, in the Tox21 data challenge and many other fields motivates us to incorporate it in our modeling process.

Both SAR and QSAR methods use numerical representation of chemicals for modeling, the choice of a good representation for each queried chemical is crucial for both models to generate accurate predictions. Over the past decades, researchers have been searching a good molecular representation to convert structural information into molecular descriptors and to establish relationships between molecular structures and properties [29, 30, 31]. Thousands of molecular descriptors have been introduced and can be obtained by many dedicated software [32, 33, 34, 35, 36]. The number of available molecular descriptors is still growing with the increase of the complexity of investigated chemical systems. But all representation lose information to some extent, as each molecular descriptor describes only a little part of the specific molecule. Besides, most of the current molecular descriptors are related to the physiochemical and biological properties of the molecules, where the mutual connection within each molecule is overlooked. A number of topological indices (also known as connectivity indices) were introduced based on the molecular graph of a chemical molecule, such as the Hosoya index (also referred to as "the" topological index), Wiener index, Randic's molecular connectivity index, Balaban's J index, etc [37, 38, 39, 40, 41, 42], however, the discrimination capabilities of these topological indices vary a lot and a comprehensive topological molecular representation is still of great concern.

Mathematically, topology studies properties of certain objects that are invariant under continuous deformations such as tunnels or rings, and cavities or voids. Topology is concerned with mutual connectivity of components in space [43], however the over-simplification and excessive abstraction of conventional topology makes it difficult for topological features to capture detailed molecular information and limits its implementations in biomolecular analysis. In contrast, geometric features are usually computationally expensive and contain too much redundancy to be practical. Persistent homology (PH) is a new branch in algebraic topology, which is widely used in topological data analysis (TDA) to extract qualitative characteristics of data that persist across multiple scales [44, 45, 43]. TDA is an interdisciplinary field that combines data analysis, algebraic topology, computer science, statistics, machine learning and other related fields, whose goal is to develop tools for examining qualitative features of data. PH provides not only precise definitions and some robustness of these qualitative features, but also computational tools in practice. PH is able to examine data types such as finite metric spaces (point cloud data), digital images, level sets of real valued functions, networks and so on. It embeds multiscale geometric information into topological invariants to bridge the gap between topology and geometry [44, 45]. The persistent diagram is visualized through persistent homology barcodes, in which horizontal line segments (bars) are the homology generators lasted over filtration scales. Since the first computational algorithm of PH that was introduced by Edelsbrunner and coworkers in 2000 [46], a number of algorithms and optimization techniques have been proposed, with new software released at increasingly fast pace. Open source libraries, such as Dionysus [47], JavaPlex [48], PERSEUS [49], DIPHA [50], and GUDHI [51] are widely used in practice and have boosted the implementation of PH in various applications including data analysis [52, 53], image analysis [54, 55, 56, 57], sensor networks [58], computer vision [59], shape recognition [60], and computational biology [61, 62]. The need of a more comprehensive molecular representation in the biomolecular model generation and the success of computational persistent homology in various fields [63, 64, 65, 66, 67, 68] motivate us to develop and design a new type of topological fingerprints that could serve the purpose.

The goal of this work is to further explore the application of PH for chemical toxicity prediction, especially two of the most important endpoints, the Ames mutagenicity, and developmental toxicity. We introduce the PH based Combinatorial Topological Fingerprints (CTF) as a unique and comprehensive molecular representation to characterize, identify, and classify chemical molecules. The design of the proposed CTF is also motivated by our attempt to get a deeper understanding of the topology-function and structure-function relationships of biomolecules.

The rest of the paper is organized as follows. In Chapter 2, we will introduce the datasets used in our work, including the details of how we prepare and preprocess the molecular structures. In Chapter 3, we will introduce the fundamental concepts of persistent homology including simplices, simplicial complex, chains, homology, filtration, and some popular computational algorithms and so on. In Chapter 4, firstly, we will provide the details of the computation of the proposed Combinatorial Topological Fingerprints (CTF) as well as some preparatory analysis; secondly, we will introduce some auxiliary features used in our work, including features from the Feature Functional Theory [69, 70], and molecular descriptors generated from relevant software such as RDKit [34], PyDPI [71], and so on. In Chapter 5, we will introduce some classic machine learning algorithms such as Support Vector Machines, Neural Networks, and ensemble methods like Gradient Boosting Trees, Random Forest, and Extra Trees Classifiers. In Chapter 6, we will report the predictive performances of our model on two external independent testing datasets AID1189 and AID1194. The predictive performance of the proposed CTF on the developmental toxicity dataset, including the comparison with some existing molecular fingerprints, are also reported in this chapter. The paper will then end with a discussion and conclusion.

Chapter 2

Biological background

2.1 Mutagenicity

Mutagens are substances that change the genetic information of an organism, usually DNA, and therefore are genotoxic. Mutagens induce mutations in the DNA and can affect the transcription and replication mechanisms of the DNA, which could lead to cell death in severe cases. Different mutagens interact with the DNA differently. Many mutations are silent mutations, causing no or minor visible effects at all, while some mutations are lethal and could cause serious disease. Mutations can even change the number of chromosomes in the cell (aneuploidy).

The Ames test [72] (named after its developer, Bruce Ames, 1970) is a widely employed bioassay that uses bacteria to determine if a chemical is a mutagen. Compared with expensive and time-consuming standard bioassays on mice and rats, the Ames test is fast, easy, and inexpensive for screening environmental substances for potential mutagens. The strains of the bacteria Salmonella Typhimurium used in the test carry a type of mutant gene that makes them unable to synthesize the amino acid histidine, without which the strains will die. The assay will add chemicals to the media to be tested for mutagenicity. Secondary mutations occur at a low spontaneous rate, enable some strains of bacteria to grow just fine in the histidine-free media. The number of surviving colonies in the media reflects the mutagenic effect of the corresponding chemical.

There are thousands of known chemical mutagens and they could affect the genetic materials in many different ways: some mutagens can resemble the nucleobases (adenine, thymine, cytosine, and guanine) found in normal DNA, some can alter the structures of existing bases, some can insert themselves in the helix between bases, while some can interact with genetic materials indirectly by creating reactive compounds that damage the DNA structure. As mutagens usually cause cancer, they are usually also carcinogens. Common carcinogens include benzene, vinyl chloride, formaldehyde, dioxane, and acrylamide, common mutagens include ethidium bromide, formaldehyde, dioxane, alkaloid, psoralen, and nicotine.

Base analogs are molecules whose chemical structure is similar to one of the four DNA nucleobases (adenine, thymine, cytosine, and guanine). The similarity enables them to be incorporated into the helix during DNA replication. The mutagenic base analogs could pair with more than one other nucleobases, which will cause genetic mutations during the next round of replication when the replication machinery tries to pair a new base with the incorporated mutagen. One of the most common base analogs is the 5-bromouracil (5BU, CAS number: 51-20-7), the abnormal base found in the mutagenic nucleotide analog BrdU, which is mainly used as an experimental mutagen. 5BU exists in three tautomeric forms (keto form, enol form, and ion form) that have different base-pairing properties. The keto form is complementary to adenine and can be incorporated into DNA by aligning opposite adenine residues during DNA replication, while the enol and ion forms are complementary to guanine. The three tautomeric forms frequently interchange so base-pairing properties also change frequently. In its keto form, 5BU can be incorporated across from an adenine. If it then tautomerizes to its enol form, during a subsequence round of replication, it will cause a guanine to enter the opposite strand, rather than the correct adenine. This results in a change in one base pair of DNA, especially a transition mutation.

Some mutagens act by chemically modifying the nucleobases of DNA. For instance, nitrite preservatives in food convert to the mutagen nitrous acid (HNO_2), which reacts with nucleobases that contain amino groups. Adenine is oxidatively deaminated to hypoxanthine (which is now complementary to cytosine), cytosine is oxidatively deaminated to uracil (which is now complementary to adenine), and guanine is oxidatively deaminated to xanthine which is still complementary to cytosine. Unlike base analog mutagens, base-altering mutagens directly alter a nucleobase without requiring subsequent DNA synthesis for its mutagenic effect.

Intercalating agents are flat, multiple ring molecules that could insert themselves between adjacent nucleobases in DNA, generating shape deformation at the intersection point. This may cause the DNA polymerase add an additional nucleobase opposite the intercalating agent (making the synthesis/repair systems think there is another base at that position.). If this occurs in a gene, it induces a frameshift mutation, which alters the reading of the gene transcript and changes the amino acids added to the encoded protein. Ethidium bromide (CAS number: 1239-45-8) is one such intercalating agent that is widely used in DNA research. For instance, it is commonly used as a fluorescent tag (nucleic acid stain) to detect nucleic acids in molecular biology laboratories such as agarose gel electrophoresis to find the DNA bands that have been separated in a gel.

Alkylators (or alkylating antineoplastic agents) are powerful mutagens used in almost all biological system. They react with DNA by attaching an alkyl group (C_nH_{2n+1}) to the guanine base at the number 7 nitrogen atom of the purine ring. This, in turn, inhibits their correct utilization by base pairing and causes a miscoding of DNA. Alkylators cause damage to DNA via several mechanisms:

- Attaching alkyl groups to DNA bases results in the DNA fragmented by repairing enzymes in their attempts to replace the alkylated bases.
- Formation of cross-links (bonds between atoms) in the DNA prevents DNA from being separated for synthesis or transcription.
- Causing mispairing of the nucleotides leading to mutations.

Because of the damaging effect of alkylators to genetic materials, they are used in cancer treatment, since in general, cancer cells proliferate with less error-correcting than healthy cells thus are more sensitive to DNA damage. Unfortunately, most of the alkylators are also carcinogenic, since they could also cause damage to normal cells, particularly cells that divide frequently.

Other mutagens include chemicals that create free radicals inside a cell, which are atoms or groups with an odd (unpaired) number of electrons and can be formed when oxygen interacts with certain molecules. These free radicals are highly reactive and can react with important cellular components such as DNA and cell membrane. This can start a chain reaction and cause several types of damage to DNA and cell [73].

2.2 Developmental toxicity

Developmental toxicity is another important toxic endpoint. It includes any abnormal structural or functional effects interfering with normal development, present both in infancy or later in life of the organism. Developmental biology is a subject to study the process by which animals and plants grow and develop, which shared a long history with the subject of teratology. It was recognized that genetic, diet, nutritional, infectious (e.g. rubella), and environmental chemicals (e.g. dioxins, tobacco smoke) could cause congenital abnormalities and perturbed development in humans and various animal systems [74].

In the 1950s, two very rare abnormality disease of human limbs, phocomelia and amelia, affected thousands of infants born in several European countries and Australia. It was then concluded by physicians that the sudden increase of this abnormality disease is due to treatment with the thalidomide (a pharmaceutical sedative and hypnotic) in the mothers of the affected babies in the early stage of their pregnancies. The abandon of thalidomide in the international market brought the epidemic to an end, and for the first time, it was tragically proved that a chemical agent had the potential to profoundly affect human development. Until then, the study of the developmental toxicity of chemicals entered the public's attention.

People soon recognized that the toxicity experimental tests for drugs were inadequate to predict the response of the embryo or fetus since they were primarily tested on adults. Guidelines for assessing and regulating chemical developmental toxicity were made by the U.S. Food and Drug Administration (FDA) and similar regulatory agencies in the U.S. and around the world, which were applied to not only to pharmaceuticals but to all chemicals with significant human exposure potential. Determine toxicity associated with abnormal development and recognition of the developmental toxic effects of various chemicals become more and more important in modern society. The field of developmental toxicity is rapidly growing, with the increasing need to understand the underlying chemical mechanisms induced developmental defects.

However, in general, there are still myriad of difficulties to understand mechanisms in sufficient details and depth for risk assessment and preventive public health purposes [74]. First of all, the developmental mechanisms are extremely complicated due to the fact that a developmental toxicant could interact with an important molecular component at many points and our study about molecular components and developmental processes are still in the early development stage. Furthermore, developmental toxicants include a wide range of chemical, physical, and biological agents, which could induce various mechanisms affecting different levels of biological systems.

Toxicants interact with cellular molecules in many different forms:

1. Chemicals may act as ligands for endogenous receptors and interact with receptors in various ways. First, the interaction may mimic the endogenous ligand and activate the receptor inappropriately causing agonist-like actions. Second, chemicals may bind to the receptor without causing resultant activation, however, will inhibit and block the binding of an endogenous ligand (antagonists). Third, chemicals may produce allosteric effects on receptors by binding to an adjacent part of the macromolecule. Some macromolecules without endogenous ligands can also bind to specific toxicants, causing physiological changes.

2. Toxicants may interact chemically with an endogenous molecule by covalent binding (e.g. forming a DNA or protein adduct), causing errors in transcription or replication of DNA. One example of such developmental toxicant is diphenylhydantoin, which can form both DNA and protein adducts in embryos.

3. Some chemicals inhibit protein functions by interfering with enzymes whose catalytic function is important in development, some by blocking protein polymerization, others by binding to protein-protein association sites or other kinds of sites.

4. There are other mechanisms found to affect development. For example, chemicals as free radicals (or generate free radicals during their metabolism) can change structures

of proteins or lipids by peroxidation; chemicals can interfere with sulfhydryl groups, causing oxidative stress and inhibit the corresponding function.

Nowadays, on the one hand, the lack of knowledge about normal development and cell biology in different levels of biological organizations is still one of the greatest limiting factors in the study of the mechanisms of developmental toxicity; on the other hand, relevant and accurate data and examples where the molecular, cellular, and developmental information is complete are far from sufficient to commendably validate our hypotheses.

Chapter 3

Data

3.1 Databases

Both the utilization of existing toxicity data and the generation of new data have fueled much of the advancements in this field. A number of public toxicity databases have developed over the last decades to serve as data mining resources.

For example, the Carcinogenic Potency Database (CPDB) was developed by the Carcinogenic Potency Project at the University of California, Berkeley and the Lawrence Berkeley National Laboratory, which contains 6540 chronic, long-term animal cancer tests from published literature, the National Cancer Institute, as well as the National Toxicology Program (NTP).

EPA's Distributed Structure-Searchable Toxicity (DSSTox) project [75] focuses more specifically on accurate and consistent structure annotation of toxicological experiments, as well as the construction of summary toxicity endpoint data. It provides a high-quality public chemistry resource for implementation in QSAR, including the CPDB-All Species (CPDBAS) Summary Tables, a number of new species-specific summary activity fields, a species-specific normalized score for carcinogenicity and mutagenicity (TD50) and sex/species-specific cancer incidences. Both the CPDB and the online NTP are "chemically-indexed" in the DSSTox. The summary activity fields and associated structural data are deposited in the large and public PubChem database as seven bioassays or PubChem AIDs (PubChem Assay Identifiers), including CPDBAS Salmonella Mutagenicity, Hamster, Rat, Mouse, Dog_Primates, SingleCellCall, and MultiCellCall. EPA's ToxCast (Toxicity ForeCaster) program [76] is part of the Toxicity in the 21st Century (Tox21) federal collaboration, it is a chemical prioritization research program to develop models for prediction toxicity using bioactivity profiling. ToxCast contains EPA's most updated publicly available high-throughput toxicity data on thousands of chemicals. The quantitative high-throughput screening (qHTS) approach is utilized to test thousands of environmental and commercial chemicals against hundreds of bioassays that are potentially relevant to toxicity pathways. All data are downloadable in EPA's website [77].

There are many other databases, such as TOXNET of the US National Library of Medicine (NLM), GENE-TOX that specifically deals with mutagenicity and carcinogenicity data, ToxRefDB developed by the NCCT that captures over 30 years and $2 billion of animal testing results, just to name a few.

3.2 Preprocessing

In this study, the training data is collected from two resources, one of them being the Bursi's mutagenicity dataset [78] containing 4337 molecules and the other one being the Hansen's Benchmark dataset [79] containing 6512 molecules, both of which are publicly available and downloadable in 2D SDF or SMILES format.

As our proposed CTF require 3D molecular structures of each chemical, an accurate and efficient chemical structure conversion from 2D to 3D is therefore a key precursor to our study. Here we use the Schrodinger's LigPrep software [80], which enable us to convert the original 2D structures into 3D low-energy structures in MOL2 format. The optimized 3D structures are then processed by the Amber molecular simulation package [81] with GAFF force field [82] and converted into 3D structures in PQR format. Our in-house software, MIBPB5 [83] and ESES [84] are then employed to obtain the atomic area, volume and solvation free energy information for each chemical molecule. The preprocessing steps for the molecular structures are illustrated in Figure 3.1. We have to discard those ill-structured molecules that could not be processed by fore-mentioned software, and we end up getting 4323 and 6468 3D structures in PQR format for the Bursi's mutagenicity dataset and the Hansen's Benchmark dataset, respectively. The training set in this study is the combination of the two, with duplicates removed according to the CAS numbers of the chemicals, resulting in 7032 3D molecular structures in PQR format in total.

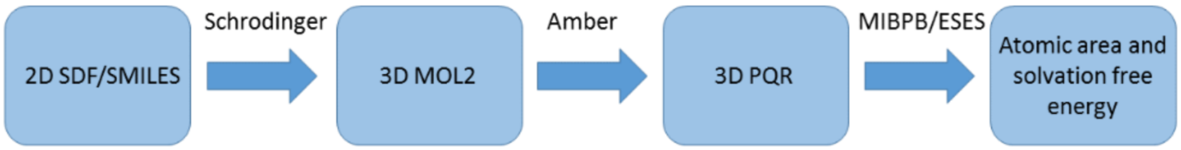


Figure 3.1: Illustration of the molecular structural data preprocessing steps.

Two independent external data sets, namely AID1189 and AID1194, are used as the testing sets. Both datasets are downloadable from EPA DSSTOX dataset in the Carcinogenic Potency Database (CPDB) [85, 86], which is a unique and widely used database that contains over 6500 bioassay cancer test results on over 1500 chemicals. The AID1189 dataset (Summary SingleCellCall Results) is based on experimental bioassay results of one or more species, which contains 1544 molecules with 806 mutagens 738 non-mutagens. The AID1194 dataset (Salmonella Mutagenicity) is a summary mutagenicity call (active and inactive) based on the Salmonella assay, which contains 860 molecules with 403 mutagens and 457 non-mutagens. The 2D structures are then downloaded from the PubChem website [87, 88] using the PubChem Download Service [89], resulting in 1480 and 832 2D structures in SDF format, respectively. By removing molecules with multiple components, such as mixtures and salts, and other molecules that do not hold unique CID(PubChem Compound Identifier) registry numbers, the numbers of molecules in AID1189 and AID1194 are reduced to 1460 and 818. Just like how we preprocess the molecules in the training set, these structures are then optimized by Schrodinger and processed by Amber and our in-house software MIBPB and ESES. Finally, 1431 and 805 3D structures in PQR format are obtained for AID1189 and AID1194. respectively. Table 3.1 shows the distribution of molecules in the training and testing sets mentioned above.

Mutagenicity dataset	Total	Mutagen	Non-mutagen
Training set	7032	3771	3261
Testing set 1(AID1189)	1431	771	660
Testing set 2(AID1194)	805	388	417

Table 3.1: Distribution of datasets used for mutagenicity prediction. The training set is the combination of the Bursi's dataset and the Hansen's benchmark dataset with duplicates removed. Two independent external testing datasets, AID1189 and AID1194, are examined in this study.

To further test the representative ability of the proposed CTF, we have also studied a developmental toxicity dataset that contains 285 chemicals created by Arena and coworkers [90]. The developmental toxicity outcomes were taken from the revised binary toxicity outcomes developed from CAESAR project [13]. The dataset is split into two parts by EPA for training and testing usages. The 2D molecular structures and CAS numbers of chemicals in the training and testing parts can both be found and downloaded from EPA's website [14]. Our prediction result is comparable with the best result reported by Toxicity Estimation Software Tool (T.E.S.T.) of EPA using Random Forest [14]. Table 3.2 shows the distribution of molecules in the developmental toxicity dataset.

Developmental toxicity dataset	Total	Developmental toxicant	Non-developmental toxicant
Training set	227	157	70
Testing set	58	41	17

Table 3.2: Distribution of the developmental toxicity dataset.

Chapter 4

Persistent homology

Unlike the field of geometry where exact measurements such as distance and angles are important, the field of topology studies properties of spaces where all it matters is the relationship of points to one another. Persistent homology examines topological structures of the data in a reliable way without having to adulterate the data in any way. Moreover, when analyzing high-dimensional data, persistent homology gives people a way to find interesting patterns in data without downgrading the data (unlike PCA or any other dimension reduction techniques) so people can visualize it.

In topological data analysis (TDA) or computational topology, a set of data points are replaced with a family of simplicial complexes indexed by a proximity parameter, after which, the topological invariants of these simplicial complexes are encoded in their persistent homology barcodes, a parameterized version of Betti numbers. In this chapter, the basic concepts of persistent homology and some popular computational algorithms will be introduced.

4.1 Simplex and Simplicial Complex

Let $u_0, u_1, ..., u_k$ be points in $\mathbb{R}^d$. A point $x = \sum_{i=0}^{k} \lambda_i u_i$ is called an affine combination of $\{u_0, u_1, ..., u_k\}$ if $\sum_{i=0}^{k} \lambda_i = 1$. The set of affine combinations is called the affine hull. It is a k-plane if the $k+1$ points are affinely independent by which we mean that for any two affine combinations $x = \sum \lambda_i u_i$ and $y = \sum \mu_i u_i$, $x = y$ iff $\lambda_i = \mu_i, \forall i$.

An affine combination $x = \sum \lambda_i u_i$ is a convex combination if all λ_i are non-negative. The set of convex combinations is called the convex hull. A **k-simplex** σ is the convex hull of $k+1$ affinely independent points:

$$\sigma = \{\sum_{i=0}^{k} \lambda_i u_i | \sum_{i=0}^{k} \lambda_i = 1; 0 \leq \lambda_i \leq 1, \forall i\} \tag{4.1}$$

We say that σ is spanned by u_0 to u_k, and write $\sigma = \{u_0, u_1, ..., u_k\}$ for short. Its dimension is dim$\sigma = k$. The simplices of a few low dimensions are known as: vertex for 0-simplex, edge for 1-simplex, triangle for 2-simplex, and tetrahedron for 3-simplex (see Figure 4.1).

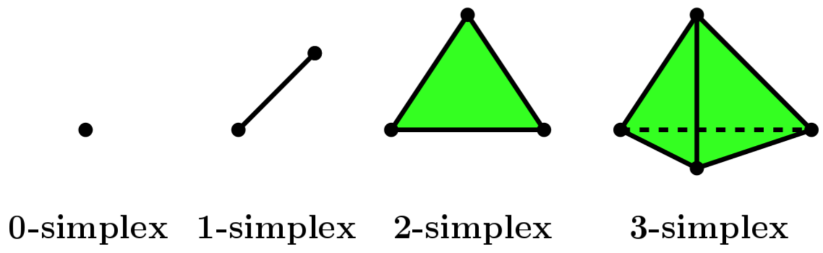


Figure 4.1: Illustration of basic simplices. Four types of simplices are displayed from left to right: a vertex, an edge, a triangle, and a tetrahedron.

A **facet** or **face** of σ is the convex hull of a non-empty subset of u_i and it is called proper if the subset is not the entire set. Specifically, the p-simplex τ spanned by any subset $\{u_{i_0}, u_{i_1}, ..., u_{i_p}\}$ is called a ***p*-face**. We write $\tau \leq \sigma$ when τ is a face of σ, and $\tau < \sigma$ when τ is proper. Since the number of subsets of a set containing $k+1$ elements is 2^{k+1} including the empty set, it is easy to see that σ^k has $2^{k+1} - 1$ faces, all of which are proper except for σ^k itself.

Given a k-simplex $\sigma = \{u_0, u_1, ..., u_k\}$, the **boundary** of σ, denoted by $\partial\sigma$, is defined to be the union of all $(k-1)$-faces. Formally,

$$\partial\sigma = \sum_{i=0}^{k}\{u_0, ..., \hat{u_i}, ..., u_k\} \tag{4.2}$$

where the hat symbol over u_i indicates omission of u_i.

The complement of the boundary is defined to be the **interior**, denoted by intσ, i.e. int$\sigma = \sigma - \partial\sigma$. A point $x \in$ intσ iff all of its coefficients λ_i are positive.

A **simplicial complex** K is a finite collection of simplices satisfying the following two conditions:

1. Every face of a simplex in K also belongs to K, i.e. $\sigma \in K$ and $\tau \leq \sigma$ implies $\tau \in K$

2. For any two simplices σ_1 and $\sigma_2 \in K$, if $\sigma_1 \cap \sigma_2 \neq \emptyset$, then $\sigma_1 \cap \sigma_2$ is a common face of both σ_1 and σ_2

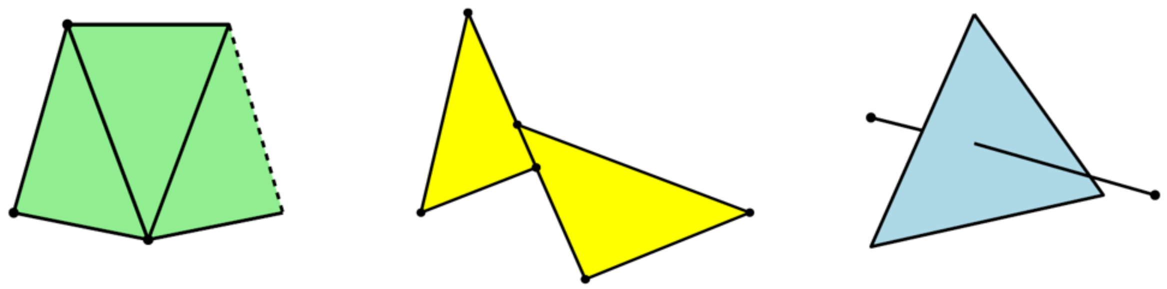


Figure 4.2: Collections of simplices that are not simplicial complexes. To the left there is a missed edge; in the middle, two triangles meet along a segment that is not an edge of either triangle; to the right the edge crosses the triangle at an interior point.

The dimension of a simplicial complex is defined to be the maximum dimension of any of its simplices. The underlying space, denoted by $|K|$, is the union of its simplices together with the topology inherited from $\mathbb{R}^d$.

4.2 Homology

Let K be a simplicial complex. A ***p*-chain** is a formal sum of p-simplices in K, i.e. $c = \sum \alpha_i \sigma_i$ where σ_i are p-simplices in K and in this work α_i is chosen in the field of $\mathbb{Z}_2$, that is 0 or 1. Simple examples of p-chains are illustrated in Figure 4.3.

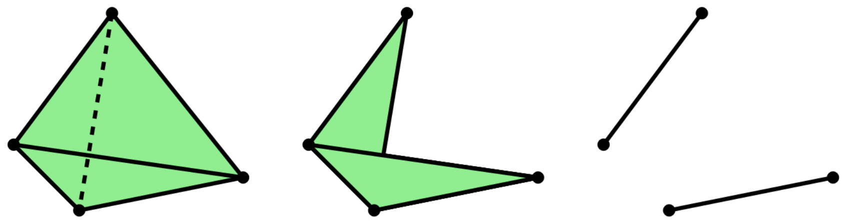


Figure 4.3: Given K = tetrahedron (left), a 2-chain (middle), and a 1-chain (right).

Two p-chains are added componentwisely, just like polynomials. Specifically, if $c_1 = \sum \alpha_i \sigma_i$ and $c_2 = \sum \beta_i \sigma_i$ then we have $c_1 + c_2 = \sum (\alpha_i + \beta_i)\sigma_i$, where the addition for the coefficients is the modulo 2 addition. Figure 4.4 gives a simple example of how to add two p-chains.

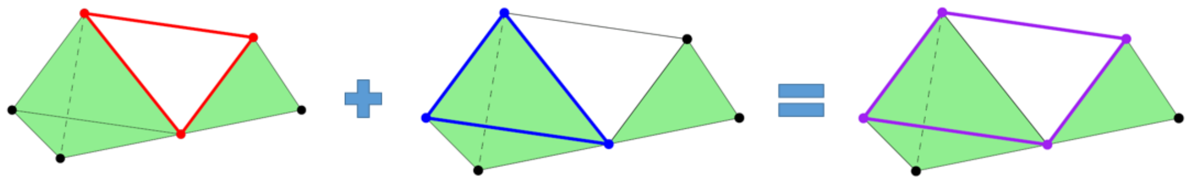


Figure 4.4: Mod-2 addition of two 1-chains. Simplices that are added by even times will cancel out. Two 1-chains are indicated in red and blue, the resultant sum is also a 1-chain, which is indicated in purple.

The set of all p-chains for a simplicial complex K, together with the addition operation defines an Abelian group $C_p(K, \mathbb{Z}_2)$, where the neutral element is $0 = \sum 0\sigma_i$.

For each integer p, we have a group of p-chains. This group is trivial for $p < 0$ and $p > \dim K$, consisting only of the neutral element. The boundary map defined in Eq. 4.2 extends to arbitrary p-chains linearly and relate these groups of p-chains. For a p-chain

$c = \sum a_i\sigma_i$, the boundary is the sum of boundaries of its simplices, i.e. $\partial_p c = \sum a_i \partial_p \sigma_i$. Writing C_p as the group of p-chains, ∂_p: $C_p \to C_{p-1}$ is a **homomorphism** since the boundary operator commutes with addition, i.e. $\partial_p(c_1 + c_2) = \partial_p c_1 + \partial_p c_2$. The **chain complex** is the sequence of chain groups connected by the boundary homomorphisms, i.e.

$$..., \xrightarrow{\partial_{p+2}} C_{p+1} \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}}, ... \tag{4.3}$$

We introduce two special types of groups, namely the **cycle group** and the **boundary group**, to define homology groups.

The k-th cycle group, Z_k and the k-th boundary group B_k are defined as follows:

$$Z_k = Ker\partial_k = \{c \in C_k \mid \partial_k c = 0\} \tag{4.4}$$

$$B_k = Im\partial_{k+1} = \{c \in C_k \mid \exists d \in C_{k+1} \; s.t. \; c = \partial_{k+1} d\} \tag{4.5}$$

Elements in the k-th cycle group and the k-th boundary group are called ***k*-cycle** and ***k*-boundary**.

Both cycle group Z_k and boundary group B_k are subgroups of the group of k-chain C_k. And since the groups of k-chains is Abelian, so are the corresponding subgroups.

The fundamental property that makes homology work is the following **Fundamental Lemma of Homology**:

$$\partial_p \partial_{p+1} d = 0 \tag{4.6}$$

for every integer p and every $(p+1)$-chain d. It follows that B_k is subgroup of Z_k, thus we have $B_k \subset Z_k$. Since Z_k is Abelian, B_k is normal.

This result has a geometric interpretation: every k-chain that is the image of some $(k+1)$-chain under the boundary map ∂_{k+1} has zero boundary under ∂_k, i.e. is a k-cycle. Moreover, The reverse inclusion is not necessarily true. For example, consider the simplicial complex containing only the edges and vertices of a triangle. The sum of of the three edges is a 1-cycle, but can not be the boundary of a 2-chain since there is no 2-simplex in the simplicial complex. As we can see from this simple example, the failure subfigure of the reverse inclusion is due to "holes" in the simplicial complex.

The k-th **Homology group** is defined as the quotient group of the k-th cycle group and the k-th boundary group:

$$H_k = Z_k / B_k \tag{4.7}$$

Two k-cycles are called homologous, which is denoted as $c_1 \sim c_2$, if they are different by a k-boundary. Basically, elements in the k-th homology group is an equivalent class of k-cycles.

The rank of the k-th homology group is called the k-th **Betti number**:

$$\beta_k = \text{rank}H_k = \text{rank}Z_k - \text{rank}B_k \tag{4.8}$$

A simple illustration can be seen in Figure 4.5.

Intuitively, the homology group captures the extent to which the inclusion $Z_k \subset B_k$ fails, and the k-th Betti number gives the number of k-dimensional holes in the simplicial complex.

4.3 Persistent Homology

Homology answers the question of the existence of topological invariants like tunnels, cavities in a 3D shape while regardless of their metric size. Persistent homology reintroduces the

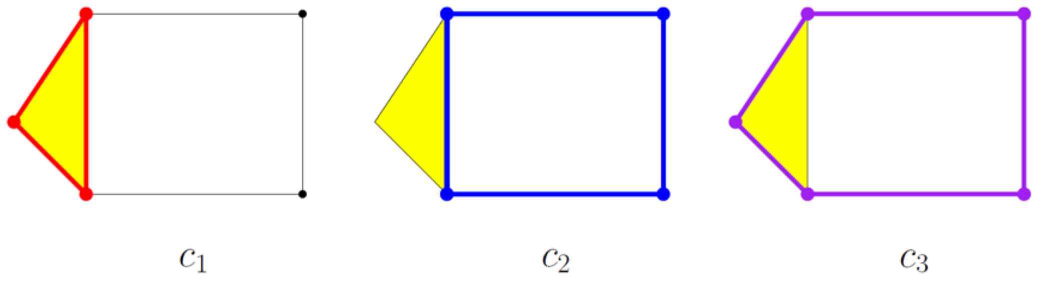


Figure 4.5: Illustration of cycle group, boundary group, and homology group. For the given simplicial complex, there exist three 1-cycles, namely, c_1, c_2, and c_3 indicated in red, blue, and purple colors, respectively. Among them, only c_1 is a 1-boundary. For this example, the 1st boundary group is $B_1 = \{0, c_1\}$; the 1st cycle group is $C_1 = \{0, c_1, c_2, c_3\}$; the 1st homology group $H_1 = \{\{0, c_1\}, \{c_2, c_3\}\}$. H_1 is isomorphic to $\mathbb{Z}_2$, thus the 1st Betti number $\beta_1 = \text{rank}(\mathbb{Z}_2) = 1$.

measurement of the scale or resolution of a topological structure. Also, from a combinatorial perspective, a set of points can form many possible complexes, Persistent Homology provides a framework to find the one that accurately reflects the topological properties of the object from which the sample originated.

Let K be a simplicial complex. A **filtration** is a nested sequence of subcomplexes,

$$\emptyset = K_0 \subset K_1 \subset ... \subset K_n = K \tag{4.9}$$

We may think of the filtration as a description of how to construct K by adding chunks of simplices at a time. The topological evolution of this sequence of subcomplexes could be described by the corresponding sequence of homology groups connected by homomorphisms:

$$0 = H_p(K_0) \to H_p(K_1) \to ... \to H_p(K_n) = H_p(K), \tag{4.10}$$

one for each dimension p.

We simplify the notation by writing $H_p^i = H_p(K_i)$, we then have:

$$0 = H_p^0 \to H_p^1 \to ... \to H_p^n \tag{4.11}$$

The homomorphisms can be composed giving maps $f_p^{i,j}$: $H_p^i \to H_p^j$. The image of $f_p^{i,j}$ consists of all p-dimensional homology classes that are born at or before K_i and die after K_j. Adding the zero homology group in the end guarantees every class eventually dies.

The dimension p persistent homology groups are the images of the homomorphisms induced by inclusion, $H_p^{i,j} = Imf_p^{i,j}$ for $0 \leq i \leq j \leq n+1$. The corresponding dimension p persistent Betti numbers are the ranks of these groups $\beta_p^{i,j} = rankH_p^{i,j}$. Formally,

$$H_p^{i,j} = Z_p^i / (B_p^j \cap Z_p^i) \tag{4.12}$$

where Z_p^i and B_p^j are the p-th cycle group and boundary group of K_i and K_j, respectively.

The definition here are analogous to those in simplicial homology group introduced in the last part. The form of $H_p^{i,j}$ suggests that the dimension p persistent homology group characterizes the p-cycles in the K_i subcomplex that are not the boundary of any $(p+1)$-chain from a larger complex K_j. Intuitively, $H_p^{i,j}$ characterizes the k-dimensional holes in K_j created by subcomplex before K_i, i.e. these holes persist in all complexes K_t for $i \leq t \leq j$

4.4 Filtered simplicial complexes

In computational topology, a collection of unordered points x_α in $\mathbb{R}^d$ (point cloud) is a common object for investigation. The global "shape" of the data points may provide important information about the underlying characteristic of the data. When the points are considered as vertices of a graph whose edges are determined by proximity, one completes the graph to a simplicial complex.

Some of the most commonly used methods of doing so are described as follows:

- **Čech complex**: given a set of points x_α in $\mathbb{R}^d$, the Čech complex (also known as nerve), denoted by C_ϵ, is the abstract simplicial complex where a set of $k+1$ vertices spans a k-simplex whenever the $k+1$ corresponding closed $\epsilon/2$-ball neighborhoods have nonempty intersection.

- **Alpha complex**: this is a variant of the Čech complex, when replacing the $\epsilon/2$-ball with the intersection of Voronoi cells and the balls for these data points.

- **Vietoris-Rips complex**: given a set of points u_α in $\mathbb{R}^d$, the Vietoris-Rips complex, denoted by $VR(\epsilon)$, is the abstract simplicial complex where a set S of $k+1$ vertices spans a k-simplex whenever the distance between any pair of points in S is at most ϵ. Formally,

$$VR(\epsilon) = \{\sigma | \text{diam}(\sigma) \leq \epsilon\} \tag{4.13}$$

In this study, the basic Euclidean distance-based filtration process is applied. We consider the open balls centered at each point in the point cloud and make the radius of the balls increase over time. Based on the different complex construction criterions described before, the simplicial complex will form and grow over time. Since there exists inclusion relation between the previously formed complex and the latter ones, the filtration is developed naturally. Figure 4.6 illustrates the difference between the construction of the Čech complex and the Vietoris-Rips complex. An illustration of the filtration process can be found in Figure 4.7.

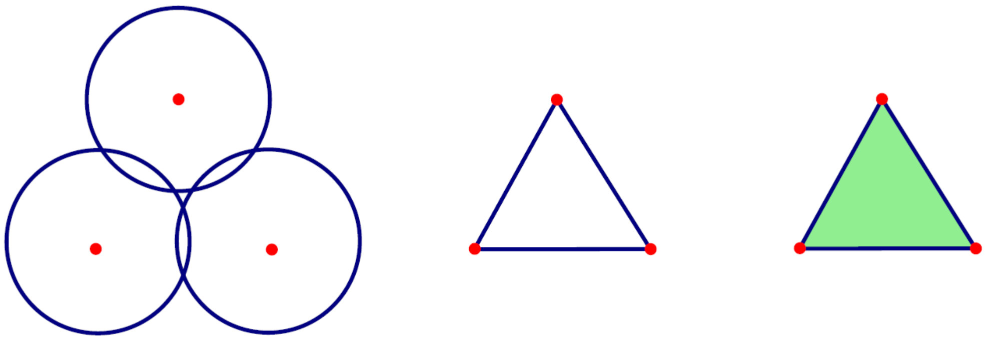


Figure 4.6: Illustration of the difference between the Čech complex and the Vietoris-Rips complex. Given three pairwise equidistant vertices in $\mathbb{R}^2$, which are denoted by three red dots, the Čech and Vietoris-Rips complexes of the covering on the left each have three edges. The Čech complex in the middle does not have any 2-simplices since three disks do not have a common intersection. However, the Vietoris-Rips complex (right) includes the green triangle since the distance between any pair of the vertices is less than ϵ.

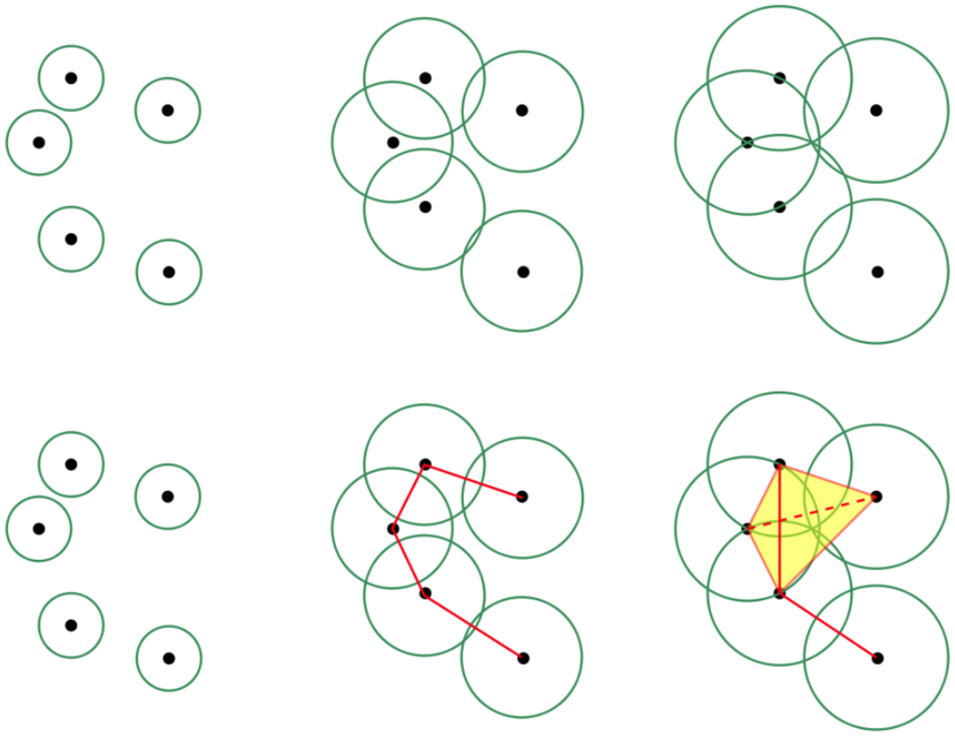


Figure 4.7: A 2D example to illustrate how the Vietoris-Rips complex is built as the filtration radius is increasing.

Take the Vietoris-Rips complex for example, the filtration is generated by an increasing sequence of ϵ, which is a nested sequence of simplicial complexes:

$$VR(\epsilon_1) \subset VR(\epsilon_2) \subset VR(\epsilon_3) \subset ... \tag{4.14}$$

From the persistent homology barcodes, we could be able to visualize at what value of ϵ does a "hole" appear (born), and how long does it persist till it is filled in (die).

The computation of dimension 0, 1, and 2 Betti numbers were done using the free software "Dionysus" [47]. For the Čech complex, one has to check for a large number of intersections, which makes its construction more expensive. Thus in our study, we only use the Alpha complex and the Vietoris-Rips complex. Since in general, the predictive performances of models built by using the Vietoris-Rips complex are better than that of the Alpha complex, we only report results from using Vietoris-Rips complex in the paper.

Chapter 5

Features

In machine learning and data mining, a feature is an individual measurable value or property of a characteristic being observed. Each sample (in our case chemical molecule) to be studied is transformed into a feature vector that embeds the essential information from various perspectives. Feature engineering is the key to success in any machine learning and data mining modeling. Well-conceived features could capture the important biomolecular information more effectively, and the design of such features requires a combination of experimentation and learning.

Persistent homology is a perfect tool characterizing structural information of a given set of points. The persistent homology barcodes encode the intrinsic molecular properties such as molecular substructures, chemical bonds and connectivities. The correlation between biological effects such as toxic activities and molecular structures enables the development of the proposed persistent homology based Combinatorial Topological Fingerprints (CTFs), a novel molecular representation for characterizing and understanding the molecular function-structure map.

In this chapter, firstly, we will introduce the proposed Combinatorial Topological Fingerprints (CTFs) based on persistent homology. The details of how the CTFs are designed including the Binned Barcode Representation (BBR) will be discussed. Secondly, we will introduce some auxiliary features used in this work, such as numerical molecular descriptors generated by using the RDKit software, most of which are related to the biochemical properties of the queried molecule. These auxiliary features are included in an attempt to further improve the predictive performance of the machine learning models.

5.1 Combinatorial Topological Fingerprints

In the current study, algebraic topology is employed to discriminate mutagens and nonmutagens. The proposed CTFs are computed through the filtration process of the molecular structural data.

Figure 5.1 shows of persistent homology barcodes calculated for all carbon atoms (point cloud) based on Vietoris-Rips complex of Benzo[a]pyrene and Furosemide. Benzo[a]pyrene is a potent mutagen and a public health concern as an environmental pollutant. It is primarily found in gasoline and diesel exhaust, cigarette smoke, etc. The structure of Benzo[a]pyrene consists of five benzene rings formed during the incomplete combustion of organic matter which is easily captured by its Betti-1 barcodes, as we can see from Figure 5.1. Furosemide is a nonmutagen. It is a medication used to treat fluid build-up due to heart failure, liver scarring, or kidney disease. It is also known as a potent loop diuretic and is widely used to treat hypertension and edema. When taking only carbon atoms, the Betti-1 barcodes captures the benzene ring structure in Furosemide very nicely. The persistent homology barcodes of Benzo[a]pyrene and Furosemide have shown quite different patterns as illustrated in Figure 5.1. The persistent homology barcodes encode the intrinsic connectivity characteristic of the corresponding molecule, based on which we could design a list of topological features in order to distinguish between mutagens and nonmutagens.

The filtration barcodes have also shown ability to distinguish between isomers. As is

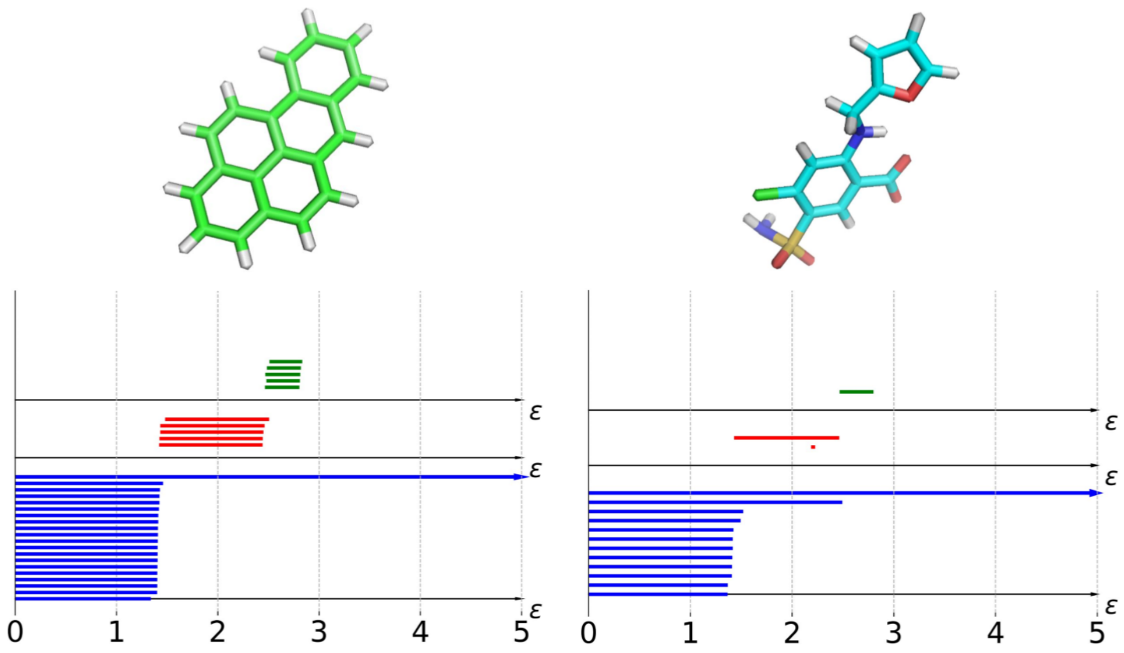


Figure 5.1: Comparison of the persistent homology barcodes for mutagen Benzo[a]pyrene (charts in the left column; CAS: 50-32-8) and nonmutagen Furosemide (charts in the right column; CAS: 54-31-9) using all carbon atoms filtration on associated Rips complexes. From top to bottom, are the 3-dimensional molecular structures, Betti-2, Betti-1, and Betti-0 barcodes of corresponding molecules.

illustrated in Figure 5.2, even though the Betti-0 and Betti-1 persistent homology barcodes of the trans and cis isomers show quite similar patterns, we can still distinguish between them according to their Betti-2 persistent homology barcodes. As it shows clearly in Figure 5.2 that the Betti-2 persistent homology barcodes of the trans isomer contains only one bar, while the Betti-2 persistent homology barcodes of the cis isomer contains two bars.

Furthermore, we have also examined the ability of filtration barcodes for distinguishing optical isomers. An optical isomer, also known as enantiomer, is one of two stereoisomers that are mirror images of each other that are non-superposable (not identical). As is illustrated in Figure 5.3, both the Betti-0 and Betti-1 persistent homology barcodes of two optical isomers have shown different patterns.

To extract knowledge from the persistent homology filtration barcodes, we employ the

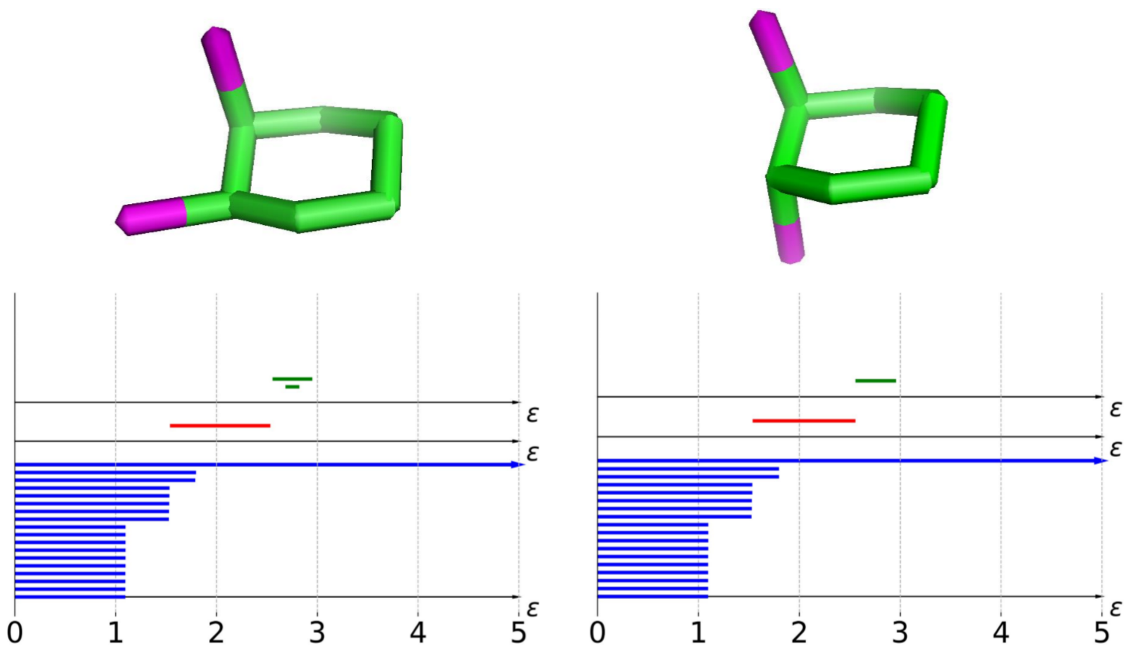


Figure 5.2: Comparison of the persistent homology barcodes for cis-1,2-Dichlorocyclohexane (charts in the left column; CAS: 10498-35-8) and trans-1,2-Dichlorocyclohexane (charts in the right column; CAS: 1121-21-7) using all atoms filtration on associated Vietoris-Rips complexes. From top to bottom, are the 3-dimensional molecular structures, Betti-2, Betti-1, and Betti-0 barcodes of corresponding molecules.

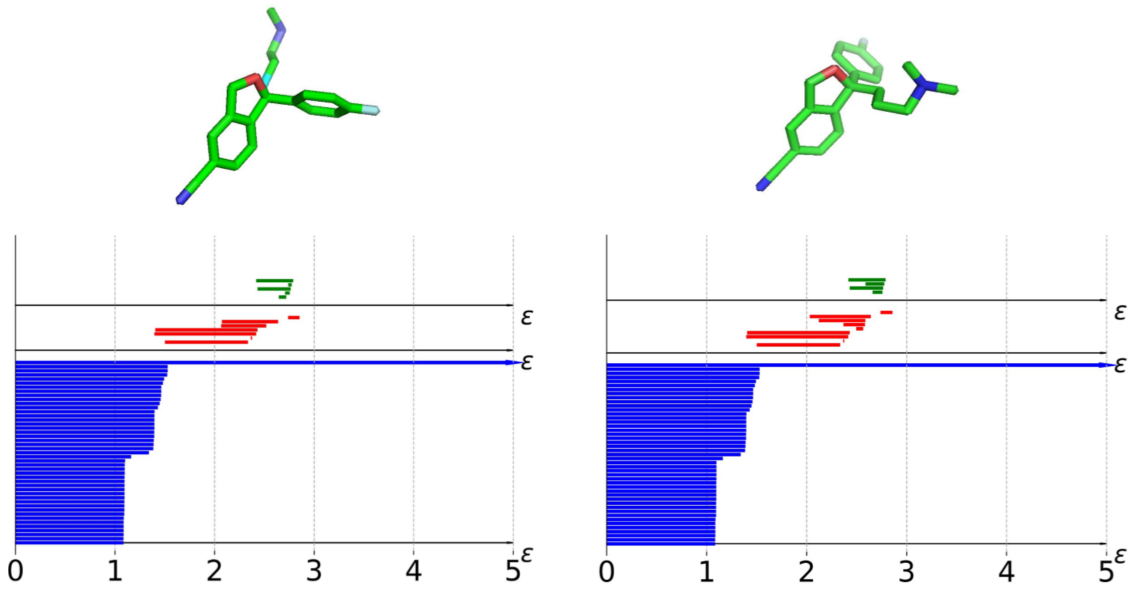


Figure 5.3: Comparison of the persistent homology barcodes for (S)-Citalopram (charts in the right column; CID: 146570) and (R)-citalopram (charts in the left column; CID: 6101829) using all atoms filtration on associated Vietoris-Rips complexes. From top to bottom, are the 3-dimensional molecular structures, Betti-2, Betti-1, and Betti-0 barcodes of corresponding molecules.

binned barcode representation (BBR) [91] by dividing persistent homology barcodes into a number of equally spaced bins on interval $[0, 4.5]$ with step size of 0.5. Total number of 9 bins are generated, namely, [0, 0.5], (0.5, 1.0], ... , (4.0, 4.5] Å. For Betti-0 barcodes, we count the number of bars whose death time is within each bin, and the number of bars that overlap with each bin (as all Betti-0 bars are born at time zero). For Betti-1 and Bett-2 barcodes, we count the number of bars whose birth time is within each bin, the number of bars whose death time is within each bin, and the number of bars that overlap with each bin. This result in 8*9 features. We also include features such as maximum, minimum, average, summation, standard deviation of bar length and so on to capture the macroscopical characteristic of the given molecule, resulting in another 35 features. Thus in total, we have designed 107 topological features for each barcodes. The list of topological features is shown in Table 5.1.

But the representability of the persistent homology barcodes using only carbon atoms of a given molecule is limited, and it is usually not good enough to capture all connectivity characteristic of the given molecule. As we can see from Figure 5.4, when we build the point cloud using both carbon and nitrogen atoms, the barcodes of Betti-1 could easily capture the two aromatic rings of quinoline, while using carbon atoms alone, the pyridine ring is missed.

As suggested by Figure 5.4, we introduce various combinations of elements in order obtain a comprehensive representation of a given molecule. Each combination of elements determines the point cloud extracted from the corresponding molecular structure and will generate a barcode plot of the given molecule. Then for each barcode plot, we will compute the topological features defined in Table 5.1. The concatenation of all topological features generated by all combinations of elements provides a comprehensive representation for each molecule. The total number of topological features is thus 107*(total number of combinations

Group	Description
1	maximum length of Betti-0 bars
2	minimum length of Betti-0 bars
3	average length of Betti-0 bars
4	standard deviation of length of Betti-0 bars
5	total length of Betti-0 bars
6	maximum birth time of Betti-1 bars
7	minimun birth time of Betti-1 bars
8	average birth time of Betti-1 bars
9	standard deviation of birth times of Betti-1 bars
10	sum of birth times of Betti-1 bars
11	maximum death time of Betti-1 bars
12	minimun death time of Betti-1 bars
13	average death time of Betti-1 bars
14	standard deviation of death times of Betti-1 bars
15	sum of death times of Betti-1 bars
16	maximum length of Betti-1 bars
17	minimum length of Betti-1 bars
18	average length of Betti-1 bars
19	standard deviation of length of Betti-1 bars
20	total length of Betti-1 bars
21	maximum birth time of Betti-2 bars
22	minimun birth time of Betti-2 bars
23	average birth time of Betti-2 bars
24	standard deviation of birth times of Betti-2 bars
25	sum of birth times of Betti-2 bars
26	maximum death time of Betti-2 bars
27	minimun death time of Betti-2 bars
28	average death time of Betti-2 bars
29	standard deviation of death times of Betti-2 bars
30	sum of death times of Betti-2 bars
31	maximum length of Betti-2 bars
32	minimum length of Betti-2 bars
33	average length of Betti-2 bars
34	standard deviation of length of Betti-2 bars
35	total length of Betti-2 bars
36	72 BBR features

Table 5.1: List of topological features

of elements).

But when we consider the top ten elements (based on their occurrences in the training set), namely C, N, O, H, S, P, F, Cl, I, Br, the total number of all possible combinations is too

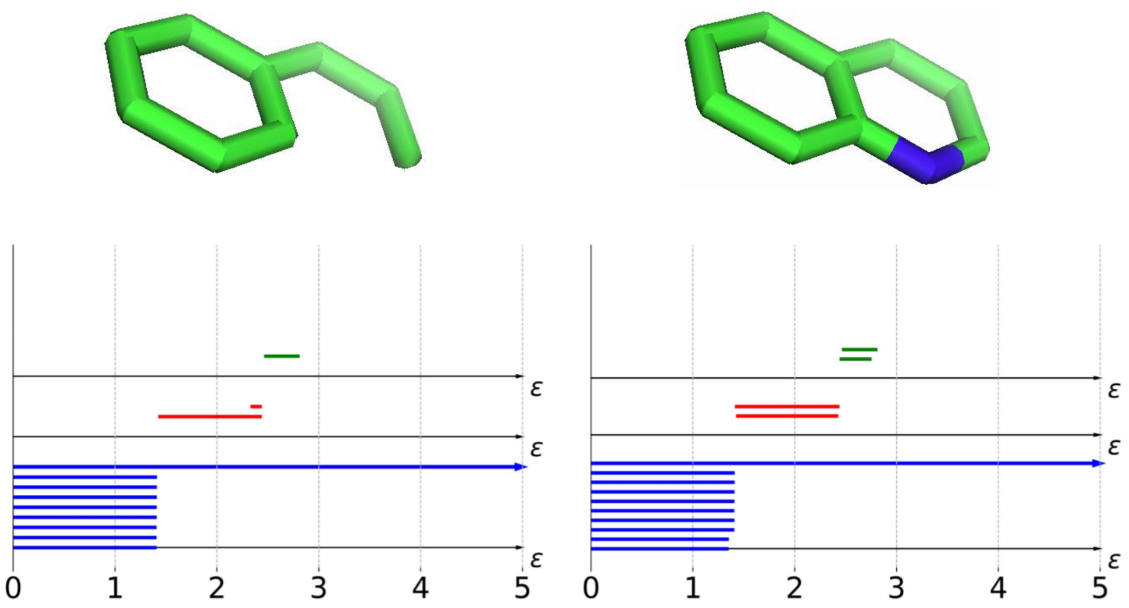


Figure 5.4: Comparison of the barcodes for quinoline using carbon atoms only (charts in the left column) and carbon-nitrogen combination (charts in the right column) filtration on associated Rips complexes. From top to bottom, are the corresponding 3-dimensional molecular structures, Betti-2 (green), Betti-1 (red), and Betti-0 (blue) barcodes.

large ($2^{10} - 1 = 1023$). Not only would these combinations introduce too much redundancy in the feature space, but also make any classic machine learning algorithm hard to implement computationally. Thus we need to properly select a number of combinations such that they would lower the level of redundancy in the feature space but could still capture most of the characteristics of a given molecule. In our work, we determine whether a combination is selected or not based on its occurrence frequency in the training set. For instance, when a molecule in the training set contains at least one element in the combination, the occurence of this combination is counted by one.

Figure 5.5 illustrates the relationship between the occurrence frequency of a combination and its 5-fold cross validation performance on the training set. Here we used all 15 possible combinations out of the 4 most important elements, namely C, N, O, H. Three of the most important performance measurements, namely accuracy, ROC-AUC, and sensitivity have shown very high correlation with the occurrence frequency of the combinations.

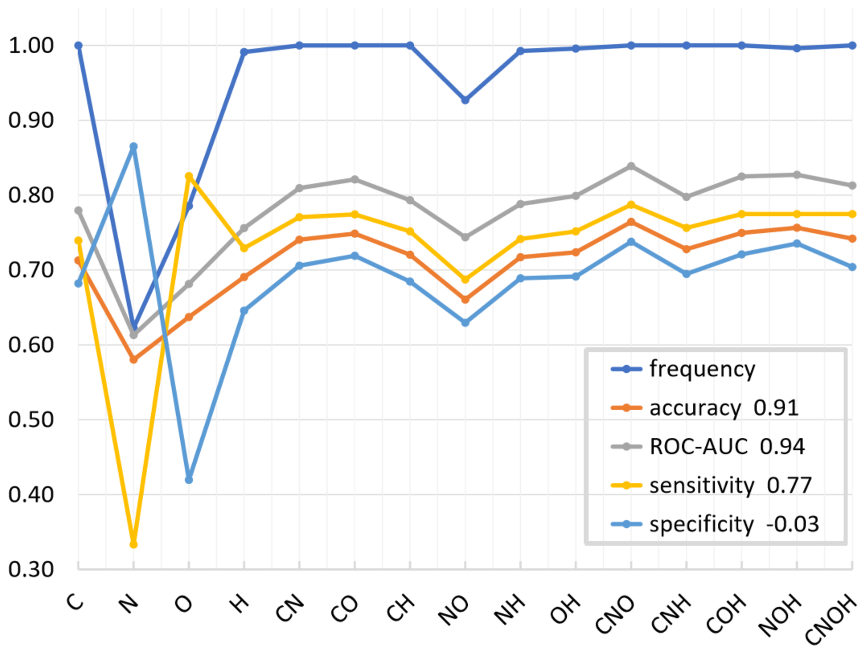


Figure 5.5: Illustration of the relationship between the occurrence frequency of a combination of elements and its 5-fold cross validation performances. The horizontal axis is the corresponding combination. The numbers in the legend are the Pearson correlation coefficients between the specific cross validation scores and the occurrence frequencies.

Finally, we have chosen 28 combinations of elements based on their occurrences in the training set. Thus the total number of topological features generated for each molecule is 28*107 = 2996. These features are then used as inputs of machine learning algorithms for classification. The 28 combinations are listed in Table 5.2.

5.2 Auxiliary features

While the proposed CTFs provide a comprehensive molecular representation by giving a deep examination of the topological evolution through the filtration process, there might be some other crucial features in mutagenicity prediction that topology can not grasp. Thus besides the topological features introduced in the last section, we have also tried to include

1	{C}	8	{Cl}	15	{C,Cl}	22	{H,S}
2	{N}	9	{I}	16	{N,O}	23	{H,Cl}
3	{O}	10	{Br}	17	{N,H}	24	{C,N,O}
4	{H}	11	{C,N}	18	{N,S}	25	{C,N,H}
5	{S}	12	{C,O}	19	{O,H}	26	{C,O,H}
6	{P}	13	{C,H}	20	{O,S}	27	{N,O,H}
7	{F}	14	{C,S}	21	{O,Cl}	28	{C,N,O,H}

Table 5.2: Combinations of elements used for generating topological fingerprints

some auxiliary features related to the geometric, biophysical, and electrostatic properties of the queried molecules, in a hope to further improve the predictive performance of our model.

The first group of features we have tried to include are microscopic features such as atomic surface areas, atomic electrostatic charges, atomic solvation free energies, and so on. The generation of these features is via the **MIBPB** package [83] by the following two steps:

1. Prepare 3D molecular structures, including the 3D molecular atomic coordinates, radii and partial charges. The MIBPB package supports molecular structures in 3D PQR format, which can be generated by the Amber molecular simulation package [81]. Refer to Section 2 for more details.

2. Poisson Boltzmann calculation, the **MS_Intersection** module is used to generate the analytical solvent excluded surface (it also prepares the atomic surface areas for each molecule), after which the **mibpb5** module is used to solve the Poisson Boltzmann equation (atomic electrostatic charges and atomic solvation energies are prepared).

Once we have obtained the atomic surface areas, atomic electrostatics charges, and atomic solvation energies for each biomolecule, we have designed a list of element specific features to be included in the feature vector. The features are listed in Table S1 in the supplementary materials.

The second group of features we have tried to include are the molecular descriptors provided in the "Descriptors" module of the RDKit software [34]. We only extract descriptors that are numerical, resulting in another 192 auxiliary features. The details of these features can be found in Table S2 in the supplementary materials.

Chapter 6

Machine learning

Machine learning can be summarized as learning a function F that maps the input variables X to output variables Y,

$$Y = f(X)$$

A machine learning algorithm enables the computer to learn this target function F without explicitly programmed, which means that the structure of a machine learning algorithm is not specified a *priori* but is instead determined by data. Different machine learning algorithms differ by making different assumptions about the structure of the target function and how it can be learned. Machine learning has shown considerable success in a wide range of practical applications where physical or empirical models with good performance are unavailable.

Machine learning algorithms are often classified into two classes: parametric and non-parametric ones. A parametric machine learning algorithm, such as linear regression, logistic regression, linear discriminant analysis, perceptron, and Naive Bayes, simplifies the target function to a predefined form with a fixed size of parameters and adjusts the parameters based on the observed data. While a non-parametric machine learning algorithm builds the model directly from the data, algorithms such as nonlinear kernel support vector machines, k-nearest neighbors, neural networks, ensemble methods, and deep learning. A parametric algorithm is computationally fast and easier to implement, but its assumption about the data is usually too strong to make a good fit. Also with limited algorithm complexity, parametric machine learning algorithms usually only work for simpler problems. On the contrary, a non-parametric machine learning algorithm makes fewer assumption about the data, is capable of fitting a large number of functional forms and usually generates more powerful models.

The so-called "No Free Lunch theorem" (Wolpert 1996) have shown that learning algorithms cannot be universally good, especially in supervised learning. The assumption of a good algorithm for one problem may not hold for another, thus it is common practice in machine learning to try multiple algorithms and find out one that works best for a particular problem. In this study, we have examined a number of non-parametric machine learning algorithms including decision trees, nonlinear kernel support vector machines, ensemble methods like random forest, extra trees, and gradient boosting trees, and multi-task deep neural networks. The results are reported in Chapter 7.

6.1 Simple learning algorithms

6.1.1 Decision trees

Classification and Regression Trees (CART) introduced by Leo Breiman [92] to refer to Decision Tree algorithms that can be used for classification or regression predictive modeling problems. A decision tree classifier is a tree-structured model (represented by a binary tree) for classification, and for facilitating decision-making in sequential decision problems, it is probably the simplest predictive modeling machine learning algorithm to implement. Classification trees are derived using recursive partitioning data algorithms that classify each incoming sample into one of the class labels for the outcome. Each internal node of the tree represents a ”test” on a feature, each branch of the tree represents an outcome of the test, and each terminal (leaf) node represents a decision made (class label for each incoming sample).

A classification tree consists of three types of nodes described as follows:

1. Root node: The top node of the tree comprising all samples.

2. Splitting node: A node that assigns an incoming sample to a subgroup.

3. Terminal (leaf) node: Decision made (class label) for each incoming sample.

A decision tree is simple to implement, understand, and interpret, it is able to handle both numerical and categorical types of data, moreover, it is frequently combined with other decision making techniques. However, decision trees can be non-robust, in that a tiny little perturbation in the training data can result in a big change of the tree, as wells as a big change in the final prediction. Also, sometimes it can create over-complex trees that overfit the training data (do not generalize well to new data). In most cases, the predictive performances of decision trees do not tend to be as accurate as other approaches.

6.1.2 Support Vector Machines

Support Vector Machine (SVM) [93] is a discriminative supervised learning method that can be employed for both classification and regression purposes, which is formally defined by a hyperplane that best divides a dataset into two classes. It becomes popular because of its success in handwritten digit recognition. SVM is now regarded one of the most important“kernel methods”, one of the key area in machine learning. Given the training samples, the SVM algorithm outputs an optimal hyperplane which categorizes new samples.

Consider a two-class, linearly separable classification problem, there could be many decision boundaries possible (see Figure 6.1), the Perceptron algorithm and many other algorithms can be used to find such boundaries. Intuitively, if the decision boundary passes too close to the data points, it will be non-robust (noise sensitive) and it will not generalize correctly. Thus the decision boundary should be as far away from the data of both classes as possible.

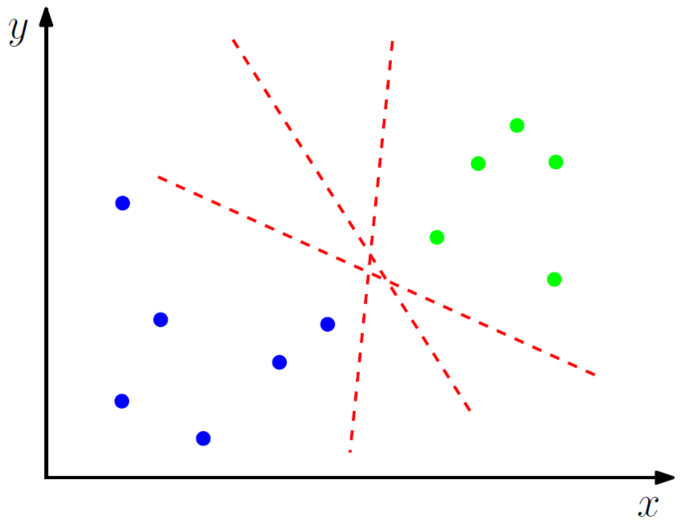


Figure 6.1: For this two-class and linearly-separable classification problem (two classes are represented by blue and green dots), the dashed lines are all possible decision boundaries. The optimal decision boundary can be selected by the SVM algorithm.

The optimal hyperplane is defined to be the one that gives the largest minimum distance (margin) to the training samples (see Figure 6.2). Since the distance between the origin and straight line $\mathbf{w}^T\mathbf{x} = k$ is $k/||\mathbf{w}||$, we have:

$$m = \frac{2}{||\mathbf{w}||} \tag{6.1}$$

Given data points (training samples) $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and let $y_i \in \{1, -1\}$ be the class label of $\mathbf{x}_i$. Ideally, the decision boundary should classify all data point correctly,

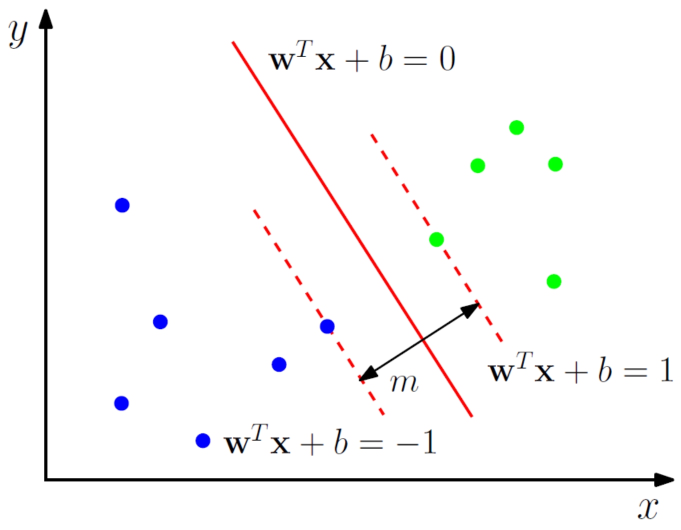

y
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+b=1$
m
$\mathbf{w}^T\mathbf{x}+b=-1$
x

Figure 6.2: For this two-class and linearly-separable classification problem (two classes are represented by blue and green dots), the optimal decision boundary is represented by the red solid line. m is the maximal margin. Two data points on the dashed lines are called supported vectors.

$$y_i(\mathbf{w}^T\mathbf{x}_i+b) \geq 1, \forall i \tag{6.2}$$

The decision boundary can be found by solving the constrained optimization problem described as follows,

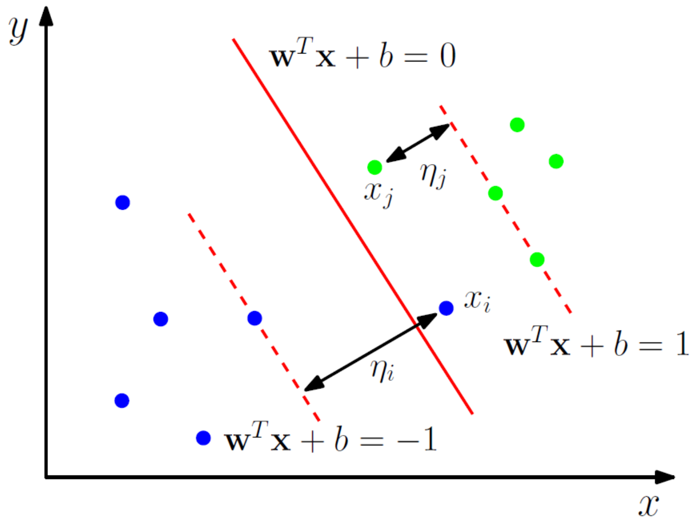
$$\begin{gathered}\text{Minimize } \tfrac{1}{2}||\mathbf{w}||^2 \\ \text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i+b) \geq 1, \forall i\end{gathered} \tag{6.3}$$

Solving this constrained optimization problem can be transformed into its dual problem, which is a quadratic programming problem that can be solved by many approaches such as the Sequential Minimal Optimization (SMO). Setting the gradient of the Lagrangian with respect to $\mathbf{w}$ and b to zero, we can get the dual problem described as follows:

$$\text{Maximize } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j^T$$

$$\text{subject to } \alpha_i \geq 0, \text{ where } \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{6.4}$$

If we allow error η_i in classification (see Figure 6.3, η_is are called slack variables in optimization and approximate the number of misclassified samples).



y
$\mathbf{w}^T\mathbf{x} + b = 0$
η_j
x_j
x_i
$\mathbf{w}^T\mathbf{x} + b = 1$
η_i
$\mathbf{w}^T\mathbf{x} + b = -1$
x

Figure 6.3: Illustration of the soft margin hyperplane. x_i and x_j are two misclassified samples, the corresponding errors are represented by η_i and η_j.

If we minimize $\sum_i \eta_i$, η_i can be obtained by:

$$\begin{cases} \mathbf{w}^T x_i \geq 1 - \eta_i, & \text{for } y_i = 1 \\ \mathbf{w}^T x_i \leq -1 + \eta_i, & \text{for } y_i = -1 \\ \eta_i \geq 0, & \text{for } \forall i \end{cases} \tag{6.5}$$

We want to minimize:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \eta_i \tag{6.6}$$

where C is the trade-off parameter between error and margin.

The optimization problem (dual problem) becomes:

$$\text{Maximize } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{n} \alpha_i\alpha_j y_i y_j x_i^T x_j^T$$

$$\text{subject to } C \geq \alpha_i \geq 0, \text{ where } \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{6.7}$$

which is similar to the optimization problem in the linearly-separable case, except there is an upper bound C for each α_i. It is also a Quadratic Programming (QP) problem and α_is can be solved by any QP solver. Once we get all α_is, $\mathbf{w}$ can be recovered very easily.

To extent large-margin classifier with a linear decision boundary to nonlinear, we need to transform sample point x_i to a higher dimension space (feature space), i.e. the space of $\phi(x_i)$, since linear operation in the feature space in equivalent to non-linear operation in the original input space. With a proper transformation, classification can become much easier. However, the feature space is typically infinite-dimensional, which makes the computation in the feature space too expensive. The kernel trick is introduced to resolve the problem. Recall Equation 6.7, all data points only appear as inner product, thus we do not need the mapping explicitly as long as we can calculate the inner product in the feature space. The kernel function K is defined as

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{6.8}$$

which can be used to avoid carrying out ϕ explicitly. Examples of kernel functions include degree d polynomial kernel $K(x, y) = (x^T y + 1)^d$, Radial Basis Function (RBF) kernel $K(x, y) = exp(-||x - y||^2)/(2\sigma^2)$, and Sigmoid kernel $tanh(\kappa x^T y + \theta)$.

We change all inner products in Equation 6.7 with kernel functions, the optimization problem becomes,

$$\text{Maximize } W(\boldsymbol{\alpha}) = \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x, y)$$
$$\text{subject to } C \geq \alpha_i \geq 0, \text{ where } \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{6.9}$$

6.1.3 Neural Networks

An Artificial Neural Network (ANN) is another popular non-parametric machine learning algorithm that can be used for regression and classification problems. It is inspired by the way biological neural networks in the human brain process information. Artificial Neural Networks are widely used in machine learning research and industry, such as speech recognition, computer vision and text processing.

An ANN is based on a collection of connected units called artificial neurons (building blocks), connections between neurons can transmit signals from one to another. A basic neuron is consists of a "black box" with weighted inputs and output, where inside the "black box" of the neuron, there exists an activation function (see Figure 6.4).

Neurons are typically organized in layers, where different layer may perform different transformations on their input. Layers are made of a number of interconnected neurons which may contain different activation functions. Patterns are presented to the network by the "input layer", which connects to one or more "hidden layers" where the actual processing is done via a system of weighted connections. Most neural networks are fully connected,

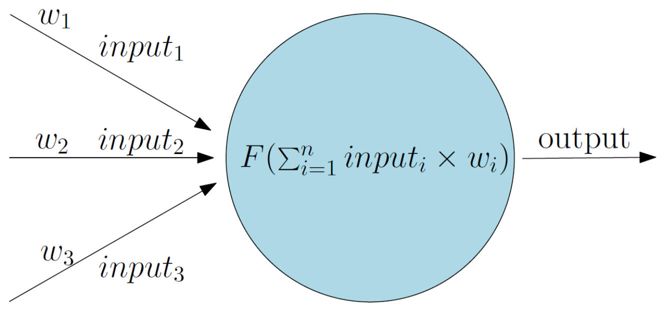


Figure 6.4: Illustration of a neuron. There are three inputs and one output, the weights for all inputs are given as w_1, w_2, and w_3. F is the activation function for this neuron.

which means each hidden and output neuron is connected to every neuron in the layers either side. Input neurons provide information from the outside world to the network, which pass on the information to hidden neurons. Hidden neurons have no direct connection with the outside world, they compute and transfer information from the input neurons to the output neurons. Within each hidden neuron is some type of activation function (step function, sigmoidal function for instance) which polarizes network activity and helps it to stabilize. The output neurons are responsible for computation and transferring information from the network to the outside world. Since the Softmax function maps any real-valued vector to a vector of values between 0 and 1 that add up to o1, generally, it is used as the activation function in the output layer to ensure that the outputs are probabilities.

Most ANNs contain some form of ”learning rule” such as the delta rule, which modifies the weights of the connections according to the input patterns that it is presented with. The delta rule is often utilized by the most common class of ANNs called Back Propagational Neural Networks (BPNNs). Initially all the edge weights are randomly assigned. For every input in the training set, the ANN is activated and its output is observed. This output is compared with the actual output that we already know, and the error is ”propagated” back to the previous layer, and the weights are adjusted accordingly. This process is repeated until

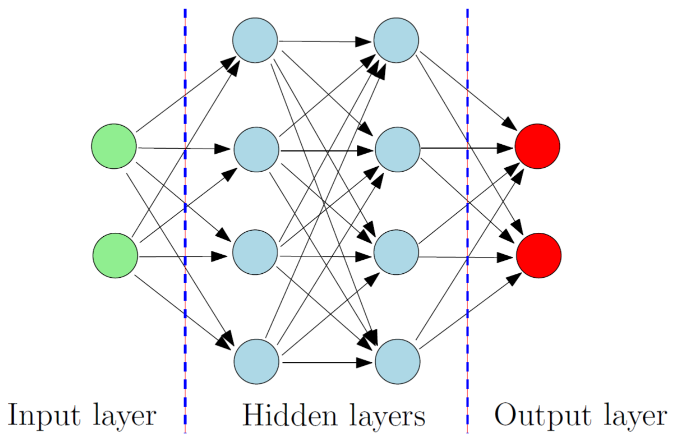


Figure 6.5: Illustration of a artificial neural network. The neurons are represented by circles, connections between neurons are indicated by arrows. All these connections have weights associated with them.

the output error is below a predetermined threshold. Intuitively, when a neural network is initially presented with a pattern it makes a random guess as to what it might be, then it checks how different its answer was from the actual one and makes adjustments to its connection weights.

6.2 Ensembles

While most common machine learning algorithms, such as decision trees, support vector machines, and neural networks, focus on building a single ”strong” model, there are some other machine learning algorithms (called ensemble methods) that focus on building a bucket, or an ensemble of ”weak” models (called weak learners) for some particular predictive tasks. One can also think of building an ensemble of ”strong” models like neural networks, but in reality, the number of models required in the ensemble is usually quite huge. Thus it is practical to build an ensemble that consists of a large number of simple weak learners and combine them altogether to make a strong learner. Even if the performance of a single weak learner may be only slightly better than random chance, the combination of a large number of weak learners would generate improved results. In fact, Ensemble algorithms are so powerful that almost half of data mining competitions' winners used some variants of tree ensemble methods. Intuitively speaking, ensemble algorithms are divide-and-conquer approaches used to improve predictive performance. Two frequently used ensemble classification algorithms are random forest [94, 95] and gradient boosting trees [96, 97].

Random Forest algorithm is based on the idea of Bagging (stands for Bootstrap Aggregation), which is parallel ensemble approach with all weak learners (uncorrelated decision trees) built independently. It combines Brieman's bagging idea and random selection of features (feature bagging), aiming at lowering the variance by introducing bias to the model (variance bias tradeoff). The following describes the procedure of how the random forest model is trained:

For some number of trees N, assume the total number of features is M,

1. Sample N random training samples with replacement.
2. For each training sample, train a decision tree f_k, where $1 \leq k \leq N$
    - At each node of the decision tree, a subset of features is selected at random.
    - The feature that provides the best split is used for the binary split at each node.
    - At the next node, another subset of features is selected and the process is repeated.

Here at each node, Brieman suggests that the number of features randomly selected can be $\frac{1}{2}\sqrt{M}$, $\sqrt{M}$, or $2\sqrt{M}$ [94]. The resultant model $\hat{F}$ is the linear combination (majority vote for classification problems) of all weak learners:

$$\hat{F} = \frac{1}{N}\sum_{k=1}^{N} f_k$$

The **Extra Trees** (or Extremely randomized trees) algorithm [98] is a variation of the Random Forest algorithm by adding one further randomization step. Unlike the Random Forest algorithm, each iteration uses the whole training set instead of a bootstrap replica, moreover, splits are selected completely at random instead of using some criterions, that is to say, for each feature under consideration, a random cut-point is selected for the split. The use of the whole training set (instead of a bootstrap replica) at each iteration helps to minimize bias, while choosing a fully random cut-point at each split helps to reduce the variance more strongly than other weaker randomization methods.

Gradient Boosting Trees is another flexible non-parametric machine learning algorithm for classification and regression which is based on the idea of boosting. It is a sequential ensemble approach in an attempt to generate a strong learner built on a sequence of weak learners (decision trees, especially CART trees of fixed size). The weights and training given to each of these weak learners ensures the high accuracy of the overall model. In gradient boosting, the model assumes an additive expansion:

$$F(X, \beta, \alpha) = \sum_{i=1}^{N} \beta_i f(X, \alpha_i)$$

where N is the number of gradient boosting iterations, $f(X, \alpha_i)$ is a weak learner (decision trees), β_i is the weight of the corresponding tree, and α_i is the weight of the training examples and parameterizes the splits.

The procedure to build a gradient boosting machine is described as follows:

1. Initialize a list of weak learners.

2. For the ith gradient boosting iteration:

- Reweight examples (update α_i) by "up-weighting" examples that current model poorly predicts.
- Build a new weak learner, i.e. $f(X, \alpha_i)$ based on the weighted examples and compute its corresponding weight β_i.
- Add the new weak learner to the sequence of weak learners built before

Thus the resultant model is the weighted combination of the sequentially built weaker learners. When we allow the tree to be greedily built from a random subsample of the training examples during each iteration, we get the stochastic gradient boosting machine. The subsampling in each iteration helps to reduce correlation between the trees.

In this study, we have employed the Random Forest Classifier, Gradient Boosting Tree Classifier and Extra Trees Classifier in the python sklearn module. Uniform sets of tuning parameters are chosen for all classifiers, which are listed in the following tables:

n_estimators	max_depth	max_features
3000	4	sqrt

Table 6.1: Parameters used for the Random Forest Classifier and the Extra Trees Classifier, the remaining parameters are set as default.

Here in Table 6.1, for the Random Forest Classifier and the Extra Trees Classifier, n_estimators denotes the number of decision trees to be combined in the final model; max_depth denotes the maximum depth of the individual decision tree and it limits the number of nodes in each tree; learning_rate (shrinkage rate) shrinks the contribution of each tree by learning_rate and slows the learning; max_features denotes the number of features to consider when looking for the best split, choosing max_features $<$ number of features will also lead to higher accuracy by introducing bias to the model.

n_estimators	max_depth	min_samples_split	learning_rate	subsample	max_features
3000	8	4	0.005	0.8	sqrt

Table 6.2: Parameters used for the Gradient Boosting Trees Classifier, the remaining parameters are set as default.

In Table 6.2, for the Gradient Boosting Trees Classifier, n_estimators denotes the number of boosting stages to perform, which can be taken a large number since gradient boosting is fairly robust against over-fitting; min_sample_split denotes the minimum number of samples required to split an internal node; subsample is the fraction of samples to be used for fitting the individual decision trees. Choosing it < 1.0 results in Stochastic Gradient Boosting; max_depth, learning_rate and max_features denote the same meaning as in the Random Forest Classifier.

6.3 Deep learning

Deep learning (also known as hierarchical learning or deep structured learning) is a collection of statistical learning techniques used to learn feature hierarchies, which is usually based on ANNs that contain more than one hidden layer. Deep learning architectures such as deep neural networks (DNN), recurrent neural networks (RNN), and deep belief networks (DBN) [99] have gained great success and produced models that surpass classical machine learning algorithms and in some cases superior to human experts in many scientific application fields including computer vision, speech recognition, natural language processing (NLP), recommendation systems, and bioinformatics.

As discussed before, a neural network is represented by layers of learning, it soared in popularity in the 1980s, peaked in 1990s, and slowly declined since then. At that time, people expected that adding more layers in the neural network's architecture could allow it to learn better models, however, in reality, it did not work out. The problem is because of the intrinsic nature of instability associated with learning by gradient descent in deep neural networks (vanishing or exploding gradient problem). The instability of gradient descent makes the learning in the early or later layers stuck during training. Intuitively, different layers in the deep network are learning at quite different speed, when the later layers are learning well, early layers might get stuck and learn nothing at all. Moreover, the choice of activation functions, the way the weights are initialized will also affect the ability to train a deep network.

The workforce learning algorithm for deep learning is the Stochastic Gradient Descent (SGD) by backpropagation, which is also the most used optimization algorithms for many other machine learning algorithms. The use of SGD in neural networks is motivated by the learning instability of gradient descent and expensive cost of backpropagation over the entire training set.

By taking an average gradient on a minibatch of m training samples, it is possible to obtain an unbiased estimate of the gradient. A description of how SDG updates the weights of the network at training iteration k is given below [100].

Given initial parameter $\boldsymbol{\theta}$ and learning rate ϵ_k at the k-th iteration,

- Sample a minibatch of m examples from the training set $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.
- Compute estimated gradient: $\hat{\mathbf{g}} = \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$
- Update weights: $\boldsymbol{\theta} = \boldsymbol{\theta} - \epsilon_k \hat{\mathbf{g}}$
- Repeat the process until the stopping criterion is met.

where L is the per-example loss function, $f(\mathbf{x}, \boldsymbol{\theta})$ is the predicted output for input $\mathbf{x}$.

Here the learning rate ϵ_k is a crucial parameter. In practice, unlike batch gradient descent (optimization using the entire training set) choosing a uniform learning rate, the learning rate of SDG is decreasing over time, since the random sampling at each iteration introduces noise that does not vanish even when the total cost function has reached its minimum. SGD and the related minibatch algorithm allow convergence of the network weights even when the number of samples in the training set is very large. When the training set is large enough, SGD may even converge within a predetermined tolerance before it has processed the entire training set.

Even though in practice the SGD works quite well (much faster than batch gradient descent), one of its biggest problems is still the learning speed. The method of momentum is designed to speed it up. Instead of moving with a velocity that is proportional to the gradient of the loss function, it is modified to be proportional to a smoothed average of previous gradients. Intuitively, for the method of momentum, each "move" on the loss function surface relies a little bit on previous moves. By taking the gradient history into account (adding velocity component to the parameter update routine), SGD is less likely to end up at a bad local minimum or some other stationary point (local maximum or saddle point), moreover, it accelerates the speed of learning, especially in the face of high curvature, small but consistent gradients, or noisy gradients. SGD with momentum is described as follows:

Given initial parameter $\boldsymbol{\theta}$ and initial velocity $\mathbf{v}$,

- Sample a minibatch of m examples from the training set $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.

- Compute estimated gradient: $\hat{\mathbf{g}} = \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$
- Update velocity: $\mathbf{v} = \alpha\mathbf{v} - \epsilon\mathbf{g}$.
- Update weights: $\boldsymbol{\theta} = \boldsymbol{\theta} - \epsilon_k \hat{\mathbf{g}}$
- Repeat the process until the stopping criterion is met.

Here α is a hyperparameter in $[0, 1)$, which determines how much previous gradients affects the current direction. A larger α with respect to ϵ means more contribution of previous gradients are made in the current "move".

6.4 Multi-task deep learning

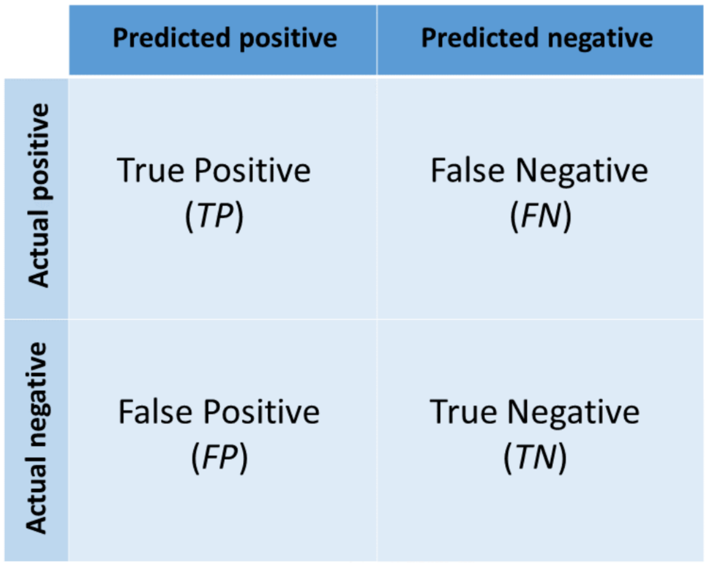
Multi-task learning (MTL) [101, 102] is a way to improve the generalization of DNN and can be cast in different ways in the framework of DNN. Based on the assumption that there exists some statistical relationship between different tasks, MTL treats different tasks in parallel and uses a shared representation of data. In the framework of DNN, the tasks of interest usually share early layers and related weights of the deep network, while later layers in the deep network are specialized to each task. Compared with Single Task Learning (STL) models that depend on having a sufficient training samples to fit good models, a MTL QSAR model integrates available data on related activities (different toxicities in our case) as an alternative to costly *in vivo* bioassays.

In this work, the multi-task deep neural network (MT-DNN) is implemented by the python module Keras [103]. The SGD with momentum optimizer is applied. Categorical crossentropy function is used as the loss function for compiling the model. Two learning tasks are trained in an alternative manner. The details of the architecture of the MT-DNN as well as the predictive result on the developmental toxicity dataset is provided in Chapter 7.

6.5 Metrics

In this section, firstly, we introduce the definition of "Accuracy", "Sensitivity", and "Specificity", which are metrics to measure the predictive performance of binary classifiers.

Figure 6.6 gives the definition of the **confusion matrix** (also called error matrix), which allows the visualization of the performance of a predictive model.



	Predicted positive	**Predicted negative**
Actual positive	True Positive (*TP*)	False Negative (*FN*)
Actual negative	False Positive (*FP*)	True Negative (*TN*)

Figure 6.6: Definition of the confusion matrix. Here "positive" and "negative" represent the queried chemical's mutagenic (or toxic) activity, where "positive" indicates that the queried molecule is mutagenic (or toxic), and "negative" indicates that the queried molecule is non-mutagenic (or nontoxic)

The **Accuracy** is defined as:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{P+N}} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

The **Sensitivity** or True Positive Rate (**TPR**) or **Recall** is defined as:

$$\text{Sensitivity} = \text{TPR} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{TP+FN}}$$

The **Specificity** or True Negative Rate (**TNR**) is defined as:

$$\text{Specificity} = \text{TNR} = \frac{\text{TN}}{\text{N}} = \frac{\text{TN}}{\text{TN+FP}}$$

Secondly, we introduce the Area Under the Curve of the Receiver Operating Characteristic curve (ROC-AUC), which is another very important metric to measure the predictive performance of a binary classifier. The ROC curve is a graphic plot created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. Using normalized units, the area under the ROC curve is equal to the probability that a classifier will rank a randomly picked positive sample higher than a randomly picked negative one (assuming ”positive” ranks higher than ”negative”). For simplicity, hereafter, we use ”ROC-AUC” to denote the area under the ROC curve.

6.6 Workflow

Figure 6.7 illustrates the workflow of this work. We start with the optimized 3D molecular structures (molecular files in PQR format); by selecting different combinations of element types, we get different point clouds (3D atomic coordinates); each point cloud determines its own filtration barcodes (Betti-1, Betti-1, and Betti-2 barcodes), based on which the proposed

CTFs topological features are generated (topological features from different combinations of element types are concatenated to fully represent a given molecule); together with the pre-determined auxiliary features (FFT atomic features and features from RDKit software) and machine learning algorithms (ensemble methods such as Random Forest, Gradient Boosting Trees, and Extra Trees Classifier), we can train the model with the training data, and then make mutagenicity predictions on molecules with unknown toxicity properties.

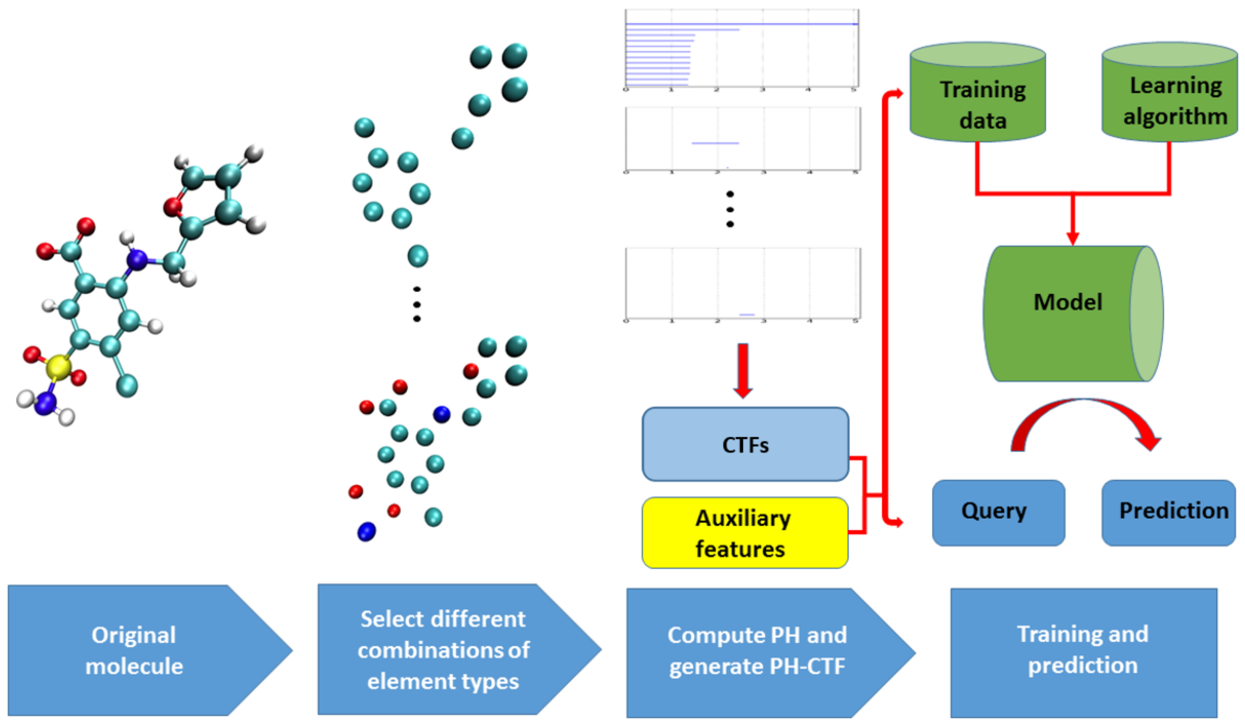


Figure 6.7: Workflow for topological based biomolecular mutagenicity prediction.

Chapter 7

Results

7.1 Mutagenicity prediction

To study the accumulative predictive power of the proposed combinatorial topological fingerprints (CTFs) and to prevent overfitting, we have examined the predictive performances of the proposed CTFs on two external testing datasets AID1189 and AID1194. Each time we consider one more combination of elements and add its corresponding 107 topological features into the feature space. The predictive performances on the two independent external testing sets during this learning process are plotted in Figure 7.1 and Figure 7.2.

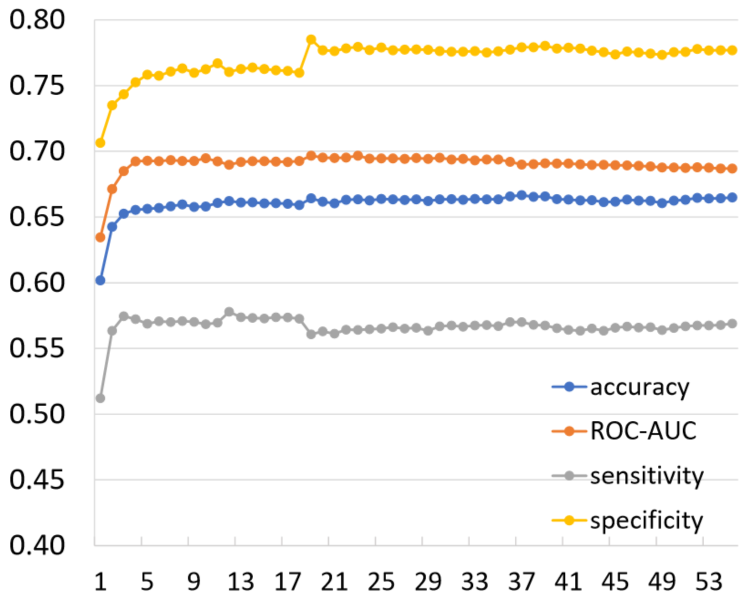


Figure 7.1: Learning curves on external testing dataset AID1189 using Gradient Boosting Trees. The horizontal axis denotes the number of combinations of elements being used.

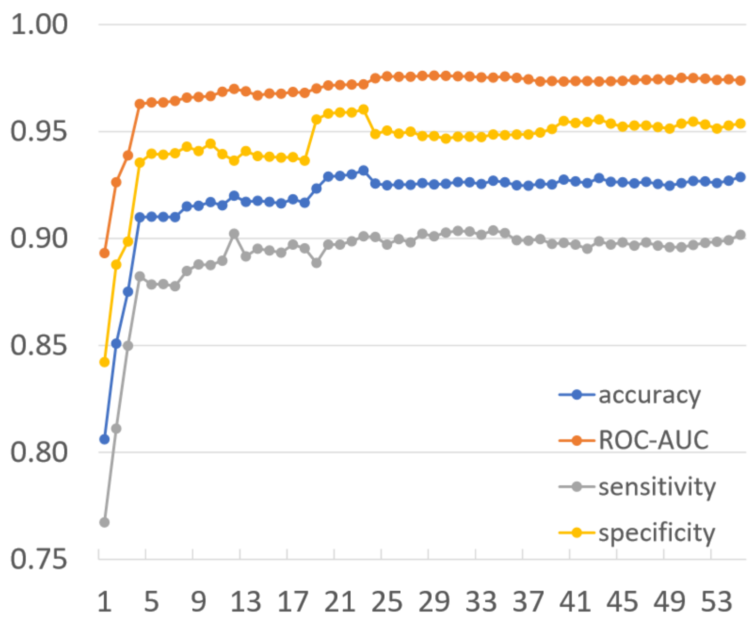


Figure 7.2: Learning curves on external testing dataset AID1194 using Gradient Boosting Trees. The horizontal axis denotes the number of combinations of elements being used.

It shows both in Figure 7.1 and Figure 7.2 that, adding more topological features into the feature space by involving more combinations of elements improves the predictive performances in general. At an early stage, the predictive performances improve relatively faster, but then after some point, they tend to stay at certain levels even when we keep adding more topological features. All learning curves showed in Figure 7.1 and Figure 7.2 suggest that our model is very robust against overfitting. Eventually, we select 28 combinations listed in Table 5.2 to build the final model.

We employ cross validation as an internal evaluation of our model. To cut down randomnesses, we repeat the 5-fold cross validation. The result reported in Table 7.1 is the average of 10 independent 5-fold cross validation results. Note that in Abhil Seal et al's paper [104], the training set contains 8208 molecules after combining Bursi's dataset and Hansen's Benchmark dataset and removing duplicates based on the molecular canonical smiles, which is over a thousand molecules more than what we have in our training set. Then they divided the resultant dataset into 80% for training and the remaining 20% for validation. Unfortunately, the details of how they splited the dataset into training and validation, as well as the CAS numbers in each set were not provided in their paper, so we think it is inappropriate to compare our cross validation result with their validation result on the remaining 20%.

	Accuracy	ROC-AUC	Sensitivity	Specificity
Gradient Boosting Trees	**0.800**	**0.874**	**0.812**	**0.787**
Random Forest	0.759	0.829	0.762	0.756
Extra Trees	0.721	0.795	0.748	0.689

Table 7.1: Repeated 5-fold cross validation result. The training set is the combination of the Bursi's dataset and the Hansen's Benchmark dataset with duplicates removed, which contains 7032 molecules.

The prediction results using the 28 combinations on external testing set AID1189 and AID1194 are provided in Table 7.2. Correponding Receiver Operating Characteristics (ROC) curves are provided in 7.3 and 7.4, respectively. Comparisons with Abhil et al's result [104] are made in Figure 7.5 and Figure 7.6. The results have proven the representative ability of the proposed CTFs by showing that our model by using topological features alone have surpassed the Abhil et al's in general.

	Testing set	Accuracy	ROC-AUC	Sensitivity	Specificity
Gradient Boosting Trees	AID1189	**0.667**	**0.691**	**0.547**	0.776
	AID1194	**0.931**	**0.974**	**0.906**	**0.955**
Random Forest	AID1189	0.647	**0.691**	0.525	**0.789**
	AID1194	0.844	0.921	0.762	0.920
Extra Trees	AID1189	0.627	0.665	0.545	0.724
	AID1194	0.778	0.867	0.719	0.833

Table 7.2: Mutagenicity prediction results on two independent external testing sets by using CTFs only.

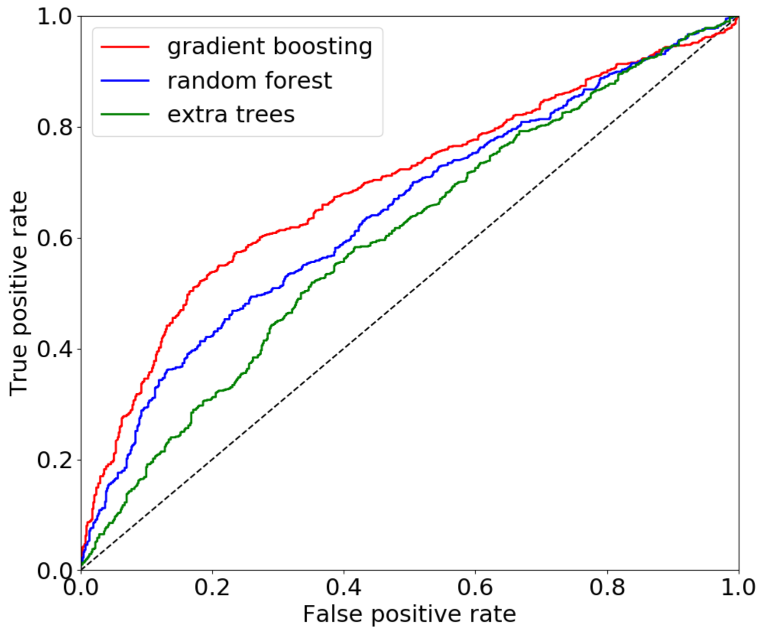


Figure 7.3: ROC curves of the three ensembles of trees methods on external testing dataset AID1189. ROC curves using the proposed CFPs combined with gradient boosting trees, random forest, and extra trees algorithms are indicated in red, blue, and green color, respectively.

7.2 Developmental toxicity prediction

We have further tested the proposed CTFs on the EPA developmental toxicity dataset that contains 285 chemicals. To further prove the representative ability of our proposed CTFs, we also compare it with some existing fingerprints such as the MACCS fingerprints (or the MACCS keys), the Morgan fingerprints, the Estate fingerprints, and the FP4 fingerprints. Moreover, besides the ensembles of trees algorithms, we have also tested the multi-task deep neural networks (MT-DNN) methods in an attempt to further improve the predictive performances. Figure 7.7 demonstrates the architecture of the MT-DNN used in our study. The predicting results of the MT-DNN and ensembles of trees algorithms are reported in Table 7.3. Since the ROC-AUC score is not reported in T.E.S.T. software, it is also not

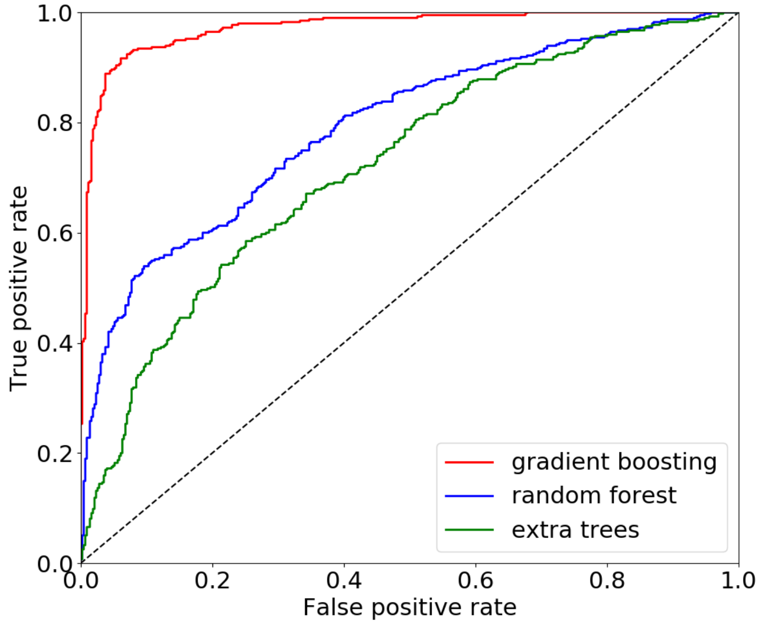


Figure 7.4: ROC curves of the three ensembles of trees methods on external testing dataset AID1189. ROC curves using the proposed CFPs combined with gradient boosting trees, random forest, and extra trees algorithms are indicated in red, blue, and green color, respectively.

reported here in this table.

The training process of the deep networks is designed in an alternative fashion, two branches (tasks) of the networks are trained alternatively, with batch size of 16. Due to the small size of the developmental toxicity dataset and the common features shared by two types of toxicities, we hope that incorporation of the relatively large mutagenicity dataset can help to improve the prediction performance of the developmental toxicity output by influencing the shared layers of the deep networks.

The MACCS fingerprints is the most widely used and known molecular fingerprints, which is a size 166 or 320 binary vector with each bit representing the presence of a specific substructure in the molecule. Here in this study, we use the 166-bit version MACCS fingerprints. For the Morgan or Circular fingerprints, we use the Morgan fingerprints with radius

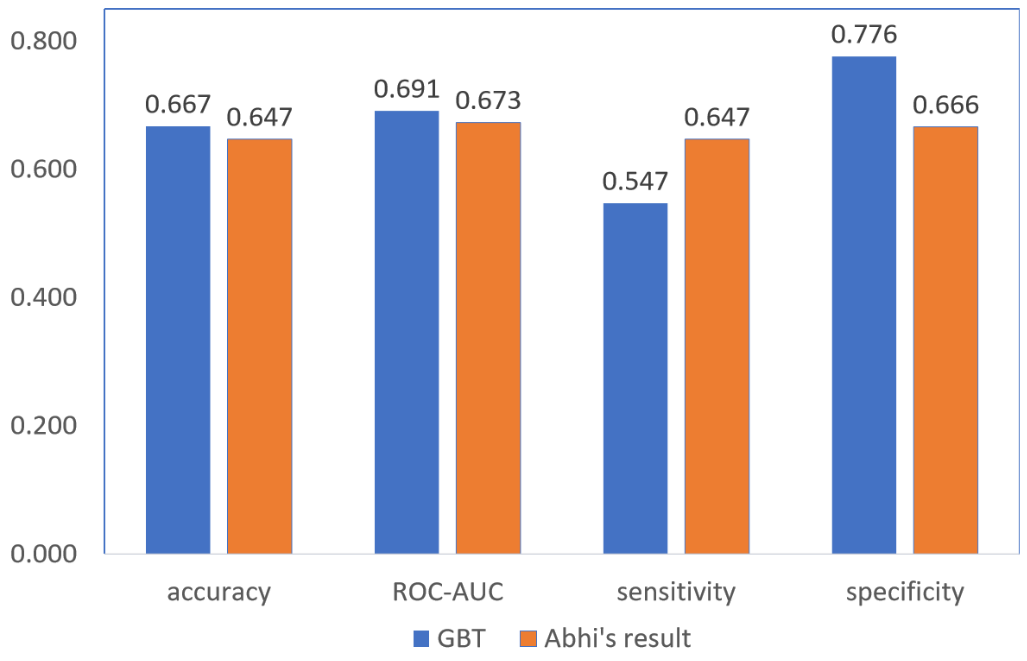


Figure 7.5: Prediction results of our model using the proposed CTFs and the Gradient Boosting Trees Classifier on external testing dataset AID1189. Our result is indicated in blue color and Abhil's result is indicated in orange color.

	Accuracy	Sensitivity	Specificity
Gradient Boosting	0.838	0.951	0.565
Random Forest	0.793	1.00	0.294
Extra Trees	0.810	0.976	0.412
Multi-task DNN	0.845	1.00	0.471

Table 7.3: Prediction result of our model using CTFs alone on the developmental toxicity dataset.

2 and 2048 bits. The Estate fingerprints refers to Kier and Halls electrotopological-state index descriptors [105]. FP4 is another substructure based fingerprints created from a list of SMART patterns defining functional groups. All of the fingerprints are generated using the RDkit software [34]. The comparison of our proposed CTFs and the aforementioned types of fingerprints can be found in Figure 7.8.

It shows in Figure 7.8 that our proposed CTFs performs better than all the other fingerprints in general. Our model using topological features from CTFs alone generates compa-

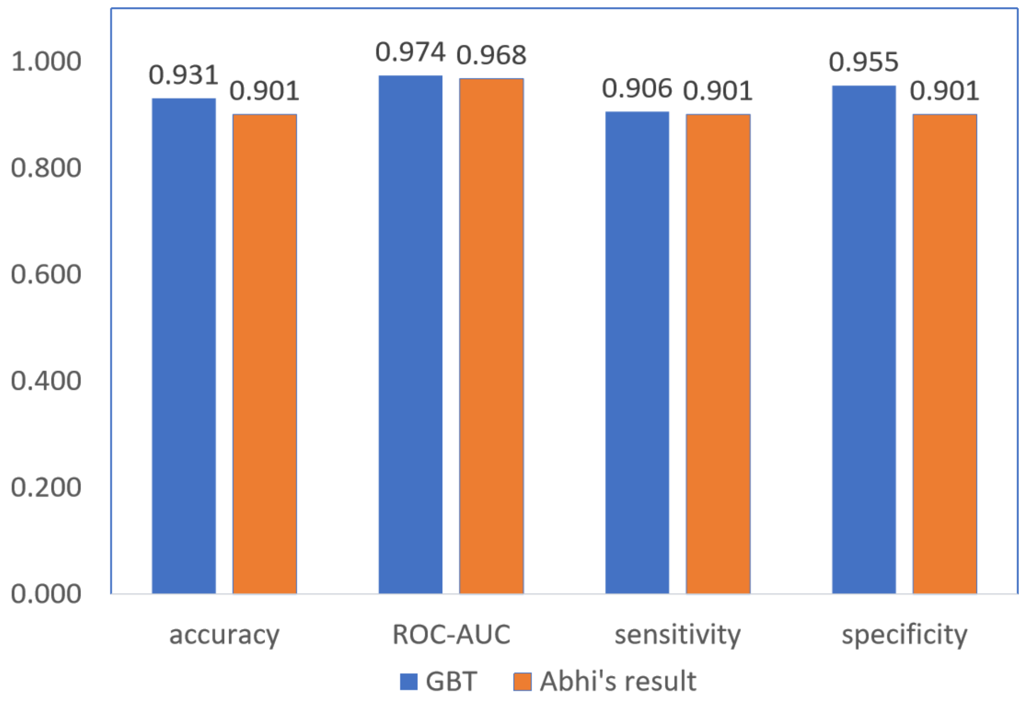


Figure 7.6: Prediction result of our model using the proposed CTFs and the Gradient Boosting Trees Classifier on external testing dataset AID1194. Our result is indicated in blue color and Abhil's result is indicated in orange color.

rable results with what is reported in T.E.S.T. software [14].

At this point, we would like to see if adding some auxiliary features mentioned in previous sections would help to improve the predictive performance on the developmental toxicity dataset. The results are reported in Table 7.4 and Table 7.5. Unfortunately, these auxiliary features does not help to improve the predictive performance of our model. When we rank the features' importance according to the gradient boosting trees algorithm, we see that the majority of the auxiliary features have very low ranking compared with the topological features of CTFs.

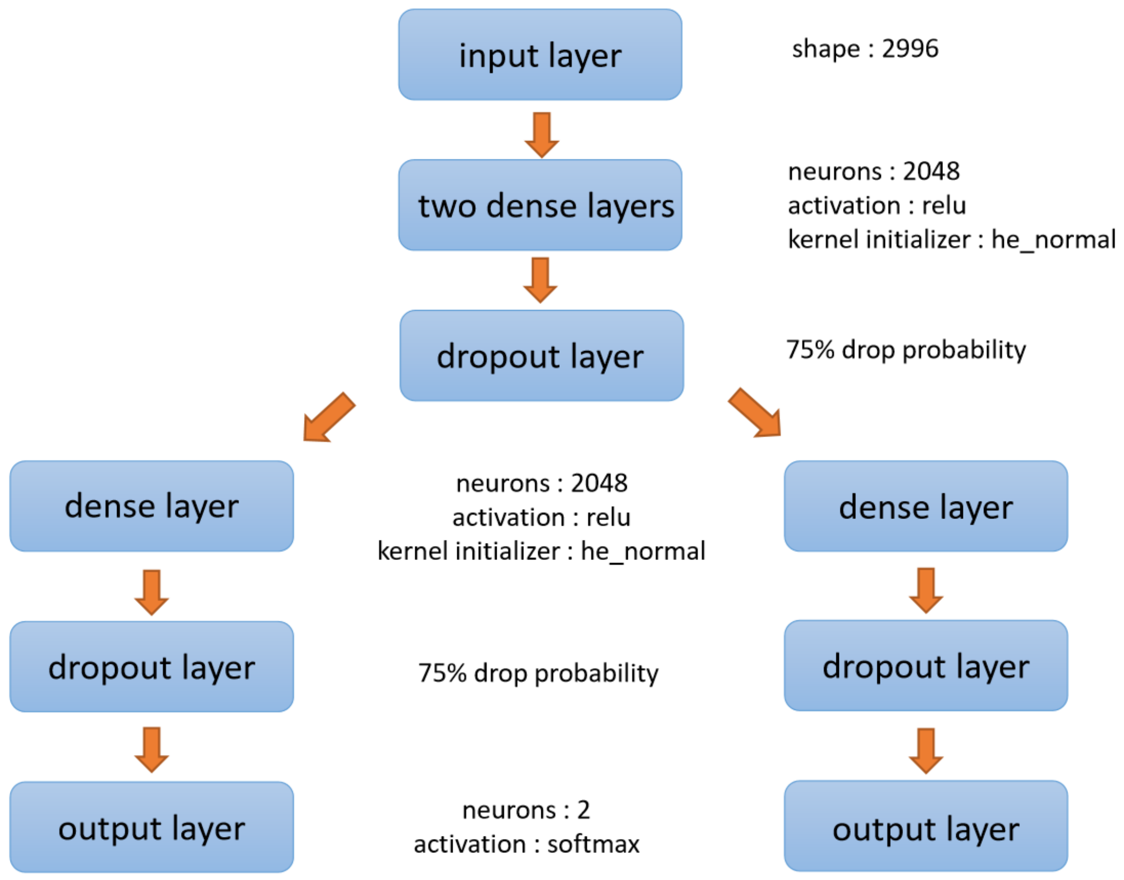


Figure 7.7: Illustration of the architecture of the multi-task neural networks used in our study for predicting chemical mutagenic and developmental toxic outputs simultaneously. The first few layers of the networks are shared by both tasks, consisting of an input layer, two dense (fully connected) layers, and a dropout layer. The rest of the layers are task-specific and not shared. Parameters corresponding to each layer are also provided in the chart.

	Accuracy	ROC-AUC	Sensitivity	Specificity
Gradient Boosting	0.831	0.797	0.949	0.547
Random Forest	0.793	0.803	0.976	0.353
Extra Trees	0.810	0.813	0.976	0.412

Table 7.4: Prediction result using CTFs and feature functional theory (FFT) based features on the developmental toxicity dataset.

	Accuracy	ROC-AUC	Sensitivity	Specificity
Gradient Boosting	0.826	0.796	0.963	0.494
Random Forest	0.793	0.824	1.00	0.294
Extra Trees	0.810	0.816	0.976	0.412

Table 7.5: Prediction result using CTFs and RDKit features on the developmental toxicity dataset.

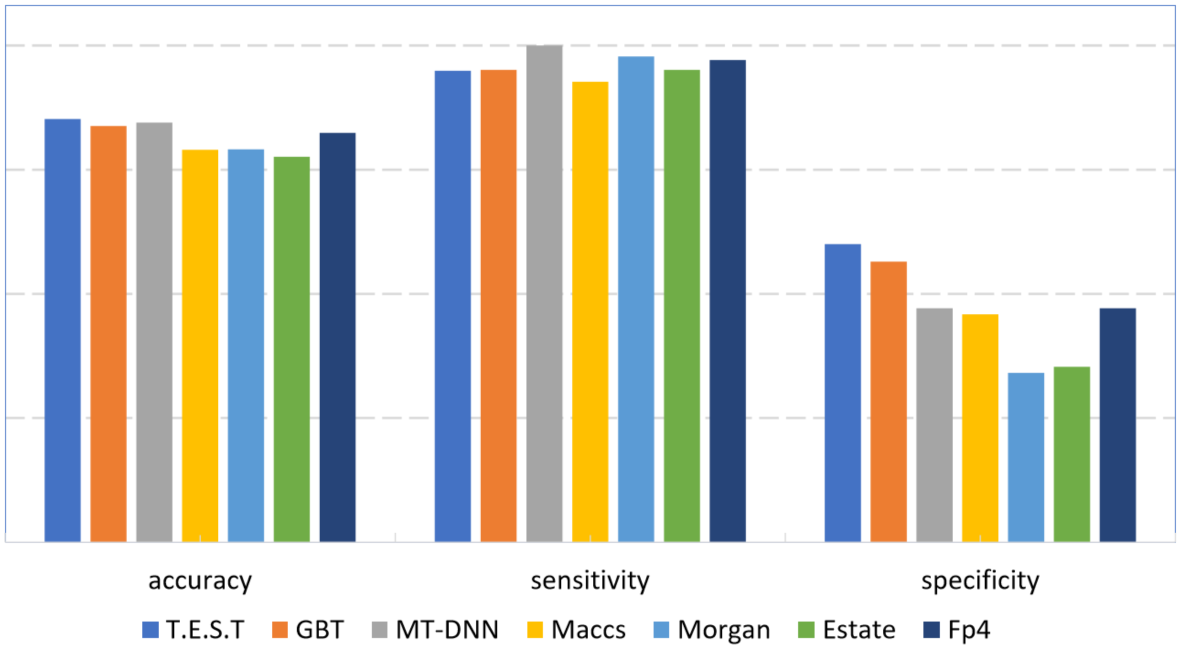


Figure 7.8: Performance comparison of our proposed CTFs with some existing fingerprints on the developmental toxicity dataset. The results reported in T.E.S.T. software is indicated in blue, the result using our proposed CTFs and gradient boosting trees is indicated in orange, the result using the proposed CTFs and multi-task deep neural networks is indicated in grey, results using the MACCS fingerprints, Morgan fingerprints, Estate fingerprints, and FP4 fingerprints are indicated in yellow, light blue, light green, and dark green, respectively.

Chapter 8

Discussion

In this study, we propose the CTF by including pre-designed topological features for 28 combinations in the feature space. The selection of the 28 combinations of elements is purely based on their occurrence frequencies in the training set, that is to say, we only select combinations with high occurrence frequencies. Previous studies have already shown that the occurrence frequency of a given combination is highly correlated with its corresponding predictive power (see Figure 5.5). These combinations with high occurrence frequencies might be important for the model to obtain a high overall accuracy, however, for the mutagenic prediction of a specific molecule, the missing of some of the low-frequency combinations might be crucial. For example, if the queried molecule contains some element that is not frequently seen in the training set, then combinations containing this "rare" element might play a crucial role in predicting the mutagenicity for this specific molecule, even if its occurrence frequency in the training set is quite low. Especially when these "rare" elements such as the Bromine (Br) are usually healthy and environmental concerns and chemical molecules containing such elements have a much higher possibility to be mutagenic. To make it more clear, take the 1-AMINO-2,4-DIBROMOANTHRAQUINONE (CID: 6681; molecular formular: $C_{14}H_7Br_2NO_2$) for instance, since there are two Bromine atoms in its structure, combinations such as {C,Br} and {C,O,Br} might play more important roles than any of the combinations listed in Table 5.2 in the mutagenic prediction of this specific molecule.

Thus, to examine the effects of combinations containing such "rare" elements but with low occurrence frequencies and to further improve the predictive performance (especially the true positive rate or sensitivity) of our model in the future, we have expanded the set of combinations by including some low-frequency combinations containing "rare" elements. Moreover, we would also like to know, for a specific "rare" element, whether those molecules containing this element in the training set would help to improve the prediction performance.

Thus, for a specific "rare" element, namely X (here X can be any element in S, P, F, Cl, I, and Br), we have designed two training sets:

(a) All 7032 molecules in the original training set

(b) A "shrunk" training set by taking only those molecules in the original training set that do not contain element X

And we have also designed two sets of topological features:

(a) Topological features from the 28 combinations listed in Table 5.2

(b) Topological features from an "expanded" set of combinations

Here for the "expanded" set of combinations, besides the 28 combinations listed in Table 5.2, we create another 28 combinations by inserting element X into each one of the listed 28 combination(if X is not already in it). After which the newly created combinations are added to the original 28 combinations with duplicated combinations removed.

Two new testing sets are designed by modifying the original testing sets AID1189 and AID1194:

(a) Subset of the original AID1189 dataset by only taking molecules that contain element X

(b) Subset of the original AID1194 dataset by only taking molecules that contain element X

Thus, for each "rare" element X, four different experiments are performed, which are described as follows:

1. Build the model with all 7032 molecules in the original training set, use the "expanded" set of combinations (58 or less), to make predictions on the subset of AID1189 and the subset of AID1194.

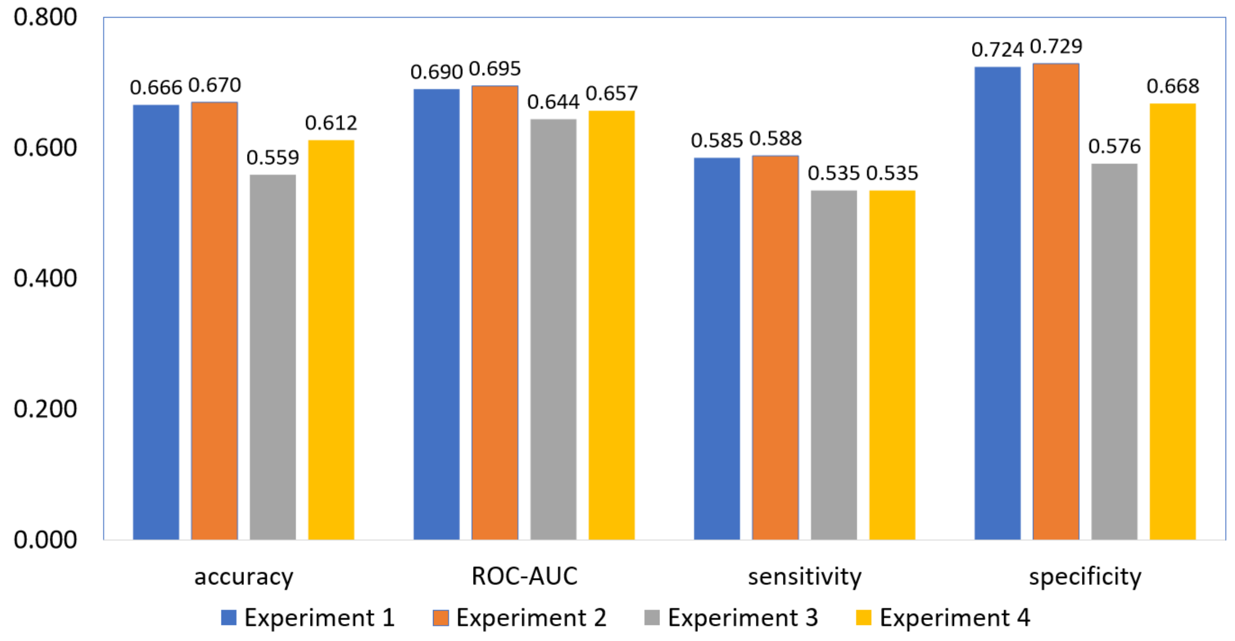
2. Build the model with all 7032 molecules in the original training set, use the original 28 combinations listed in Table 5.2, to make predictions on the subset of AID1189 and the subset of AID1194.

3. Build the model with the "shrunk" training set mentioned above, use the "expanded" set of combinations (58 or less), to make predictions on the subset of AID1189 and the subset of AID1194.

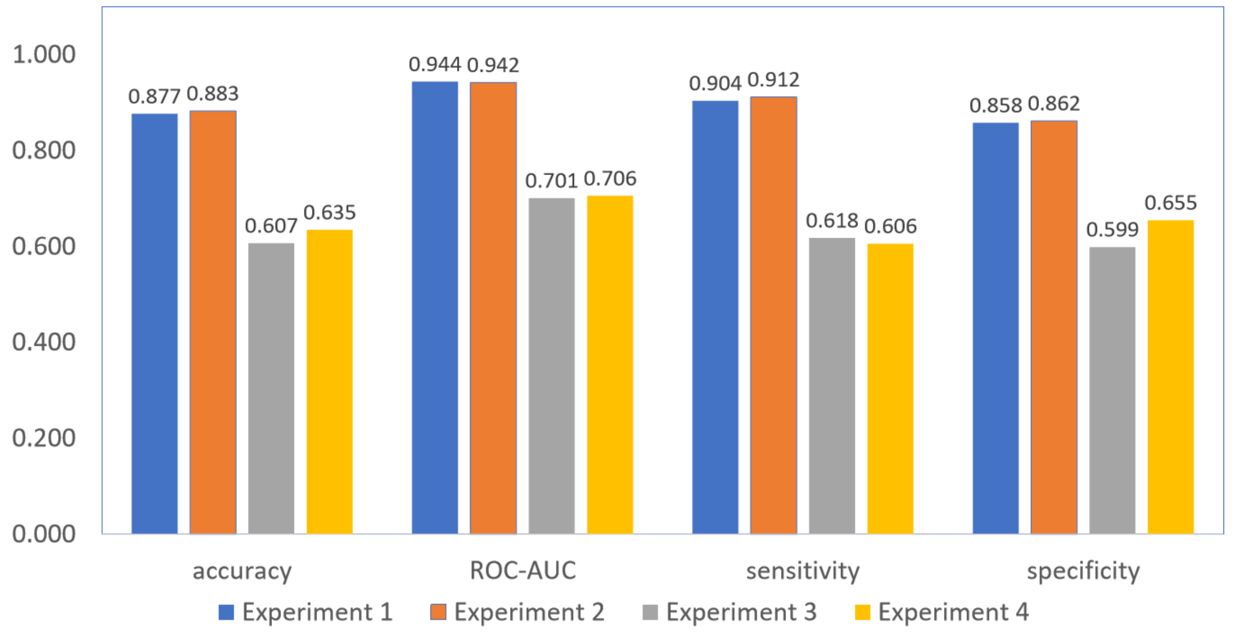
4. Build the model with the "shrunk" training set mentioned above, use the original 28 combinations listed in Table 5.2, to make predictions on the subset of AID1189 and the subset of AID1194.

For each "rare" element, we compare the predictive performances of the aforementioned experiments. The comparisons are shown in Figure 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, respectively.

The experimental results from Figure 8.1 to Figure 8.6 show that, in general, the predictive performances of models using the original 7032 molecules perform better than those built on "shrunk" training sets. It suggests that to predict the mutagenic outcome for molecules containing certain "rare" elements, molecules in the training set that also contain the same "rare" element play very important roles in building an accurate model. This also agrees

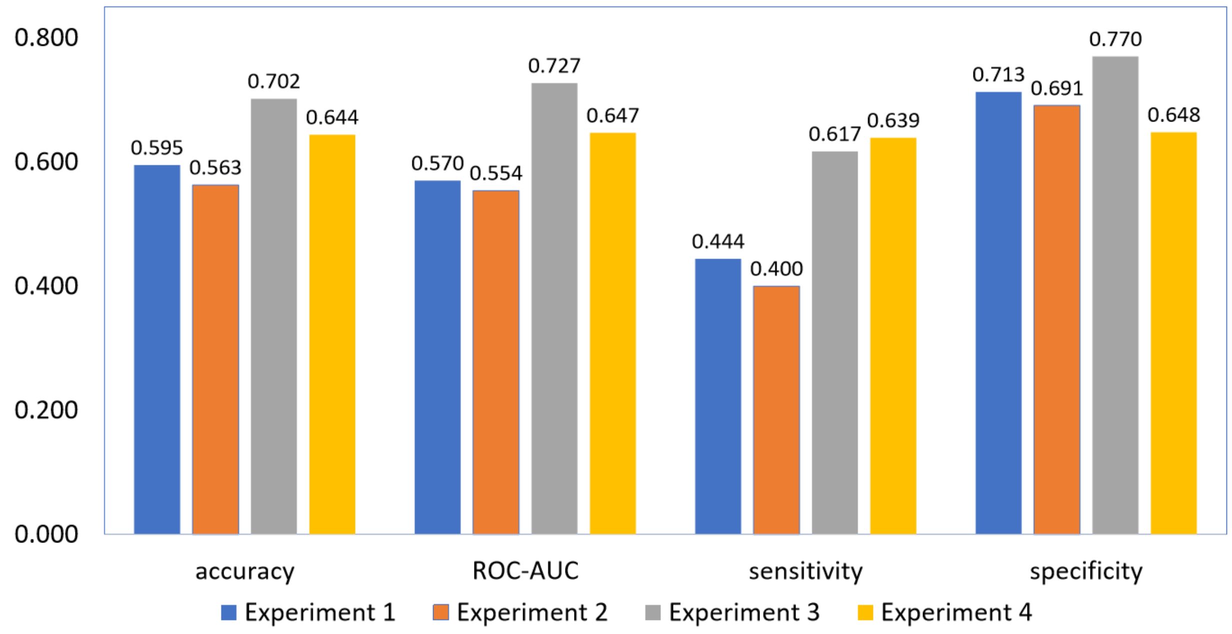


(a)

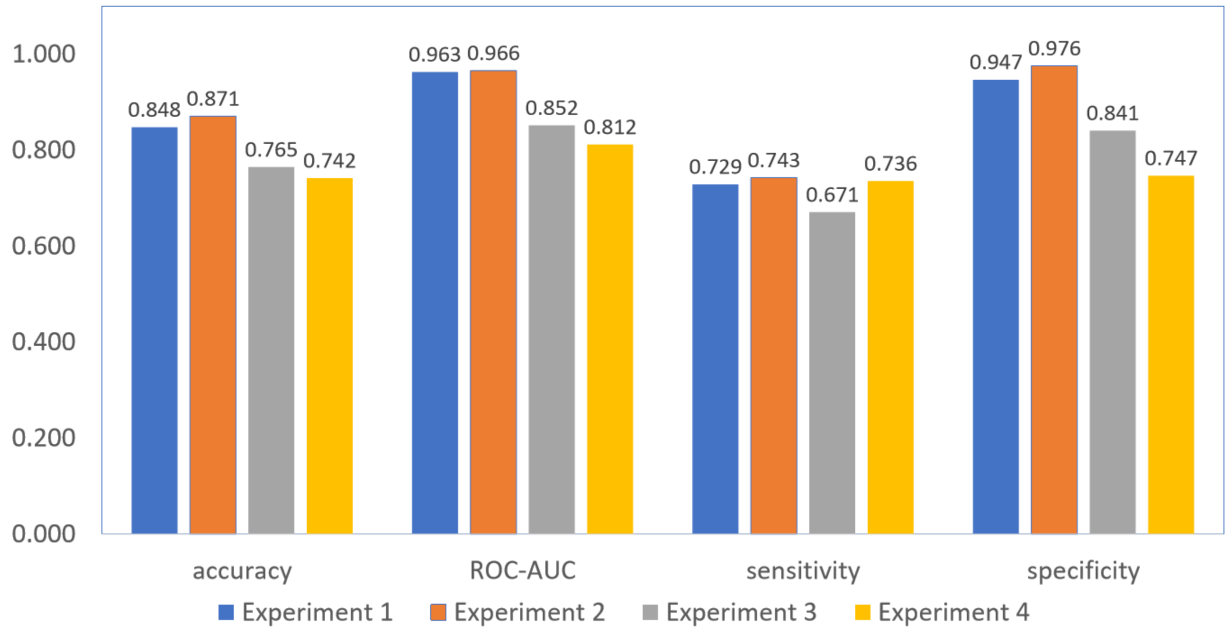


(b)

Figure 8.1: Predictive results of four experiments for Sulfur (S). The top figure shows the predictive results on the subset of AID1189, the bottom one is the predictive results on the subset of AID1194.

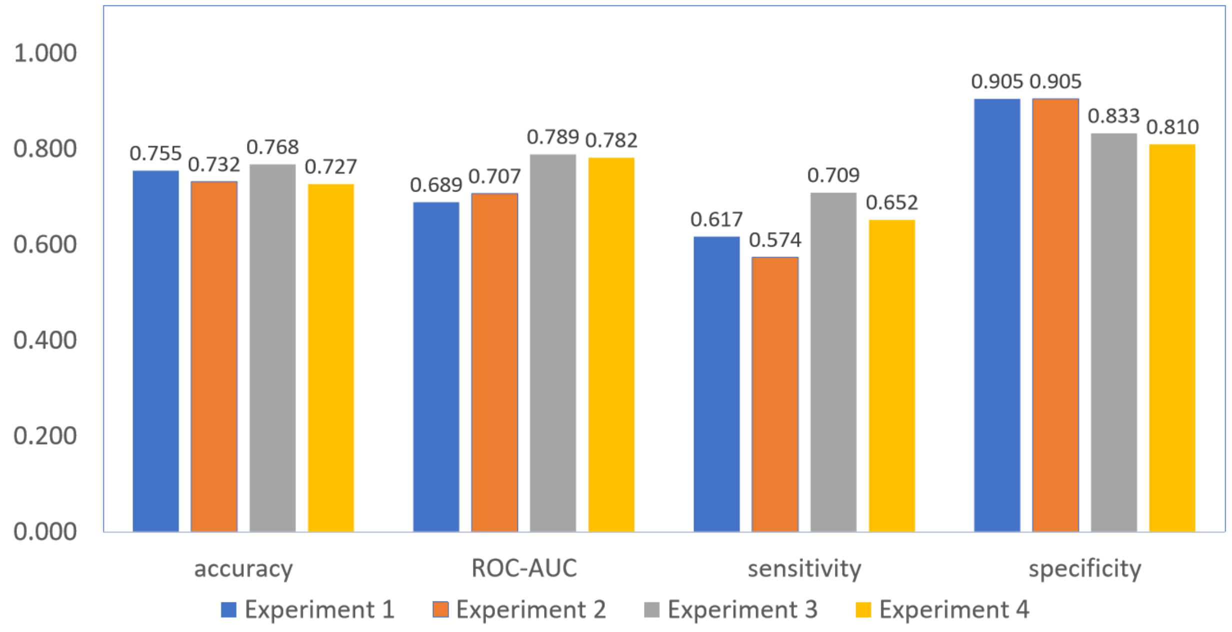


(a)

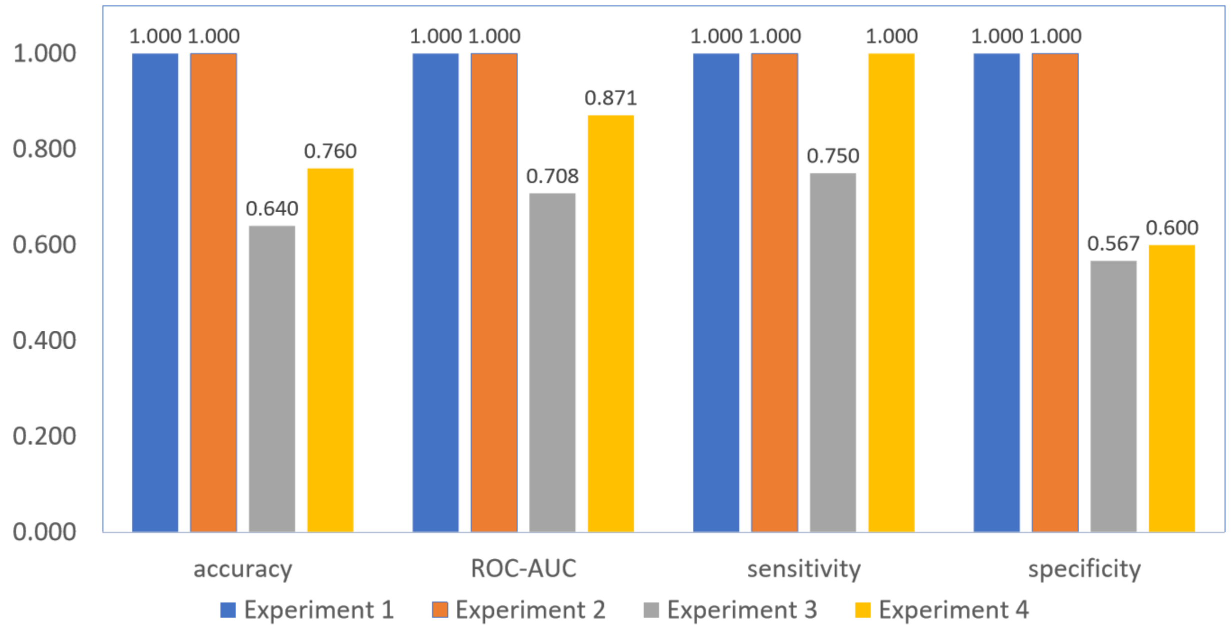


(b)

Figure 8.2: Predictive results of four experiments for Phosphorus (P). The top figure shows the predictive results on the subset of AID1189, the bottom one is the predictive results on the subset of AID1194.

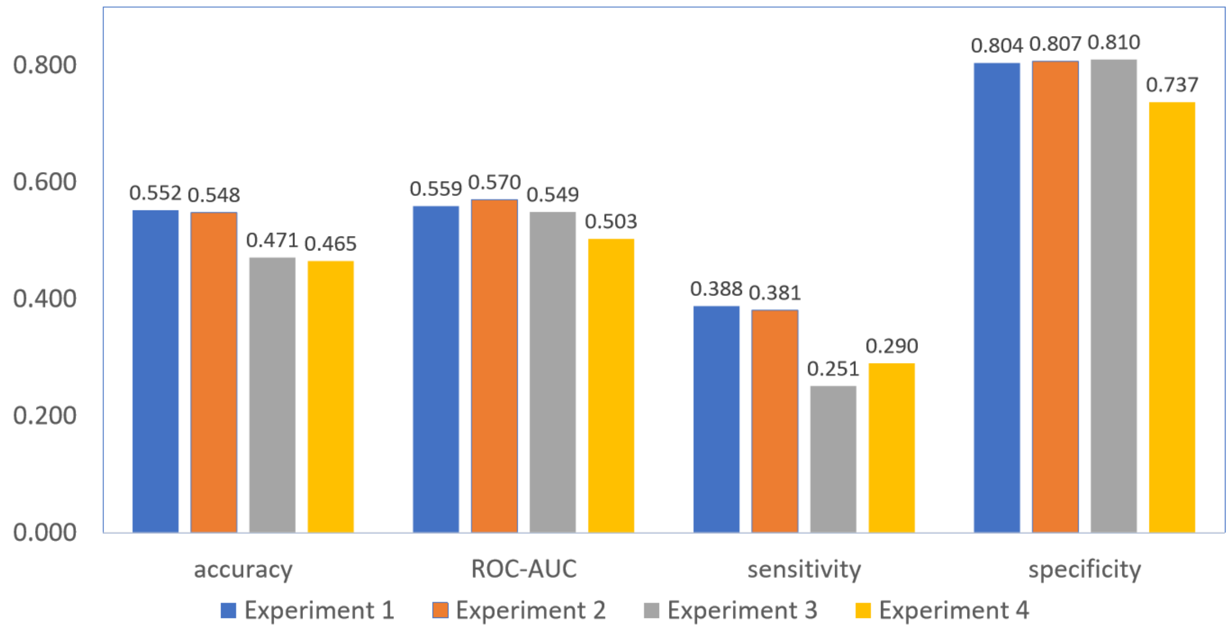


(a)

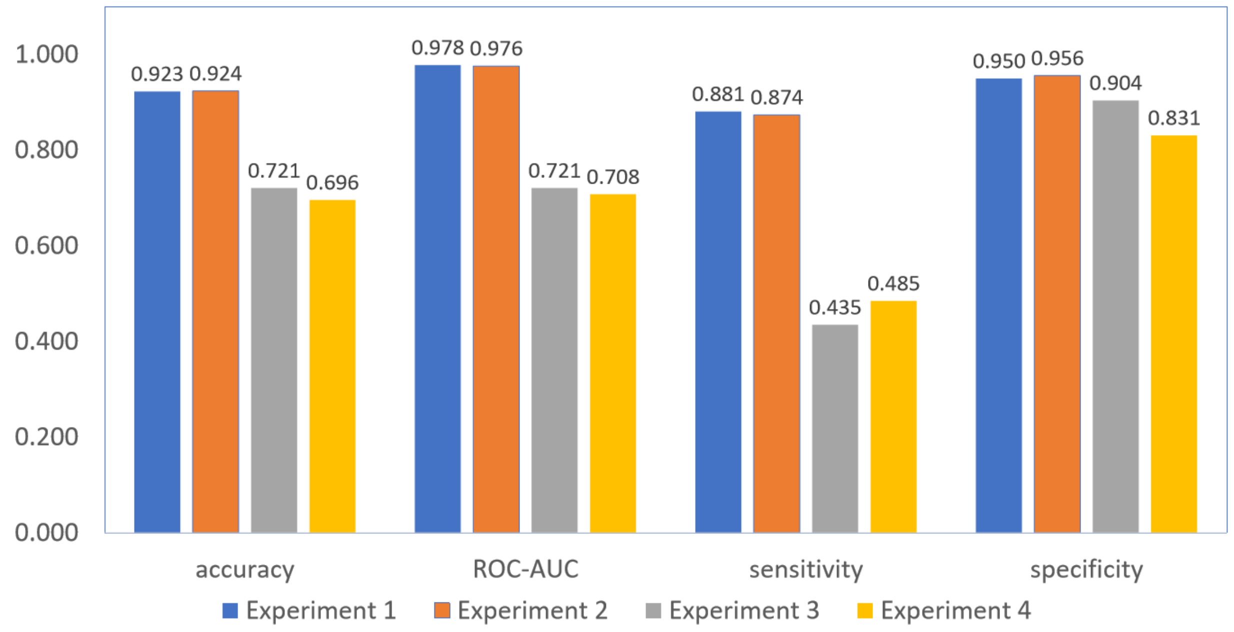


(b)

Figure 8.3: Predictive results of four experiments for Fluorine (F). The top figure shows the predictive results on the subset of AID1189, the bottom one is the predictive results on the subset of AID1194.

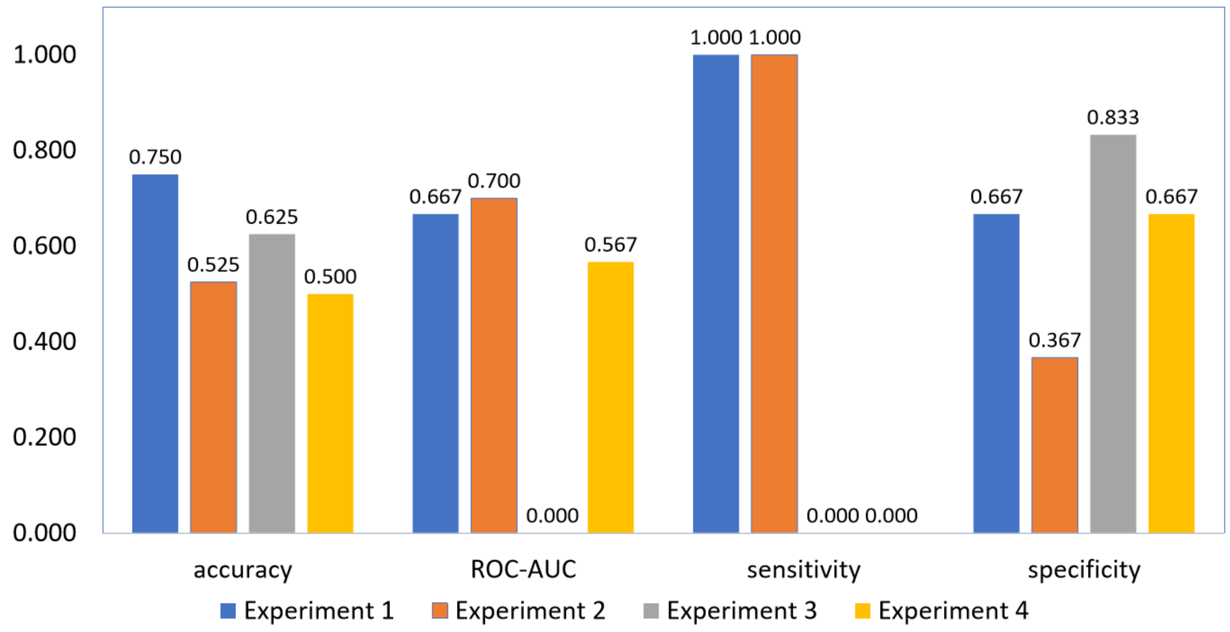


(a)

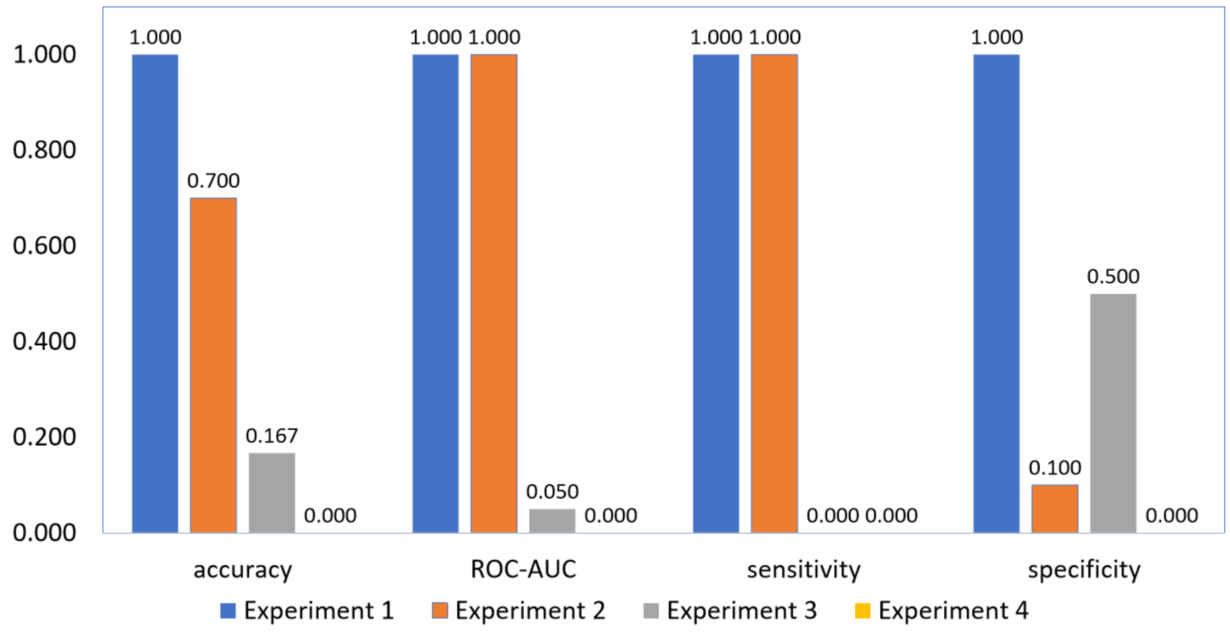


(b)

Figure 8.4: Predictive results of four experiments for Chlorine (Cl). The top figure shows the predictive results on the subset of AID1189, the bottom one is the predictive results on the subset of AID1194.

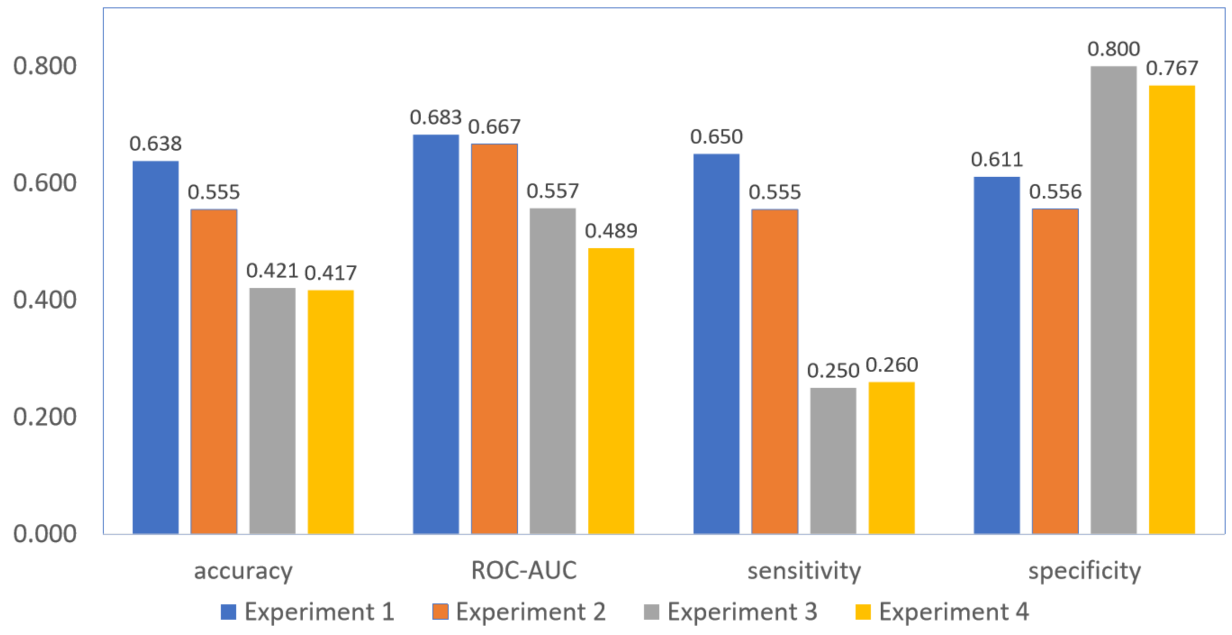


(a)

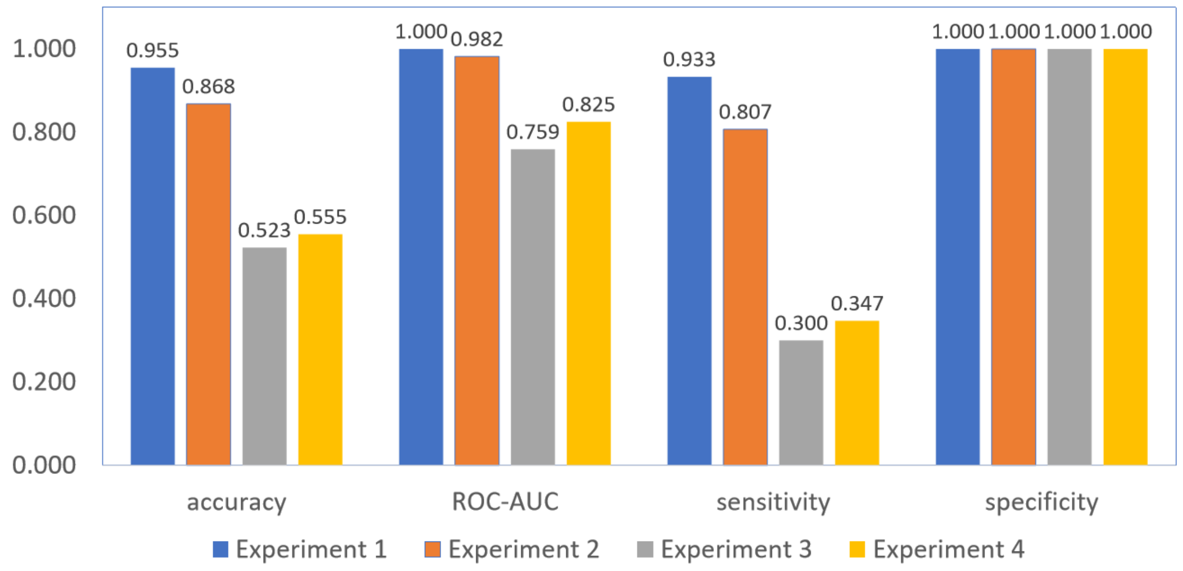


(b)

Figure 8.5: Predictive results of four experiments for Iodine (I). The top figure shows the predictive results on the subset of AID1189, the bottom one is the predictive results on the subset of AID1194.



(a)



(b)

Figure 8.6: Predictive results of four experiments for Bromine (Br). The top figure shows the predictive results on the subset of AID1189, the bottom one is the predictive results on the subset of AID1194.

with our knowledge that automated machine learning algorithms are usually not good for doing extrapolation (the samples to be predicted are out of the data space in which the model or algorithm learned). The figures also show that topological features specific to the "rare" element are also crucial, as we can see that models using the "expanded" set of combinations perform better, in general, than those of using only the 28 combinations listed in Table 5.2. Specifically, for each "rare" element, the exclusion of samples containing this element in training set and the exclusion of features specific to this element will both cause degenerated results. This phenomenon suggests the possibility of building an adaptive model that could adapt to different compositions of the testing samples instead of one simple model.

Chapter 9

Conclusion

The need of a more comprehensive topological representation for chemical molecules in toxicological risk assessment and the success of computational algebraic topology in topological data analysis (TDA) to capture the mutual connectivities between points in space have motivated us to design and propose the combinatorial topological fingerprints (CTFs) for building computational models to predict chemical toxicities, especially mutagenicity and developmental toxicity, two of the most important toxicological endpoints.

In the present work, persistent homology (PH) is introduced for the qualitative prediction of chemical toxic activities, whose theoretical concepts and algorithms are briefly reviewed, including simplex, simplicial complex, chain, filtration, and different criteria of constructing complexes during the filtration process such as Čech complex, Alpha complex, and Vietoris-Rips complex.

The proposed CTFs are a new way of encoding the structure of a molecule, which are designed from a sequence of persistent homology barcodes in three dimensions (dimension 0, 1, and 2) by taking different combinations of element types. Molecular structural information such as chemical bonds, atom-atom connectivities, and substructures such as rings and voids are all encoded in CTFs. The topological features of CTFs could serve perfectly as input to modern machine learning algorithms.

A number of machine learning algorithms are examined and compared, including single learning algorithms such as support vector machines, (shallow) neural networks; ensemble of trees methods such as random forest, extra trees, and gradient boosting trees; and multi-task deep neural networks (DNNs), among which the gradient boosting trees and multi-task DNNs produce some of the best results.

The investigation of the multi-task DNNs in this work is inspired by the success of deep learning methods (especially multi-task DNNs) in recent Tox21 data challenge. The predictive sensitivity (the most important metric in toxicity prediction) of the multi-task DNN model over the developmental toxicity dataset can be as high as 100%, which surpasses all other machine learning models.

The topological features that we designed from persistent homology filtration barcodes could retain the essential structural information without over-simplifying biomolecular complexity. The representability of the proposed CTFs is demonstrated by a number of computational experiments. Combining CTFs and various machine learning algorithms, we have built models that outperform existing models in literature in predicting chemical mutagenicity. In predicting developmental toxicity, models built with CTFs could make predictions comparable to (sometimes even better than) the best result reported by Environmental Protection Agency (EPA) over the developmental dataset created by Arena and coworkers. Experimental results over the developmental toxicity dataset have also shown the advantage of the proposed CTFs over some of the existing fingerprints.

Some directions of future works include:

1. Further study of small molecular featurization. In this work, 2996 topological features are selected from 28 predetermined combinations of element types according to each of the combination's occurrence frequency in the training set. But the selection of these combinations, as well as the design of topological features can be further studied. An

automatic way of selecting the number of combinations and the related topological features (length of the CTFs) is also of great concern.

2. Different ways of training the multi-task DNNs as well as different architectures of the networks should be further explored. In the current study when training the multi-task DNN model, the mutagenicity dataset (large set) and the developmental toxicity dataset (small set) are trained alternatively in all shared layers. We could also control how many training samples from each set go into a minibatch. For example, in the case of two types of toxicities (mutagenicity and developmental toxicity), we could create minibatches of 30 samples by selecting 20 samples at random from the small set (the one we would like to emphasize) and 10 samples from the large set. Different architectures of networks with respects to the width and depth of the network should also be further examined.

3. There are a large number of datasets that could be studied and investigated. The proposed CTFs could easily be generalized to the study of different toxicity endpoints besides mutagenicity and developmental toxicity.

4. In the present study, qualitative predictions are performed. In the future, the quantitative prediction could be studied in the same setting, such as predicting the value of IC50.

Chapter 10

Contribution

In this work, we propose the Combinatorial Topological Fingerprints (CTFs) based on persistent homology, which is a novel molecular topological representation with molecular structural information encoded. The predictive power and representability of the proposed CTFs are demonstrated by a number of computational experiments. The proposed CTFs can also be easily generalized to modeling (either qualitative or quantitative) of other types of toxicity endpoints.

Our second contribution of this work is the use of multi-task deep neural networks (DNNs) in an attempt to improve the predictive result over the developmental toxicity dataset. We suspect that data scarcity is one source of the performance bottlenecks for classic machine learning algorithms. Multi-task DNNs could make use of extra information in the training signals of related tasks (in our case, the mutagenicity prediction), which effectively increase the sample size. In the case when multiple tasks share some level of features, all tasks could benefit from computing a shared hidden layer feature of the inputs. The predictive sensitivity of the multi-task DNN model over the developmental toxicity dataset outperforms all other classic machine learning methods.

Besides the work presented in this paper, I have also worked on a project about designing numerical scheme to solve the Elliptic Interface Problems (EIPs), which is published in the Journal of Computational and Applied Mathematics [106].

Elliptic partial differential equations (PDEs) with discontinuous coefficients and singular source terms, commonly referred to as elliptic interface problems, occur in many applications, including fluid dynamics, material science, electromagnetics, biological systems, and heat or mass transfer. It is usually cast as elliptic Partial Differential Equations (PDEs) defined on piecewise-smooth subdomains coupling together via interface conditions.

A finite volume formulation of the matched interface and boundary (FVM-MIB) method is proposed to solve the two-dimensional and three-dimensional EIP. Compared with collocation formulation of the problem, the finite volume formulation and the related integral form have the merit in dealing with relatively low solution regularity.

The proposed FVM-MIB method takes the advantages of both MIB and FVM for solving EIPs, and its second order convergence in both L_∞ and L_2 norms are proved by a large number of numerical experiments of EIPs with complex interface geometries.

APPENDIX

Appendix

Supplementary materials

Auxiliary features

Besides topological features generated from the persistent homology barcodes, we have also studied some auxiliary features in an attempt to further improve the predictive performance of our model.

The first group auxiliary features are related to the geometric and electrostatic properties of the queried molecules. For each queried molecule, first we group all its atoms into 10 groups according to their element types. 10 element types are considered, they are {C, N, O, H, S, P, F, Cl, I, Br}. Some group may be empty when the queried molecule does not contain corresponding element type. Within each group associated with certain element type, we extract the most basic statistics such as the sum, mean, and so on from three categories: atomic surface area, atomic electrostatic charges, and atomic solvent free energies. Details of these element specific features are listed in Table S1.

Number	Description
1	sum of atomic surface areas
2	average of atomic surface areas
3	maximum of atomic surface areas
4	minimum of atomic surface areas
5	stand deviation of atomic surface areas
6	sum of atomic electrostatic charges
7	average of atomic electrostatic charges
8	maximal value of atomic electrostatic charges
9	minimal value of atomic electrostatic charges
10	standard deviation of atomic electrostatic charges
11	sum of atomic solvation free energies
12	average of atomic solvation free energies
13	maximum of atomic solvation free energies
14	minimum value of atomic solvation free energies
15	standard deviation of atomic solvation free energies

Table S1: For each element type, the element specific features listed in this table are computed, resulting in 15*10 = 150 element specific features.

The second group of auxiliary features is macroscopic features obtained from the RDKit software [34]. Only numerical descriptors are extracted.

Name	Description
	Physical properties
HeavyAtomMolWt	Average molecular weight of the molecule ignoring hydrogens
ExactMolWt	Exact molecule weight of the molecule
MolWt	Average molecular weight of the molecule
MolLogP	Wildman-Crippen LogP value
MolMR	Wildman-Crippen MR value
BalabanJ	Balaban's J value for a molecule.
BertzCT	A topological index meant to quantify "complexity" of molecules.
	Estate descriptors
MaxEStateIndex	Maximum of the Estate indices
MinEStateIndex	Minimum of the Estate indices
MaxAbsEStateIndex	Maximum of the absolute value of Estate indices
MinAbsEStateIndex	Minimum of the absolute value of Estate indices
VSA_EState1	VSA Estate descriptor 1
VSA_EState2	VSA Estate descriptor 2
VSA_EState3	VSA Estate descriptor 3
VSA_EState4	VSA Estate descriptor 4
VSA_EState5	VSA Estate descriptor 5
VSA_EState6	VSA Estate descriptor 6
VSA_EState7	VSA Estate descriptor 7
VSA_EState8	VSA Estate descriptor 8
VSA_EState9	VSA Estate descriptor 9
VSA_EState10	VSA Estate descriptor 10
	Hall & Kier connectivity and Kappa indices
Chi0	Atomic connectivity index (order 0)
Chi1	Atomic connectivity index (order 1)
Chi0v	Hall & Kier atomic connectivity index (order 0)
Chi1v	Hall & Kier atomic connectivity index (order 1)
Chi2v	Hall & Kier atomic connectivity index (order 2)
Chi3v	Hall & Kier atomic connectivity index (order 3)
Chi4v	Hall & Kier atomic connectivity index (order 4)
Chi0n	Modified Hall & Kier atomic connectivity index (order 0)
Chi1n	Modified Hall & Kier atomic connectivity index (order 1)
Chi2n	Modified Hall & Kier atomic connectivity index (order 2)
Chi3n	Modified Hall & Kier atomic connectivity index (order 3)
Chi4n	Modified Hall & Kier atomic connectivity index (order 4)
HallKierAlpha	Hall & Kier alpha value
Ipc	Information content of the coefficients of the characteristic polynomial of the adjacency of a hydrogen-suppressed graph of a molecule.
Kappa1	1st Kappa shape index
Kappa2	2nd Kappa shape index

Table S2: Molecular descriptors obtained from the RDKit software.

Table S2 (cont'd)

Kappa3	3rd Kappa shape index
LabuteASA	Labute's approximate surface area of a molecule
	Partial charge descriptors
MinAbsPartialCharge	Minimum of absolute values of partial charges in the molecule
MaxAbsPartialCharge	Maximum of absolute values of partial charges in the molecule
MinPartialCharge	Minimum of partial charges in the molecule
MaxPartialCharge	Maximum of partial charges in the molecule
PEOE_VSA1	MOE charge VSA descriptor 1
PEOE_VSA2	MOE charge VSA descriptor 2
PEOE_VSA3	MOE charge VSA descriptor 3
PEOE_VSA4	MOE charge VSA descriptor 4
PEOE_VSA5	MOE charge VSA descriptor 5
PEOE_VSA6	MOE charge VSA descriptor 6
PEOE_VSA7	MOE charge VSA descriptor 7
PEOE_VSA8	MOE charge VSA descriptor 8
PEOE_VSA9	MOE charge VSA descriptor 9
PEOE_VSA10	MOE charge VSA descriptor 10
PEOE_VSA11	MOE charge VSA descriptor 11
PEOE_VSA12	MOE charge VSA descriptor 12
PEOE_VSA13	MOE charge VSA descriptor 13
PEOE_VSA14	MOE charge VSA descriptor 14
	Subdivided surface areas
SMR_VSA1	MOE MR VSA descriptor 1
SMR_VSA2	MOE MR VSA descriptor 2
SMR_VSA3	MOE MR VSA descriptor 3
SMR_VSA4	MOE MR VSA descriptor 4
SMR_VSA5	MOE MR VSA descriptor 5
SMR_VSA6	MOE MR VSA descriptor 6
SMR_VSA7	MOE MR VSA descriptor 7
SMR_VSA8	MOE MR VSA descriptor 8
SMR_VSA9	MOE MR VSA descriptor 9
SMR_VSA10	MOE MR VSA descriptor 10
SlogP_VSA1	MOE logP VSA descriptor 1
SlogP_VSA2	MOE logP VSA descriptor 2
SlogP_VSA3	MOE logP VSA descriptor 3
SlogP_VSA4	MOE logP VSA descriptor 4
SlogP_VSA5	MOE logP VSA descriptor 5
SlogP_VSA6	MOE logP VSA descriptor 6
SlogP_VSA7	MOE logP VSA descriptor 7
SlogP_VSA8	MOE logP VSA descriptor 8
SlogP_VSA9	MOE logP VSA descriptor 9
SlogP_VSA10	MOE logP VSA descriptor 10
SlogP_VSA11	MOE logP VSA descriptor 11

Table S2 (cont'd)

SlogP_VSA12	MOE logP VSA descriptor 12
EState_VSA1	EState VSA descriptor 1
EState_VSA2	EState VSA descriptor 2
EState_VSA3	EState VSA descriptor 3
EState_VSA4	EState VSA descriptor 4
EState_VSA5	EState VSA descriptor 5
EState_VSA6	EState VSA descriptor 6
EState_VSA7	EState VSA descriptor 7
EState_VSA8	EState VSA descriptor 8
EState_VSA9	EState VSA descriptor 9
EState_VSA10	EState VSA descriptor 10
EState_VSA11	EState VSA descriptor 11
TPSA	Molecular topological polar surface area
Molecular fragments	
NumRadicalElectrons	Number of radical electrons in the molecule
NumValenceElectrons	Number of valence electrons in the molecule
HeavyAtomCount	Number of heavey atoms in a molecule
NHOHCount	Number of NHs or OHs
NOCount	Number of nitrogens and oxygens
FractionCSP3	Fraction of C atoms that are SP3 hybridized
NumAliphaticCarbocycles	Number of aliphatic carbocycles for a molecule
NumAliphaticHeterocycles	Number of aliphatic heterocycles for a molecule
NumAliphaticRings	Number of aliphatic rings for a molecule
NumAromaticCarbocycles	Number of aromatic carbocycles for a molecule
NumAromaticHeterocycles	Number of aromatic heterocycles for a molecule
NumAromaticRings	Number of aromatic rings for a molecule
NumHAcceptors	Number of hydrogen bond acceptors
NumHDonors	Number of hydrogen bond donors
NumHeteroatoms	Number of heteroatoms
NumRotatableBonds	Number of rotatable bonds
NumSaturatedCarbocycles	Number of saturated carbocycles for a molecule
NumSaturatedHeterocycles	Number of saturated heterocycles for a molecule
NumSaturatedRings	Number of saturated rings for a molecule
RingCount	Number of rings in a molecule
fr_Al_COO	Number of aliphatic carboxylic acids
fr_Al_OH	Number of aliphatic hydroxyl groups
fr_Al_OH_noTert	Number of aliphatic hydroxyl groups excluding tert-OH
fr_ArN	fr_Al_COO Number of N functional groups attached to aromatics
fr_Ar_COO	Number of Aromatic carboxylic acids
fr_Ar_N	Number of aromatic nitrogens
fr_Ar_NH	Number of aromatic amines
fr_Ar_OH	Number of aromatic hydroxyl groups
fr_COO	Number of carboxylic acids

Table S2 (cont'd)

fr_COO2	Number of carboxylic acids
fr_C_O	Number of carbonyl
fr_C_O_noCOO	Number of carbonyl O, excluding COOH
fr_C_S	Number of thiocarbonyl
fr_HOCCN	Number of C(OH)CCN-Ctert-alkyl or C(OH)CCNcyclic
fr_Imine	Number of Imines
fr_NH0	Number of Tertiary amines
fr_NH1	Number of Secondary amines
fr_NH2	Number of Primary amines
fr_N_O	Number of hydroxylamine groups
fr_Ndealkylation1	Number of XCCNR groups
fr_Ndealkylation2	Number of tert-alicyclic amines
fr_Nhpyrrole	Number of H-pyrrole nitrogens
fr_SH	Number of thiol groups
fr_aldehyde	Number of aldehydes
fr_alkyl_carbamate	Number of alkyl carbamates
fr_alkyl_halide	Number of alkyl halides
fr_allylic_oxid	Number of allylic oxidation sites excluding steroid dienone
fr_amide	Number of amides
fr_amidine	Number of amidine groups
fr_aniline	Number of anilines
fr_aryl_methyl	Number of aryl methyl sites for hydroxylation
fr_azide	Number of azide groups
fr_azo	Number of azo groups
fr_barbitur	Number of barbiturate groups
fr_benzene	Number of benzene rings
fr_benzodiazepine	Number of benzodiazepines with no additional fused rings
fr_bicyclic	Number of bicyclic rings
fr_diazo	Number of diazo groups
fr_dihydropyridine	Number of dihydropyridines
fr_epoxide	Number of epoxide rings
fr_ester	Number of esters
fr_ether	Number of ether oxygens (including phenoxy)
fr_furan	Number of furan rings
fr_guanido	Number of guanidine groups
fr_halogen	Number of halogens
fr_hdrzine	Number of hydrazine groups
fr_hdrzone	Number of hydrazone groups
fr_imidazole	Number of imidazole rings
fr_imide	Number of imide groups
fr_isocyan	Number of isocyanates
fr_isothiocyan	Number of isothiocyanates
fr_ketone	Number of ketones

Table S2 (cont'd)

fr_ketone_Topliss	Number of ketones excluding diaryl, a,b-unsat.
fr_lactam	Number of beta lactams
fr_lactone	Number of cyclic esters (lactones)
fr_methoxy	Number of methoxy groups -OCH3
fr_morpholine	Number of morpholine rings
fr_nitrile	Number of nitriles
fr_nitro	Number of nitro groups
fr_nitro_arom	Number of nitro benzene ring substituents
fr_nitro_arom_nonortho	Number of non-ortho nitro benzene ring substituents
fr_nitroso	Number of nitroso groups, excluding NO2
fr_oxazole	Number of oxazole rings
fr_oxime	Number of oxime groups
fr_para_hydroxylation	Number of para-hydroxylation sites
fr_phenol	Number of phenols
fr_phenol_noOrthoHbond	Number of phenolic OH excluding ortho intramolecular Hbond substituents
fr_phos_acid	Number of phosphoric acid groups
fr_phos_ester	Number of phosphoric ester groups
fr_piperdine	Number of piperdine rings
fr_piperzine	Number of piperzine rings
fr_priamide	Number of primary amides
fr_prisulfonamd	Number of primary sulfonamides
fr_pyridine	Number of pyridine rings
fr_quatN	Number of quarternary nitrogens
fr_sulfide	Number of thioether
fr_sulfonamd	Number of sulfonamides
fr_sulfone	Number of sulfone groups
fr_term_acetylene	Number of terminal acetylenes
fr_tetrazole	Number of tetrazole rings
fr_thiazole	Number of thiazole rings
fr_thiocyan	Number of thiocyanates
fr_thiophene	Number of thiophene rings
fr_unbrch_alkane	Number of unbranched alkanes of at least 4 members (excludes halogenated alkanes)
fr_urea	Number of urea groups

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Kristien Mortelmans and Errol Zeiger. The ames salmonella/microsome mutagenicity assay. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, 455(1):29–60, 2000.

[2] Elizabeth C Miller and James A Miller. Mechanisms of chemical carcinogenesis. *Cancer*, 47(S5):1055–1064, 1981.

[3] Alessandra Roncaglioni, Andrey A Toropov, Alla P Toropova, and Emilio Benfenati. In silico methods to predict drug toxicity. *Current opinion in pharmacology*, 13(5):802–806, 2013.

[4] Chiara Milan, Onofrio Schifanella, Alessandra Roncaglioni, and Emilio Benfenati. Comparison and possible use of in silico tools for carcinogenicity within reach legislation. *Journal of Environmental Science and Health, Part C*, 29(4):300–323, 2011.

[5] NPNJJCA Greene, PN Judson, JJ Langowski, and CA Marchant. Knowledge-based expert systems for toxicity and metabolism prediction: Derek, star and meteor. *SAR and QSAR in Environmental Research*, 10(2-3):299–314, 1999.

[6] Carol A Marchant, Katharine A Briggs, and Anthony Long. In silico tools for sharing data and knowledge on toxicity and metabolism: derek for windows, meteor, and vitic. *Toxicology mechanisms and methods*, 18(2-3):177–187, 2008.

[7] G Patlewicz, N Jeliazkova, RJ Safford, AP Worth, and B Aleksiev. An evaluation of the implementation of the cramer classification scheme in the toxtree software. *SAR and QSAR in Environmental Research*, 19(5-6):495–524, 2008.

[8] Yin-Tak Woo, David Y Lai, Mary F Argus, and Joseph C Arcos. Development of structure-activity relationship rules for predicting carcinogenic potential of chemicals. *Toxicology letters*, 79(1-3):219–228, 1995.

[9] Raghuraman Venkatapathy and Nina Ching Y Wang. Developmental toxicity prediction. *Computational Toxicology: Volume II*, pages 305–340, 2013.

[10] Oecd: The oecd qsar toolbox. Software available at `http://www.oecd.org/env/ehs/risk-assessment/theoecdqsartoolbox.htm`, 2017.

[11] Compudrug ltd. hazardexpert pro. Software available at `http://www.compudrug.com/hazardexpertpro`, 2013.

[12] Emilio Benfenati. The caesar project for in silico models for the reach legislation. *Chemistry Central Journal*, 4(1):I1, 2010.

[13] Computer assisted evaluation of industrial chemical substances according to regulations. Software available at `http://www.caesar-project.eu/`.

[14] U.S. Environmental Protection Agency. Test: Toxicity estimation software tool. Software available at `https://www.epa.gov/chemical-research/toxicity-estimation-software-tool-test`, 2016.

[15] XJ Yao, Annick Panaye, Jean-Pierre Doucet, RS Zhang, HF Chen, MC Liu, ZD Hu, and Bo Tao Fan. Comparative study of qsar/qspr correlations using support vector machines, radial basis function neural networks, and multiple linear regression. *Journal of chemical information and computer sciences*, 44(4):1257–1266, 2004.

[16] D Villeminl, D Cherqaouil, and JM Cense. Neural networks studies: quantitative structure-activity relationship of mutagenic aromatic. *J Chim Phys*, 90:1505–1519, 1993.

[17] David T Manallack and David J Livingstone. Neural networks in drug discovery: have they lived up to their promise? *European Journal of Medicinal Chemistry*, 34(3):195–208, 1999.

[18] James Devillers. Methods for building qsars. *Computational Toxicology: Volume II*, pages 3–27, 2013.

[19] Omar Deeb and Mohammad Goodarzi. In silico quantitative structure toxicity relationship of chemical compounds: some case studies. *Current drug safety*, 7(4):289–297, 2012.

[20] Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Graph kernels for molecular structure- activity relationship analysis with support vector machines. *Journal of chemical information and modeling*, 45(4):939–951, 2005.

[21] Zohre Hajisharifi, Moien Piryaiee, Majid Mohammad Beigi, Mandana Behbahani, and Hassan Mohabatkar. Predicting anticancer peptides with chou s pseudo amino acid composition and investigating their mutagenicity via ames test. *Journal of Theoretical Biology*, 341:34–40, 2014.

[22] Ashwin Srinivasan, Stephen H Muggleton, Michael JE Sternberg, and Ross D King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85(1-2):277–299, 1996.

[23] AK Madan, Sanjay Bajaj, and Harish Dureja. Classification models for safe drug molecules. *Computational Toxicology: Volume II*, pages 99–124, 2013.

[24] Johann Gasteiger and Thomas Engel. *Chemoinformatics: a textbook*. John Wiley & Sons, 2006.

[25] Artem Cherkasov, Eugene N Muratov, Denis Fourches, Alexandre Varnek, Igor I Baskin, Mark Cronin, John Dearden, Paola Gramatica, Yvonne C Martin, Roberto Todeschini, et al. Qsar modeling: where have you been? where are you going to? *Journal of medicinal chemistry*, 57(12):4977–5010, 2014.

[26] Andrew R Leach and Valerie J Gillet. *An introduction to chemoinformatics*. Springer Science & Business Media, 2007.

[27] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

[28] Ahmed Abdelaziz, Hilde Spahn-Langguth, Karl-Werner Schramm, and Igor V Tetko. Consensus modeling for hts assays using in silico descriptors calculates the best balanced accuracy in tox21 challenge. *Frontiers in Environmental Science*, 4:2, 2016.

[29] Roberto Todeschini and Viviana Consonni. *Handbook of molecular descriptors*, volume 11. John Wiley & Sons, 2008.

[30] Mati Karelson. *Molecular descriptors in QSAR/QSPR*. Wiley-Interscience, 2000.

[31] Roberto Todeschini and Viviana Consonni. *Molecular descriptors for chemoinformatics, volume 41 (2 volume set)*, volume 41. John Wiley & Sons, 2009.

[32] Andrea Mauri, Viviana Consonni, Manuela Pavan, and Roberto Todeschini. Dragon software: An easy approach to molecular descriptor calculations. *Match*, 56(2):237–248, 2006.

[33] R Todeschini, V Consonni, A Mauri, and M Pavan. Dragon-software for the calculation of molecular descriptors. *Web version*, 3, 2004.

[34] G Landrum et al. Rdkit: Cheminformatics and machine learning software. *RDKIT. ORG*, 2013.

[35] Chun Wei Yap. Padel-descriptor: An open source software to calculate molecular descriptors and fingerprints. *Journal of computational chemistry*, 32(7):1466–1474, 2011.

[36] Kejun Liu, Jun Feng, and S Stanley Young. Powermv: a software environment for molecular viewing, descriptor generation, data analysis and hit evaluation. *Journal of chemical information and modeling*, 45(2):515–522, 2005.

[37] Harry Wiener. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*, 69(1):17–20, 1947.

[38] Milan Randic. Characterization of molecular branching. *Journal of the American Chemical Society*, 97(23):6609–6615, 1975.

[39] James Devillers and Alexandru T Balaban. *Topological indices and related descriptors in QSAR and QSPAR*. CRC Press, 2000.

[40] Peter J Hansen and Peter C Jurs. Chemical applications of graph theory. part i. fundamentals and topological indices. *J. Chem. Educ*, 65(7):574, 1988.

[41] Alan R Katritzky and Ekaterina V Gordeeva. Traditional topological indexes vs electronic, geometrical, and combined molecular descriptors in qsar/qspr research. *Journal of chemical information and computer sciences*, 33(6):835–857, 1993.

[42] John C Dearden. The use of topological indices in qsar and qspr modeling. In *Advances in QSAR Modeling*, pages 57–88. Springer, 2017.

[43] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational homology*, volume 157. Springer Science & Business Media, 2006.

[44] Herbert Edelsbrunner. Persistent homology: theory and practice. 2014.

[45] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

[46] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 454–463. IEEE, 2000.

[47] Dmitriy Morozov. Dionysus: A c++ library for computing persistent homology. Software available at `http://www.mrzv.org/software/dionysus/`, 2012.

[48] Andrew Tausz, Mikael Vejdemo-Johansson, and Henry Adams. Javaplex: A research software package for persistent (co)homology. Software available at `http://code.google.com/p/javaplex`, 2011.

[49] Vidit Nanda. Perseus: the persistent homology software. *Software available at http://www. sas. upenn. edu/˜ vnanda/perseus*, 2012.

[50] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Distributed computation of persistent homology. In *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 31–38. SIAM, 2014.

[51] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *International Congress on Mathematical Software*, pages 167–174. Springer, 2014.

[52] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[53] Partha Niyogi, Stephen Smale, and Shmuel Weinberger. A topological view of unsupervised learning from noisy data. *SIAM Journal on Computing*, 40(3):646–663, 2011.

[54] Gunnar Carlsson, Tigran Ishkhanov, Vin De Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. *International journal of computer vision*, 76(1):1–12, 2008.

[55] Deepti Pachauri, Chris Hinrichs, Moo K Chung, Sterling C Johnson, and Vikas Singh. Topology-based kernels with application to inference problems in alzheimer's disease. *IEEE transactions on medical imaging*, 30(10):1760–1770, 2011.

[56] Paul Bendich, Herbert Edelsbrunner, and Michael Kerber. Computing robustness and persistence for images. *IEEE transactions on visualization and computer graphics*, 16(6):1251–1260, 2010.

[57] Patrizio Frosini and Claudia Landi. Persistent betti numbers for a noise tolerant shape-based approach to image retrieval. *Pattern Recognition Letters*, 34(8):863–872, 2013.

[58] Vin De Silva, Robert Ghrist, and Abubakr Muhammad. Blind swarms for coverage in 2-d. In *Robotics: Science and Systems*, pages 335–342, 2005.

[59] Gurjeet Singh, Facundo Memoli, Tigran Ishkhanov, Guillermo Sapiro, Gunnar Carlsson, and Dario L Ringach. Topological analysis of population activity in visual cortex. *Journal of vision*, 8(8):11–11, 2008.

[60] Barbara Di Fabio and Claudia Landi. A mayer–vietoris formula for persistent homology with an application to shape recognition in the presence of occlusions. *Foundations of Computational Mathematics*, 11(5):499–527, 2011.

[61] Peter M Kasson, Afra Zomorodian, Sanghyun Park, Nina Singhal, Leonidas J Guibas, and Vijay S Pande. Persistent voids: a new structural metric for membrane fusion. *Bioinformatics*, 23(14):1753–1759, 2007.

[62] Marcio Gameiro, Yasuaki Hiraoka, Shunsuke Izumi, Miroslav Kramar, Konstantin Mischaikow, and Vidit Nanda. A topological measurement of protein compressibility. *Japan Journal of Industrial and Applied Mathematics*, 32(1):1–17, 2015.

[63] Yuan Yao, Jian Sun, Xuhui Huang, Gregory R Bowman, Gurjeet Singh, Michael Lesnick, Leonidas J Guibas, Vijay S Pande, and Gunnar Carlsson. Topological methods for exploring low-density states in biomolecular folding pathways. *The Journal of chemical physics*, 130(14):04B614, 2009.

[64] Kelin Xia and Guo-Wei Wei. Persistent homology analysis of protein structure, flexibility, and folding. *International journal for numerical methods in biomedical engineering*, 30(8):814–844, 2014.

[65] Gabriell Máté, Andreas Hofmann, Nicolas Wenzel, and Dieter W Heermann. A topological similarity measure for proteins. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1838(4):1180–1190, 2014.

[66] Kelin Xia and Guo-Wei Wei. Multidimensional persistence in biomolecular data. *Journal of computational chemistry*, 36(20):1502–1520, 2015.

[67] Bao Wang and Guo-Wei Wei. Object-oriented persistent homology. *Journal of computational physics*, 305:276–299, 2016.

[68] Zixuan Cang, Lin Mu, Kedi Wu, Kristopher Opron, Kelin Xia, and Guo-Wei Wei. A topological approach for protein classification. *Molecular Based Mathematical Biology*, 3(1), 2015.

[69] Bao Wang, Chengzhang Wang, and GW Wei. Feature functional theory-solvation predictor (fft-sp) for the blind prediction of solvation free energy. *Journal of Chemical Information and Modeling, submitted*, 2016.

[70] Bao Wang, Zhixiong Zhao, Duc D Nguyen, and Guo-Wei Wei. Feature functional theory–binding predictor (fft–bp) for the blind prediction of binding free energies. *Theoretical Chemistry Accounts*, 136(4):55, 2017.

[71] Dong-Sheng Cao, Yi-Zeng Liang, Jun Yan, Gui-Shan Tan, Qing-Song Xu, and Shao Liu. Pydpi: freely available python package for chemoinformatics, bioinformatics, and chemogenomics studies, 2013.

[72] Bruce N Ames. Identifying environmental chemicals causing mutations and cancer. *Jurimetrics J.*, 20:326, 1979.

[73] Chemical mutagens. `http://medicine.jrank.org/pages/2575/Mutagen-Chemical-Mutagens.html`.

[74] National Research Council et al. *Scientific frontiers in developmental toxicology and risk assessment*. National Academies Press, 2000.

[75] Ann M Richard and ClarLynda R Williams. Distributed structure-searchable toxicity (dsstox) public database network: a proposal. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, 499(1):27–52, 2002.

[76] David J Dix, Keith A Houck, Matthew T Martin, Ann M Richard, R Woodrow Setzer, and Robert J Kavlock. The toxcast program for prioritizing toxicity testing of environmental chemicals. *Toxicological Sciences*, 95(1):5–12, 2006.

[77] Toxicity forecaster (toxcast) data. Data available at `https://www.epa.gov/chemical-research/toxicity-forecaster-toxcasttm-data`.

[78] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1):312–320, 2005.

[79] Katja Hansen, Sebastian Mika, Timon Schroeter, Andreas Sutter, Antonius Ter Laak, Thomas Steger-Hartmann, Nikolaus Heinrich, and Klaus-Robert Muller. Benchmark data set for in silico prediction of ames mutagenicity. *Journal of chemical information and modeling*, 49(9):2077–2081, 2009.

[80] Schrdinger release 2017-2: Ligprep, schrdinger, llc, new york, ny. Software available at `https://www.schrodinger.com/ligprep`, 2017.

[81] David A Pearlman, David A Case, James W Caldwell, Wilson S Ross, Thomas E Cheatham, Steve DeBolt, David Ferguson, George Seibel, and Peter Kollman. Amber, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Computer Physics Communications*, 91(1-3):1–41, 1995.

[82] Junmei Wang, Romain M Wolf, James W Caldwell, Peter A Kollman, and David A Case. Development and testing of a general amber force field. *Journal of computational chemistry*, 25(9):1157–1174, 2004.

[83] Duan Chen, Zhan Chen, Changjun Chen, Weihua Geng, and Guo-Wei Wei. Mibpb: A software package for electrostatic analysis. *Journal of computational chemistry*, 32(4):756–770, 2011.

[84] Beibei Liu, Bao Wang, Rundong Zhao, Yiying Tong, and Guo-Wei Wei. Eses: Software for eulerian solvent excluded surface. *Journal of Computational Chemistry*, 2017.

[85] Pubchem bioassay database; aid=1189. National Center for Biotechnology Information `https://pubchem.ncbi.nlm.nih.gov/bioassay/1189` (accessed June 18, 2017).

[86] Pubchem bioassay database; aid=1194. National Center for Biotechnology Information `https://pubchem.ncbi.nlm.nih.gov/bioassay/1194` (accessed June 18, 2017).

[87] Evan E Bolton, Yanli Wang, Paul A Thiessen, and Stephen H Bryant. Pubchem: integrated platform of small molecules and biological activities. *Annual reports in computational chemistry*, 4:217–241, 2008.

[88] Yanli Wang, Jewen Xiao, Tugba O Suzek, Jian Zhang, Jiyao Wang, and Stephen H Bryant. Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic acids research*, 37(suppl 2):W623–W633, 2009.

[89] Pubchem download service. National Center for Biotechnology Information `https://pubchem.ncbi.nlm.nih.gov/pc_fetch/pc_fetch.cgi`.

[90] VC Arena, NB Sussman, S Mazumdar, S Yu, and OT Macina. The utility of structure–activity relationship (sar) models for prediction and covariate selection in developmental toxicity: comparative analysis of logistic regression and decision tree models. *SAR and QSAR in Environmental Research*, 15(1):1–18, 2004.

[91] Zixuan Cang and Guo-Wei Wei. Analysis and prediction of protein folding energy changes upon mutation by element specific persistent homology. *arXiv preprint arXiv:1703.10966*, 2017.

[92] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[93] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[94] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[95] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[96] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[97] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[98] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[99] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[100] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[101] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[102] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[103] Franois Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[104] Abhik Seal, Anurag Passi, UC Abdul Jaleel, and David J Wild. In-silico predictive mutagenicity model generation using supervised learning approaches. *Journal of cheminformatics*, 4(1):10, 2012.

[105] Lemont B Kier and Lowell H Hall. An electrotopological-state index for atoms in molecules. *Pharmaceutical research*, 7(8):801–807, 1990.

[106] Yin Cao, Bao Wang, Kelin Xia, and Guowei Wei. Finite volume formulation of the mib method for elliptic interface problems. *Journal of Computational and Applied Mathematics*, 321:60–77, 9 2017.