Genome-Wide Protein-Chemical Interaction Prediction

Aaron Smalter Hall

Submitted to the graduate degree program in Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas School of Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:

Dr. Jun Huan: Chairperson

Dr. Gerald Lushington

Dr. Mahesh Visvanathan

Dr. John Karanicolas

Dr. Swapan Chakrabarti

Dr. Brian Potetz

Date Defended: 8/10/2011

The Dissertation Committee for Aaron Smalter Hall certifies
That this is the approved version of the following dissertation:

Genome-Wide Protein-Chemical Interaction Prediction

Committee:

Chairperson

Date Approved

Abstract

The analysis of protein-chemical reactions on a large scale is critical to understanding the complex interrelated mechanisms that govern biological life at the cellular level. Chemical proteomics is a new research area aimed at genome-wide screening of such chemical-protein interactions.

Traditional approaches to such screening involve *in vivo* or *in vitro* experimentation, which while becoming faster with the application of high-throughput screening technologies, remains costly and time-consuming compared to *in silico* methods. Early *in silico* methods are dependant on knowing 3D protein structures (docking) or knowing binding information for many chemicals (ligand-based approaches). Typical machine learning approaches follow a *global* classification approach where a single predictive model is trained for an entire data set, but such an approach is unlikely to generalize well to the protein-chemical interaction space considering its diversity and heterogeneous distribution. In response to the global approach, work on *local models* has recently emerged to improve generalization across the interaction space by training a series of independant models localized to each predict a single interaction.

This work examines current approaches to genome-wide protein-chemical interaction prediction and explores new computational methods based on modifications to the boosting framework for ensemble learning. The methods are described and compared to several competing classification methods. Genome-wide chemical-protein interaction data sets are acquired from publicly available resources, and a series of experimental studies are performed in order to compare the the performance of each method under a variety of conditions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Protein-chemical interaction (PCI) prediction for drug discovery has been studied extensively by researchers for many years [3,29,31]. There exist many relevant databases [51, 55, 66] as well as features [43, 52, 61, 62] and classifiers [3, 30, 47]. Computational methods for PCI are not competitive with physical experiments in regards to prediction accuracy, and traditionally play a supportive role in drug discovery programs. With the availability of large structure and interaction databases, researchers are now aiming their computational tools at screening molecular libraries against proteins from the entire genome. This approach has several advantages, such as increased opportunities for hit/lead discovery, drug repositioning, and insights into the chemical-protein interaction network that controls all cellular processes. Initial research into such genome-wide screening efforts leverage existing technologies such as docking and statistical learning, and the protein-chemical interaction prediction problem has been framed as learning a bipartite graph which can be seen as a special case of link prediction. The current approaches are unsatisfactory because they use models that are either strictly local or strictly global and do not incorporate chemical and protein similarity information to guide model generation. The work described in this manuscript describes a genome-wide protein-chemical interaction prediction method that addresses these shortcomings.

Drug discovery is a well established process consisting of multiple screening stages followed by tuning and optimization of drug characteristics before beginning clinical trials [25,38]. Of the hundreds of thousands or more compounds that are examined in initial screens, only a small number will become drug candidates and even fewer will become drugs. Given the enormous investments required to develope new drugs, research into computer-based *virtual screening* techologies continues to be an area of active interest. The establishment and growth of databases such as PubcChem and DrugBank ensure that researchers have many samples of both structures and interactions with which to test and develop competitive methods. Researchers are also equipped with a wide range of features to represent chemicals, and advances in kernel functions have also produced a number of embeddings corresponding to implcit features useful for learning complex structures such as chemicals [16, 30, 47]. Despite access to this ample data and these powerful computation modelling tools, virtual screening remains a challenging problem, contributing to the drug design process mostly in a support capacity. For example, *docking* is a widely-used method for simulation the physical interactions between a chemical compund and a protein of interest [42]. This method is highly accurate, yet very resource-intensive, with the main drawback that three-dimensional structures must be known or simulated for docking to be applied. For drug-like small chemicals this is not such an issue, but for very large protein structures the case is different, and the limited number of know structures prevents this approach from coming into use for genome-wide screening. Another example is traditional qualitative structure activity prediction (QSAR) [24,35,57], where chemical attributes and substructures are used to infer binding profiles or other responses of interest, such as absorption, dispersion, metabolism, and excretion (ADME) properties. These *ligand-based* approaches are limited in that they focus on only a single protein interaction of interest, and cannot generalize when only a few ligands are known to bind to a particular protein, again making such approaches unsuitable for genome-wide screening.

The difficulty of generating comprehensive activity models for many thousands of compounds owes much to the complexity of chemicals as well as the systems they interact with. Nevertheless, the rise of informatics methods across biological disciplines, along with the continued bloom in rich sample and features sets has continued to encourage research in this area. Current work can be divided into two categories: those that use a single global model to predict all interactions, and those that use a series of individual, interaction specific models. *Global prediction* [22, 30, 47] utilizes samples that represent protein-chemical interaction pairs. All known interacting and non-interacting pairs are used to train a machine learning model to discriminate between the two classes. This approach is straightforward for screening of interactions between chemical/proteins from one or two related chemotypes/families. When the diversity of molecules and proteins is greater however, such as in the case of genome-wide screening, this approach becomes less attractive since the probability is low that a single model will generalize between interactions drawn from many different distributions.

Recently, *local prediction* approaches have emerged in response to global models [5]. Local model approaches generate a separate model to predict every protein-chemical interaction independently. The interaction specific model could take a variety of forms such as a docking simulation, or a machine learner. The local model approach addresses the generalization and accuracy issues mentioned regarding global models above, but are still not satisfactory. While not limited by a single model for all interactions, local approaches fail to leverage the similarities between proteins and chemicals that are known to exist, and also fail to share predictions between models. Additionally, local model approaches require a number of models equal to the number of samples, which is not ideal when screening a large volume of interactions. A new framework is needed that avoids the use of a global model and can automatically partition the sample space into well-classified regions.

The construction of a strong learner through the composition of many weak learners is a well known strategy in ensemble learning and is has been studied by many researchers. There are several challenges evident in addressing this problem and the particular limitations of current work. First of all, considering the complexity of the protein-chemical interaction space, feature dimensionality in the hundreds and thousands is not unexpected. Such high dimensionality can pose trouble for even the best classifiers, and hence developing a method that supports kernel similarity is key in order to avoid such high dimensional contexts. Second, the interaction sample space must be covered evenly, but also in such a way that the assembled models highlight difficult areas. Finally, class distributions and boundary complexity may change drastically throughout the sample space.

There are many possible approaches to such issues. One way to address them is through multi-task learning or transfer learning. In these approaches, several different but related data sets are required, making the segmentation of the sample space a critical component. There is some natural division in the chemical-protein interaction space by protein families or chemical chemotypes, but there is no clear division and either expert or automated partition is required. Another approach is to use a classifier such as k-Nearest Neighbor (kNN), which assumes from the start that similar samples should be classified similarily. The issue with kNN is that the prediction can become confused at class boundaries or overlapping areas, and performance depends on the parameter k; and like other global models, the optimal parameterization in one region of protein-chemical interaction space may not be optimal for other regions. Local regression is a statistical technique that also assumes model locality, but also suffers from a need for either uniform segmentation (which may not be optimal for model coverage) or expert segmentation. Additionally, local regression lacks the ability to share information between models for improved generalization.

Among the many ensemble learning methods available for adaptation to address the challenges described above by, Boosting [27, 39] in particular has many similarities to the local model approach. Boosting constructs a set of individual models to make predictions targeting a specific subset of samples, and the predictions are combined into an overall model for all samples. The boosting algorithm is agnostic to model used, and thus models utilizing kernels or requirements are readily applied. Model coverage and sample space segmentation is guided by explicitly targeting samples that are currently misclassified. The base learners used are adapted to specific regions of the sample space, but are still combined into a weighted model for improved generalization. Boosting differs from local models in that predictions from each boosting model are used for classifying every unknown interaction, while in local models the predictions from each model are used for only a specific interaction. In boosting, variation between specific models is controlled by adjusting a weight distribution over the samples such that additional models focus on samples that are poorly classified by previous models. Thus, while standard boosting draws on many models, it does not leverage any information regarding similarities between samples to train specialized models.

This work describes several methods based on modified boosting algorithms that build a moderate number of models that are automatically specialized for subsets of the overall interaction space. This approach is evaluated using a series of chemical activity prediction experiments comparing it to competing methods. Chemical structure data, protein sequence data, as well as chemical-protein interaction data sets are obtained from publicly available repositories. Chemical features are generated using publicly available software. Feature selection filter features has been examined as well. Studies have been performed to compare predictive performance across parameter choices and data sets. Experiment execution and competing method implementations are handled via the Weka software package implemented in java. Results from prediction experiments are evaluated using cross-validation and compared according to accuracy an darea under ROC curve.

The remainder of this text discusses the research details as follows. First a background section will describe the drug discovery process with discussion of target and ligand-based approaches along with wide-scale approaches using large sets of targets as well as ligands. This section also duscisses background on the state of virtual screening as well as data fusion for chemical-protein interaction prediction. Next, a series of chapters describes work on protein-chemical interaction prediction. Finally, the problem of genome-wide protein-chemical interaction prediction is examined in greater detail and two approaches for ensemble methods tailored for this problem are presented with technical details. Finally, the experimental studies comprehensivly evaluating the new methods are presented and the results are discussed.

Chapter 2

Background and Related Work

While the research described here focuses on finding and testing computational approches for problems described in general mathematical terms, the solutions are motived by issues in the fields of biology and pharmaceutical chemistry. Understanding the purpose and merit of this research requires understanding the drug discovery process from which it originated. This section will outline this process, as well as the role of computer-based techniques used to support and guide it.

2.1 Chemical Structures and Graph Representations

Chemical compounds are organic molecules that are easily modeled by a graph representation. In this approach, *nodes* in a graph model *atoms* in a chemical structure and *edges* in the graph to model chemical *bonds* in the chemical structure. In this representation, nodes are labeled with the atom element type, and edges are labeled with the bond type (single, double, and aromatic bond). The edges in the graph are undirected, since there is no directionality associated with chemical bonds. Figure 2.1 shows an example chemical structure, where unlabeled

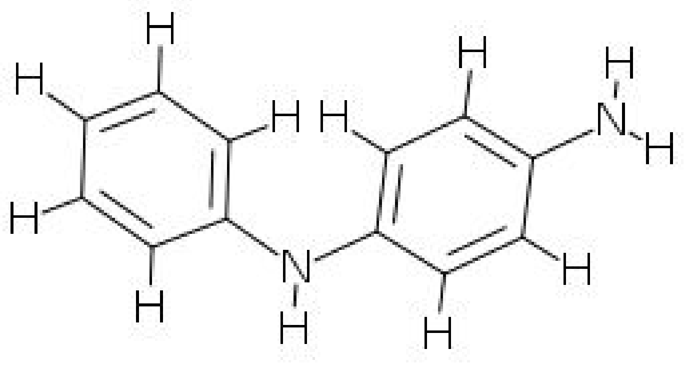

Figure 2.1. An example chemical structure.

vertices are assumed to be carbon (C).

Figure 2.2 shows three sample chemical structures on the left, and their graph representation on the right.

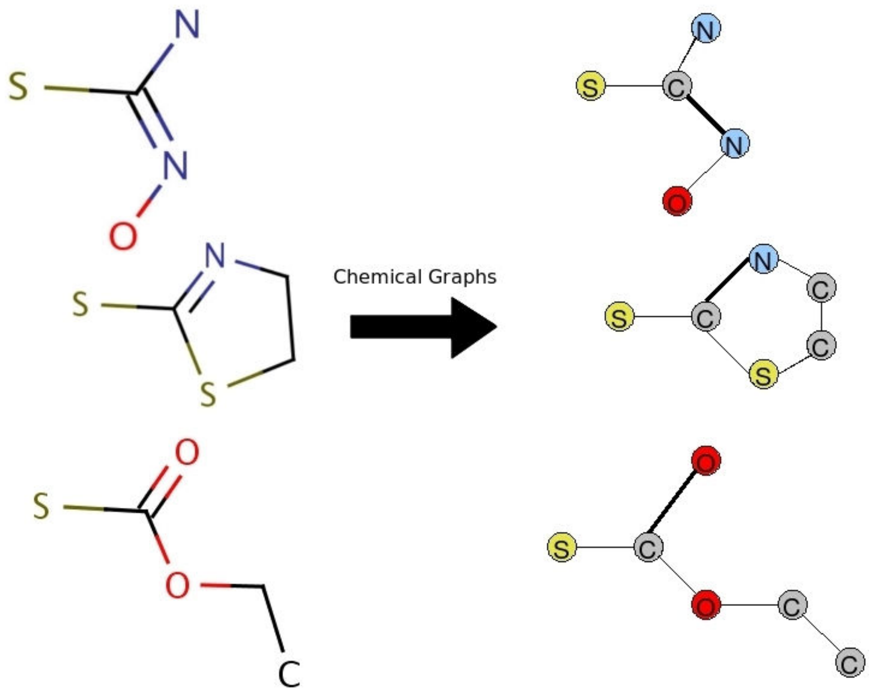

Figure 2.2. Graph representations of chemicals.

Let us make some formal definitions regarding the graph representations for these chemical structures.

Definition 2.1.1 *A* ***labeled graph*** *G is a quadruple $G = (V, E, \Sigma, \lambda)$ where V is a set of vertices or nodes and $E \subseteq V \times V$ is a set of undirected edges. Σ is a set of (disjoint) vertex and edge labels, and λ: $V \cup E \rightarrow \Sigma$ is a function that assigns labels to vertices and edges. Assume that a total ordering is defined on the labels in Σ.*

Definition 2.1.2 *A graph $G' = (V', E', \Sigma', \lambda')$ is* ***subgraph isomorphic*** *to $G = (V, E, \Sigma, \lambda)$, denoted by $G' \subseteq G$, if there exists a 1-1 mapping $f : V' \rightarrow V$ such that*

- $\forall v \in V', \lambda'(v) = \lambda(f(v))$
- $\forall (u, v) \in E', (f(u), f(v)) \in E$, *and*
- $\forall (u, v) \in E', \lambda'(u, v) = \lambda(f(u), f(v))$

.

The function f is a *subgraph isomorphism* from graph G' to graph G. It is said G' *occurs* in G if $G' \subseteq G$. Given a subgraph isomorphism f, the image of the domain V' ($f(V')$) is an *embedding* of G' in G.

Example 2.1.1 *Figure 2.3 shows a set of three labeled graphs. The mapping (isomorphism) $q_1 \rightarrow p_3$, $q_2 \rightarrow p_1$, and $q_3 \rightarrow p_2$ demonstrates that graph Q is subgraph isomorphic to P and hence Q* occurs in *P. Set $\{p_1, p_2, p_3\}$ is an embedding of Q in P. Similarly, graph S occurs in graph P but not Q.*

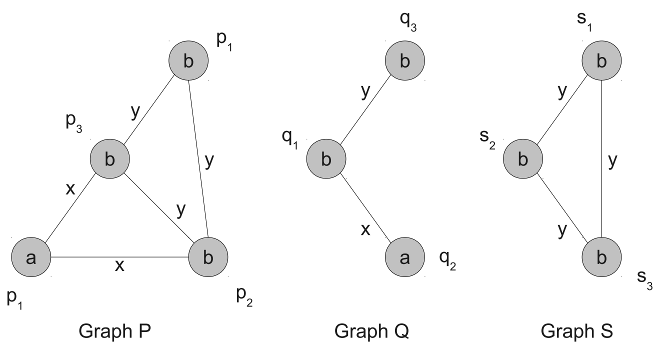

Figure 2.3. A Database of three labeled graphs.

2.2 Data Formats for Cheminformatics

Many data formats exist for the representation of chemical structures. In general these representation formats vary in terms of descriptive richness. Chemical structures naturally occur as a three-dimensional configuration of atoms connected by bonds, but are often written as two-dimensional connectivity diagrams, or even one dimension strings such as a chemical formula. This subsection reviews common chemical representation formats with attention to the level of information provided.

The MDL molfile format is a widely used representation (named after the now defunct company MDL Information Systems that developed the format) that can encode structural information about a chemical compound [10]. This representation forms the basis for the better known Structured Data Format (SDF) file which contains MDL definitions for many appended compounds, along with additional name/value pairs of arbitrary tagged information (e.g., physicochemical properties or metadata) available on a per-compound basis. Along with a chemical identifier and header information, the MDL format lists a set of atoms and bonds occurring in a particular chemical, with two or three dimensional positions attached to the atoms, and connectivity information attached to the bonds. The

SDF format is very flexible and most software systems that utilize chemical data will accept this format for import/export.

The Simplified Molecular Line Input Specifications (SMILES) [65] format is a one-dimensional string representation, constructed in such a way that the two-dimensional structure of a chemical can be recovered from the one-dimensional string. This is possible because the atom letters in a SMILES string are ordered according to a depth-first traversal of the chemical connectivity graph. Additionally, SMILES strings can encode a number of structural and physical properties of chemicals such as aromaticity, branching, stereochemistry and isotopes.

The Smiles Arbitrary Target Specification) SMARTS [11] is a string-based representation of chemicals that extends the SMILES format. The focus of SMARTS is on substructure specification a modifications to the representation language that allow SMARTS strings to function not just as molecular encodings, but also as queries that can be performed against chemical databases. These queries are not performed on the string representations directly, but rather the SMARTS/SMILES strings are converted to connectivity graphs and subgraph isomorphism is used as criteria for matching.

The IUPAC International Chemical Identifier (InChI) [48] is a one-dimensional string representation of chemicals, developed by the IUPAC and NIST as a world-wide, human-readable standard for molecular representation. Unique InChI identifiers are generated from chemical structures by first normalizing the structure to eliminate redundant information, then unique indices are assigned to atoms and bonds, and finally converted into a serial string representation. The final string representation contains several layers of information such as chemical formula and connectivity, positive/negative atomic charge, stereochemistry, and isotopes.

The Chemical Markup Language (CML) [49] is a XML-based schema used to describe a variety of chemical concepts such as chemicals, reactions, crystal structures, spectra, and computational chemistry information. CML is noteworthy in that is was the first attempt by scientists to create a common exchange format for an entire discipline. The Java Universal Molecular Browser for Objects (JUMBO) tool is a freely available program for reading and writing CML files, as well as converting other chemistry file formats into CML. While in development for over a decade, CML has become part of popular ChemAxon and CambridgeSoft cheminformatics software, and hence enjoys continued support.

2.3 Protein Databases

The number and size of databases storing protein information have both grown significantly throughout the last decade. Many different kinds of protein databases have emerged, each targetting different research purposes and/or organisms. This section outlines several databases encompassing protein knowledge.

The Protein Data Bank (PDB) [4] is a large, ubiquitous resource similar to PubChem in the scale of data and amount of use. PDB stores protein sequences and also focuses on storing the 3D structure of various proteins, often obtaine dthrough x-ray crystallography. A unique feature of PDB is that many scientific organizations around the world mandate the submission of published experimental results to PDB.

The Biomolecular Interaction Network Database (BIND) BIND [21] stores descriptions of interactions, molecular complexes and pathways. BIND stores only biological information about proteins, genes and their interactions, and is structured around so called interaction pairs i.e., two interacting molecules contain information about the molecules and the interaction itself. BIND also provides the protein interaction datasets in XML format based on the PSI MI standard.

The Munich Information center for Protein Sequences (MIPS) [53] provides whole genome protein sequence-based information for various model organisms, integrating a number of databases (each devoted to a specific organism or contextual focus) including: yeast, Neurospora crassa, human, mammalian, Arabidopsis thaliana, and rice. While distinct, these databases are all comprehensive organism genome resources. MIPS contains both automatic and manually curated records, with systematic classification schemes and functional protein annotations.

The Database of Interacting Proteins (DIP) [68] stores protein-protein interactions, including physical associations and chemical reactions and chemical states of those proteins. DIP represents interactions via a binary interaction scheme, depicted schematically with a graph abstraction and a visual navigation tool. DIP enforces a formal grammar for data specification, but allows description of the interacting proteins, experimental methods underlying the interaction determination, quantifies the dissociation constant for physical associations, reports the amino acid residue ranges for the interaction site and provides references for the interaction.

The Molecular Interaction Network Database (MINT) [7] is a molecular interactions database assembled from the literature and manual input. In addition to a simple relational schema for representing binary relations, MINT records information about protein post-translational modifications, experimental metadata, cellular location, pathway participation and known complexes. MINT supports experimental verification of protein-protein interactions, and provides detailed interaction data including kinetic and binding constants and associative domain annotation. MINT uses an automated software system to scan abstracts and suggest literature studies to be manually curated by domain experts. As of March 2011, MINT reported 90290 interactions among 31870 proteins.

2.4 Chemical Databases

Chemical databases have seen growth similar to that of protein databases. With the increasing amount of high-throughput chemical biology screening results generated by centers and labs around the world, a number of databases have arisen to collect, categorize, and manage these results. Typically these databases consist of two components, 1) a large library of chemical structures, and 2) a set of assay results linking selections of chemicals compounds to observed bioactivities.

PubChem (http://pubchem.ncbi.nlm.nih.gov) [55] is a large resource of chemical structure and bioactivity information organized into three affiliated databases: Compounds, Substances, and BioAssays. PubChem is maintained by the National Center for Biotechnology Information (NCBI), part of the NIH Molecular Libraries Program, and has become a ubiquitous resource for research in cheminformatics and related fields. As of January 2011, the Substances section of PubChem contains over 75 million records describing chemical mixtures, extracts, and complexes, while the Compounds section contains over 31 million records of well-characterized chemicals. The compound and substance records describe not only the two-dimensional structures of chemicals, but also compound metadata, chemical properties and links to bioactivity results and other related compounds. These structures and other information can be obtained from PubChem in a variety of common formats such as XML or SDF. The BioAssays section of PubChem contains results from a number of high-throughput screening programs, cataloging millions of bioactivity endpoints such as toxicity or target inhibition. BioAssay records contain descriptions of experimental protocols and other relevant information along with the specific activity results. PubChem databases are publicly accessible through a web-based interface that supports a fairly broad range of query and analysis options, and the raw data and structures can be downloaded via FTP. PubChem allows voluntary deposition of new records by researchers, and while depositions are screened by curators, PubChem data is not exhaustively validated.

ChemBank (http://chembank.broad.harvard.edu/) [36] is designed as not only a database, but also an environment for the analysis of small molecules and their bioactivity. It was created by the Chemical Biology Program of the Broad Institute at MIT and Harvard, with funding from the United States National Cancer Institute. ChemBank is an intensively curated resource with a focus on high-throughput screening data. As of August 2007 it contained information on more than 1.2 million unique chemical structures with from at least 2500 assays. In addition to structure and activity data, ChemBank also calculates more than 300 molecular descriptors and organizes bioassays hierarchically using metadata. It also offers a suite of analysis tools such as searching by similarity, descriptors, or substructures, as well as visualization of screening results and chemical-genetic profiles.

Chemical Entities of Biological Interest (ChEBI) [32], is a database of molecular entities which can refer to chemical structures or mixtures containing any number of components. Whole molecules as well as molecule fragments, complexes, or even individual atoms are indexed. The criterion for inclusion in the database is that all of the entities are natural products of organism biology, or can interfere in biological processes (such as toxins or synthetic drugs). Biomolecules that result from genetic processes (nucleic acids, proteins, etc.) are excluded from this database. ChEBI uses standard representations and formats from the International Union of Pure and Applied Chemistry (IUPAC) and Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB).

ChemDB [8] is a publicly available database of small molecules. The distinguishing feature of this database is that is compiled from the compound collections maintained by over one hundred industry and public laboratories. Additionally, ChemDB includes computationally derived predictions and annotations such as solubility and three-dimensional structure. Currently the ChemDB database contains more than 4.1 million commercially available chemical compounds, with over 8.2 million isomers. Like other drug discovery databases, an important component of ChemDB is the analysis tools available to visualize and search chemicals and chemical reactions, among other uses.

2.5 Cheminformatics Databases

There are many publicly available data sources from which to retrive a) chemical structures, b) protein sequences, c) protein-chemical interactions. While many of the previously discussed databases can certainly be used in drug discovery research, there exist several databases specifically designed with drug discovery in mind. The chemicals stored in these databases are often filtered to contain molecules that are already FDA approved or drug-like to some extent. Additionally, these databases provide detailed information regarding drug targets as well as computational tools to guide the drug development process. A few such databases are described here.

DrugBank [66] is a drug discovery database combining detailed target information with pharmaceutically relevant drug information. With regards to drug targets, DrugBank stores raw sequence data, as well as conformational structures and links to associated pathways and splice variants. The drug information records contain a number of properties relevant to drug discovery such as mode of action and pharmacokinetic profile. As of March 2011, Drugbank stores over 6800 records for drugs, with about 1400 of those drugs considered FDA-approved and about 5200 considered experimental. It also stores over 4400 protein sequences with links to interacting drugs. DrugBank can be browsed or searched using textual or similarity based queries. Records are publicly available for download. DrugBank was developed at the University of Alberta.

BioDrugScreen [40] is a combined web-based drug discovery database and analysis server. The database portion of the service consists of the Docked Proteome Interaction Network (DOPIN). The distinguishing feature of DOPIN is that, in addition to storing chemical-protein and protein-protein interactions, it reports results from pre-computed docking simulations. These docking scores can be combined with customized scoring functions in order to rank chemical compounds and their possible targets. The scoring function can be arbitrarily defined to take into account structural information or other descriptors extracted from existing databases. BioDrugScreen also offers tools to evaluate the scoring functions and resulting predictions, and can also perform on demand docking of user-uploaded molecules against preprocessed targets from the PDB database.

The Therapeutic Target Database (TTD) [67] is a drug-target database focused on therapeutic applications. It indexes protein and nucleic acid targets, annotating them with information about pathway participation, disease association, corresponding drugs and other relevant properties. These records are cross-referenced with other databases noting sequences, structures, binding affinities, and clinical results, among other details. TTD is created by the Bioinformatics and Drug Design Group at the National University of Singapore and currently contains over 1,900 known, clinical, and research targets and over 5,000 approved, clinical, and experimental drugs. This set of targets and drugs covers 61 classes of proteins and 140 classes of drugs. TTD allows searching via textual queries on multiple fields.

2.6 Biological Descriptors

No matter the chemical-protein interaction approach used, chemical and protein *descriptors* or features must first be extracted before a predictive model can be learned. The features extracted for both protein and chemical domains can either be represented explicitly or implicitly. In the case of explicit features, examples include features that represent unstructured numeric properties like mass or charge [**?**], or features that represent sequence and graph structured data such as paths [6, 34], cycles [61], trees [46], and general subgraphs [14]. Fragment-based features are popular for learning on chemicals structured as graphs [28], however physicochemical properties are also well studied for interaction prediction [**?**]. Protein sequences are not as complex as chemicals and hence k-mer [37] sequence features are popular as well as protein annotations such as from Gene Ontology. Implicit features computed using a kernel function can take advantage of any of the previously mentioned features, decomposing complex structured objects into simpler fragments that are used in the implcit representations. Using this approach, graph kernels based on several types of fragments have been studied [20,41,64].

2.6.1 Numeric Property Descriptors

Perhaps the most common characterization of biological entities is a vector of numeric property values. In this approach, numeric properties describing a biological entity (such as a chemical or protein) are collected and used to project entities into a Euclidean space with dimension equal to the number of properties. Each entity is described by the same set of properties. There are many different kinds of numeric properties that can be calculated for chemicals or proteins, and each property describes the entitiy as a whole. Simple examples of such properties include total molecule mass, charge, solubility, and other physicochemical properties. Tools for generating numeric property descriptors for chemicals and proteins are capable of calculating hundreds of even thousands of such properties.

2.6.2 Structure Fragment Descriptors

While using numeric propery descriptors is widely accepted, the observation that chemical/protein properties are directly dependant on their corresponding molecular structure has lead to the use of descriptors that denote presence or absence of structural fragments in a chemical or protein. Since the space of possible chemicals structures as well as protein sequences is infinite in theory, indexing all structure fragment descriptors is not possible. In response to this, descriptors for fragments up to a specified size may be generated, or else an interesting set of structure patterns may be mined directly from a set of biological structures.

2.6.3 Kernels for Implicit Descriptors

Given the complexity of chemical and protein structures, it can be quite difficult to capture their essential properties using numeric descriptors. Further, using structure fragments to index chemicals and proteins is problematic in that it can be difficult to know *a priori* what the best set of fragment descriptors is. Using a large number (hundreds or thousands) of either kind of descriptor can lead to long model training times as well as poor generalization ability. In response to these difficulties, many researchers use kernel functions to compute structural similarity between two chemicals or proteins. Kernel functions are attractive in their ability to enhance model training and generalization, but are limited in that researchers do not have direct access to relevant descriptors and hence kernel-based models lack explanatory capability.

2.7 Drug Discovery Process

The creation of new drugs to treat both new and old diseases a lengthy, risky, and expensive endeavor. The complexity of this task combined with possible health risks of new drugs has shaped modern drug design into a rigorous and well-established practice. There are many different approaches to drug development, but traditional non-computational methods typically fall into two categories: those using drug ligands with known pharmcological activity in order to design a novel drug, and those that use the 3D structure of a chemical and its target in order to select drugs with high binding affinity. These techniques are referred to as liband-based and structure-based approaches, respectively.

Many investigators frame drug discovery using *rational drug design* whereby they develope a small molecule that shows the desired activation/inhibition of a biomolecular target with known therapeutic activity. This drug discovery process consists of several stages in series over which a set of possible drugs is narrowed and modified until finally a candidate drug is found, or else all possible compounds are exhausted. The stages of this process, are roughly: target identification and validation, in which a biological target is isolated and shown to be effective for disease treatment; screening and lead identification, where a large number of compounds are screened for activity against the chosen target and promising ”leads” with desired activity are identified; and lead optimization and development, where lead compounds are modified and tuned into non-toxic, druggable compounds with the desired target selectivity. Of course these stages may have multiple steps and are not entirely distinct, but these are the broad strokes. Before approval for sale, candidate drugs must pass clinical trials, which are themselves a lenghty and difficult process, though beyond the scope of discussion here.

2.7.1 Validating Candidate Hits

The drug discovery process typically begins with a set of candidate chemicals, or hits. A hit is a compound that has been shown to have some activity against a protein or condition of interest. Hits are generated by preliminary experiments, typically a biological screening experiment but virtual screening may be employed as well. Screens are typically performed on large libraries of chemicall-diverse compounds, with hits generally falling into one or more chemically related clusters of compounds. These candidate hit compounds must then be experimentally confirmed with more detailed experimentation. Compounds may be re-tested using the same screening procedure, or a subjected to secondary screening procedures. One important component of hit confirmation is generation of a dose-response curve where the activity of a compound is quantified relative to it's concentration. Hit candidates may also be evaluated according to practical qualities like ease of synthesis or other development costs. The most promising hits and their chemically related clusters are selected for further evaluation.

2.7.2 Analysis of Hit Clusters

After validation, hit compounds and clusters are analyzed not only according to their activity, but also in terms of their potential to be used as a drug. Compounds must be selected according to several important criteria. Cellular toxicity is one important property that must be minimized. Quantification of ADME characteristics is another goal of analysis: Absorption (how well can an administered drug be absorbed by the body), Dispersion (how is a drug distributed throughout the body and to the target of interest), Metabolism (how quickly is the drug broken down and what are the relevant properties of the metabolites), and Excretion (how quickly and through what route is the drug eliminated from the body). Other properties such as solubility, cell membrane permeability, and interaction selectivity are considered as well.

2.7.3 Optimization of Lead Candidates

Once hit compounds are analyzed to ensure effective action as a drug, the most promising compounds are further optimized by synthesizing and testing analogs or structural variations with different functional groups. Often this process is informed through the analysis of structure-activity relationships in hit compounds exhibiting desireable drug-like properties. A hit compound that has been successfully optimizaed to exhibit all desired drug properties is known as a lead.

Lead compounds may be subjected to further analysis and if promising may be developed for clinical trials.

2.8 Computer Aided Drug Discovery

Before the development and wide-scale adoption of computers and informatics technologies, drug design experiments were carried out strictly in wet labs, with little automation. When technologies began to mature, however, researchers quickly developed computer-based tools to aid in the drug design process. This research has resulted in two distinct approaches (among others): simulation and machine learning. Simulation or *docking* involves detailed modeling of the physical laws and molecular dynamics governing the binding of a specific drug ligand to a target molecule. Using machine learning, on the other hand, results in a rapid assessment of target activity for a large number of chemical compounds. These twos methods address different needs and find utility at different points of the drug discovery process, with machine learning being used primarily to guide screening for hit identification at the beginning, while docking can be used later on to gauge the precise binding activity for a potential drug as well as predict affects of specific adjustments to a molecule.

2.8.1 Docking

Docking refers to a molecular simulation in which the optimal orientations of two molecules are found such that a binding event occurs to form a stable molecular complex. The problem can be stated as a search problem, where the search space is the set of all possible orientations of the two molecules. Of course this space is much too large to search exhaustively and hence heuristics are necessary.

A common approach is to use molecular dynamics to produce orientations resulting from known physical laws and processes. Such approaches, know as *ab initio* modeling, can produce high quality binding assessments, but accurate physical simulation requires a large computational investment.

2.8.2 Machine Learning

Machine learning is a broad field of research concerned with building models that make predictions. The application of such models to drug discovery and other biomedical domains has received much attention by researchers. This approach uses sets of descriptors of *features* to describe a chemical compound, and then builds a model to predict some target activity.

Chemical features are numeric representations of a non-numeric object (such as a molecule). Typically, a set of n molecules is represented with m features using an n by m matrix. Features may be dscrete or continuous. The *target activity* of a chemical compound refers to the ability of that compound to physically bind (reversibly or irreversibly) to the target molecule. Like features, there are two categories of target activity values: continuous (e.g., weak/medium/strong binding affinities) and discrete target properties (e.g. active vs. inactive compounds).

The relationship between a chemical compound and its target activity is typically investigated through a quantitative structure-activity relationship ($QSAR$). Abstractly, any QSAR method may be generally defined as a function that maps a chemical space to an activity space in the form of

$$P = \hat{k}(D) \tag{2.1}$$

where D is a chemical structure, P is a property, and the function $\hat{k}$ is an estimated mapping from a chemical space to an activity space.

Different QSAR methodologies can be understood in terms of the types of target property values (continuous or discrete), types of features, and algorithms that map descriptors to target properties.

2.9 Drug Discovery Approaches

The drug discovery process is not identical in every case. Typically the methods used to progress through the stages of drug discovery are tailored to the particular biological considerations at hand and hence some variation is necessary. This section outlines some genreal approaches to drug discovery from different standpoints. Ligand-based, target-based, and genome-wide protein-chemical interaction prediction are described.

2.9.1 Ligand-based Prediction

Ligand-based approaches are useful when the activity of some new set of compounds must be deduced using only their similarity to a set of compounds with known activity. Ligand-based prediction is useful when an interesting chemical ligand has been identified, often because of observed binding activity to a protein target, and researchers wish to understand what other protein targets the chemical might bind to. This is often the case when drug design researchers have found a promising drug candidate after a round of target-based screening, and wish to understand the selectivity profile of the candidate. Knowing the selectivity profile is important for knowing what side-effects a drug may have in a patient, and hence candidate drugs must be very selective in the targets they bind to. Many such studies have been performed [25, 38].

Analysis of chemicals and activities toward protein targets other responses first requries a suitable vocabulary for describing those chemicals. Typically this is accomplished by enumerating sets of numeric attributes of chemicals, these attributes are known as *descriptors* or *features*. Many different types of chemical descriptors exists, from simple chemical weight, to binary values indicating presence of specific functional groups. The usage of standard sets of features ensures that analysis of protein-chemical relationships is consistent across experiments and researchers. The representation of chemicals using features also allows the application of generalized statistical and machine learning methods to assist in chemical analysis.

Quantitative structure-activity analysis (QSAR) is a widely used approach in ligand based protein-chemical interaction prediction that leverages feature-based representations of chemicals. Many statistical modeling techniques have been utilized to build QSAR models, such as generalized linear models [63,70], random forest [56, 60], Support Vector Machines [9, 12], and ensemble methods [1, 69] among others. In principle, all the aforementioned techniques (and many other regression and classification methods) can be utilized to construct models for predicting the interactions of proteins and chemicals if we have a sufficiently large training data set with high quality features.

2.9.2 Target-based Prediction

Target-based prediction is useful when a specific protein or family of proteins has been identified as a candidate target, often for treatment of some disease or condition, and researchers wish to identify all candidate chemical ligands that bind to the target in order to treat the disease or condition. Many targets have been studied in target-based screening studies such as G-protein coupled receptors GPCRs, kinase-inhibitors, ion channels, and others [5, 29, 71]. Target-based screening is dominant is drug design methodologies in order to identify possible candidate drugs to treat a disease. Once proteins have been identified that are relevant to a disease condition, chemical libraries are screened in order to determine the most promising molecules for treatment.

2.9.3 Genome-wide Prediction

Ligand and target-based approaches typically focus on the activity of a single protein or chemical against a variety of other proteins or chemicals. This approach is useful when investigators are interested in a specific protein/chemical of value. The space of possible protein-chemical interactions is much larger than the numnber of known interactions, however, and recently the availability of large amounts of data characterizing numerous chemicals and proteins have made different screening approaches possible. Now the opportunity exists to employ methods for interaction prediction on a much larger scale, where the goal is to predict the activity of many protein targets against many possible chemical ligands. This many-to-many approach is able to analyze a much broader range of interactions and has the potential to reveal many insights into the nature of the genome and interactome. Thus such genome-wide interaction prediction has received much interest.

This section summarizes recent work on genome-wide chemical-protein interaction, beginning with initial work on global mapping of the interaction space [54] and efforts to apply existing docking [18] and support vector machine [50] technologies. More sophisticated kernel methods were later applied [29, 50], and the interaction prediction problem was also formalized as link prediction on a bipartite graph [5, 71]. Recent work has focused on visualization of different protein/chemical data sets in order to understand their overlap, but has not discounted the predictive abilities of such models [59].

Research by Paolini et al. [54] focused on making use of the large-volume of available protein-chemical interaction data in order to construct a large drug-target matrix that was as accurate as possible. Their work was driven by a need to consolidate and manage a large amount of data: 4.8 million chemical structures (275,000 biologically active), and over 600,000 chemical activity profiles from Pfizer combined with commercial and other screening data; all totalling 25 years worth of published medicinal chemistry data. This information is used to construct a polypharmacology network, where proteins are represented as vertices in a graph, and edges connect protein vertices if they both bind with one or more of the same molecules. This network reveals many clusters of proteins interacting with similar chemicals. Besides discovering many interesting properties of this network, the researchers also constructed predictive models for 698 protein targets using Laplacian Bayesian classification. The models were trained for each target using known positive and negative chemical interations and tested on the unknown interactions. Molecules are represented in the model using fingerprint features that index chemicals by the presence linear atom sequences. Next, linear discriminant analysis was used to predict the activity of specific gene families against a set of compounds. This analysis uses several molecular properties such as molecular weight, number of hydrogen donor/acceptor bonds, and molecular solubility.

Among these earlier efforts in genome wide in silico interaction prediction research, Gao et al. [18] developed a natural extension of accepted docking technology to the genome wide interaction prediction problem. Assembling a data set of 1,055 small molecules drugs and 1,548 binding pockets from 78 unique human targets, they applied a series of forward and inverse docking experiments. Forward docking consists of molecular simulations of binding energies of many chemicals to a single protein. Inverse docking consists of the opposite, where many simulations of a single chemical are run for docking many different proteins. By performing both forward and inverse docking for the set of drugs and targets in a library, the entire network of drug-target interactions can be built, and potential new targets of know drugs can be discovered, thus potentially minimizing the costly drug approval process.

Nagamine and Sakakibara [50], following research into genome wide docking simulations, concluding that such methods are not entirely feasible due to the limited availability of 3D structural information, although more and more such information has become available through the years. Approaching the problem from a machine learning viewpoint, they frame protein-chemical interaction as a general prediction problem and apply Support Vector Machine technology. Each protein-chemical interaction becomes a sample in their model, and a feature vector is built by concatenating both chemical and protein features. These include chemical path frequencies, mass spectrum fragment and gap intensities, and amino acid sequence signature clusters. Large scale experiments were conducted on DrugBank data using 519 approved drugs and 291 associated targets, with 980 known interacting pairs. The method is further validated using experiments with the single drug MDMA against 13,487 human proteins, producing several biologically correct predictions.

Jacob et al. [29] refined the machine learning approaches with the incorporation of many kernel functions to investigate genome wide interaction prediction for investigation of G-protein coupled receptors, which is a protein superfamily containing a large number of known therapeutic targets. Their goal was to discover novel chemical ligands for known and unknown GPCR targets, and followed previous machine learning approaches by utilizing a Support Vector Machine classifier. Their contribution, however, to this approach is the thorough investigation of kernel functions to embed protein-chemical interactions in an implicity defined space that avoids high dimensional representations. They investigate several kernel functions for both chemicals and proteins, and combine the two using a product kernel. Jacob and Vert later extend the work on kernel functions for genome-wide protein-chemical interaction prediction from the GPCR protein superfamily to both enzymes and ion channel proteins.

Yamanishi et al. [71] also focus on prediction of small molecule interactions against several families of human proteins: enzymes, ion channels, GPCRs, and nuclear receptors. The contribution of this work is in the formalization of the protein-chemical interaction prediction as a supervised learning problem on a bipartite graph where each chemical is represented as a vertex in one partition, and every protein is represented as a vertex in the other partition. Kernel functions are used to embed chemicals and proteins into separate chemical/genomic spaces, and then combined so that both chemicals and proteins inhabit a shared pharmacological space. In this pharmacological space, interacting chemicals and proteins are near each other and thus interactions between new proteins and chemicals can be predicted by examining their distance in the pharmacological space.

Bleakley and Yamanishi [5] build on the concept of using a bipartite graph model to predict genome-wide protein-chemical interactions. The contribution of this work is in the notion of a local models where many interaction specific models are built, instead of the typical paradigm where a single model is used to predict all interactions. The bipartite local model approach predicts an interaction between a protein and chemical by first building a model for the chemical against all known proteins, and then for the protein against all known chemicals, thus generating two predictions for the interaction. The process is repeated for each interaction. A support vector machine classifier is used for each of the individual models. Like earlier work, the approach is tested and validated on four protein family datasets, enzymes, ion channels, GPCRs, and nuclear receptors. Several literature-confirmed predictions are reported.

Following the work on genome wide protein chemical interaction prediction, Strombergsson and Kleywegt [59] shift the focus somewhat from prediction to visualization of protein-chemical spaces, although they also comment on how their approach can be used to identify interaction complexes in the protein-chemical space. They compared two datasets, one from the DrugBank database and the second from the PDB database in order to visualize the differences between the distribution of protetin-chemical interactions in each. Proteins are represented using sequence features such as composition, order, and physicochemical properties; chemicals are represented with a wide variety of features such as molecular properties, fingerprints, functional group presence, etc. Principle component analysis is used to visualize the overlap of the two datasets in 3 dimensions using protein features, chemical features, and the concatentation of both feature sets. Overlap is quantified by measuring the percentage of nearest neighbors of each point that are from each database. Nearest neighbor analysis is also used to make predictions about new protein chemical interactions, by classifying an interaction in the protein-chemical space according to the classes of the nearest interactions.

2.10 Ensemble Learning for Genome-wide Interaction Prediction

Some of the most recent work in genome-wide protein chemical interaction prediction has focused on ensemble methods built from local models that make predictions about a small region of interaction space. This approach is well motivated, since it is unlikely that a single model can generalize to make predictions regarding many diverse protein families and chemotypes. There are several issues with this current work on interaction prediction, and the research described in this section aims to address each of these issues. Boosting is examined as a generable ensemble learning framework, under which two modifications are made: 1) adjust sample reweighting based on sample similarity as well as misclassification, and 2) weight model predictions based on sample similarity as well, so that models make predictions only for samples most similar to previous samples that were correctly classified.

The approach proposed in [5] using local models strongly resembles the use of ensemble methods in which a number of weakly performing classifiers are organized in such a way that their predictions can be combined into a strongly performing classifier. In the previous approach, a local model for each unknown protein-chemical interaction is built using only known samples that share a protein or chemical with the unknown interaction. This method is analogous to an ensemble approach where each model makes a prediction for only a single sample, and is built using only the most relevant data to that sample. For genome-wide protein-chemical interaction prediction, local models are useful because the chemical-protein interaction space is large and built from a diverse number of protein families as well as chemotypes. Partitioning that space into more homogenous segments and building specialized models for each segment is a much more reasonable approach than attempting to build a single model to make predictions predictions across all interactions. The drawback of using local models is that each model is *too* specialized, and broader similarities between samples are ignored during both learning and prediction. Further, known ensemble learning approaches have not been explored in this context.

Chapter 3

Pattern-based Kernel Methods for Protein-chemical Interaction Prediction

Traditional approaches to graph similarity rely on the comparison of compounds using a variety of molecular attributes known *a priori* to be involved in the activity of interest. Such methods are problem-specific, however, and provide little assistance when the relevant descriptors are not known in advance. Additionally, these methods lack the ability to provide explanatory information regarding what structural features contribute to the observed chemical activity. The method described here, referred to as OAPD for Optimal-Assignment with Pattern-based Descriptors, alleviates both of these issues through the mining and analysis of structural patterns present in the data in order to identify highly discriminating patterns, which then augment a graph kernel function that computes molecular similarity.

3.1 Structure Pattern Mining

The frequent subgraph mining problem can be phrased as such: given a set of labeled graphs, the support of an arbitrary subgraph is the fraction of all graphs in the set that contain that subgraph. A subgraph is frequent if its support meets a certain minimum threshold. The goal is to enumerate all the frequent, connected subgraphs in a graph database. The extraction of important subgraph patterns can be controlled by selecting the proper frequency threshold, as well as other parameters such as size and density of subgraph patterns.

3.2 Optimal Assignment Kernel

The optimal assignment kernel function computes the similarity between two graph structures. This similarity computation is accomplished by first representing the two sets graph vertices as a bipartite graph, and then finding the set of weighted edges assigning every vertex in one graph to a vertex in the other. The edge weights are calculated via a recursive vertex similarity function. This section presents the equations describing this algorithm in detail, as discussed by Frölich et al [16]. The top-level equation describing the similarity of two molecular graphs is:

$$k_A(M_1, M_2) := max_\pi \sum_{h=1}^{m} k_{nei}(v_{\pi(h)}, v_h) \tag{3.1}$$

Where π denotes a permutation of a subset of graph vertices, and m is the number of vertices in the smaller graph. This is needed to assign all vertices of the smaller graph to vertices in the large graph. The k_{nei} function, which calculates the similarity between two vertices using their local neighbors, is given as follows:

$$k_{nei}(v_1, v_2) := k_v(v_1, v_2) + R_0(v_1, v_2) + S_{nei}(v_1, v_2) \tag{3.2}$$

$$S_{nei}(v_1, v_2) := \sum_{l=1}^{L} \gamma(l) R_l(v_1, v_2) \tag{3.3}$$

The functions k_v and k_e compute the similarity between vertices (atoms) and edges (bonds), respectively. These functions could take a variety of forms, but in the OA kernel they are RBF functions between vectors of vertex/edge labels.

The $\gamma(l)$ term is a decay parameter that weights the similarity of neighbors according to their distance from the original vertex. The l parameter controls the topological distance within which to consider neighbors of vertices. The R_l equation, which recursively computes the similarity between two specific vertices is given by the following equation:

$$R_l(v_1, v_2) = \frac{1}{|v_1||v_2|} \sum_{i,j} R_{l-1}(n_i(v_1), n_j(v_2)) \tag{3.4}$$

Where $|v|$ is the number of neighbors of vertex v, and $n_k(v)$ is the set of neighbors of v. The base case for this equation is R_0, defined by:

$$R_0(v_1, v_2) := \frac{1}{|v_1|} max_\pi \sum_{i=1}^{|v_2|} (k_v(a, b)|k_e(x, y)) \tag{3.5}$$

$$a = n_{\pi(i)}(v_1),\ b = n_i(v_2) \tag{3.6}$$

$$x = v_1 \rightarrow n_{\pi(i)}(v_1),\ y = v_2 \rightarrow n_i(v_2) \tag{3.7}$$

The notation $v \rightarrow n_i(v)$ refers to the edge connecting vertex v with the ith neighboring vertex. The functions k_v and k_e are used to compare vertex and edge descriptors, by counting the total number of descriptor matches.

3.3 Reduced Graph Representation

One way in which to utilize the structure patterns that are mined from the graph data is to collapse the specific subgraphs into single vertices in the original graph. This technique is explored by Frölich et al. [17] with moderate results, although they use predefined structure patterns, so called pharmacophores, identified *a priori* with the help of expert knowledge. The method described here ushers these predefined patterns in favor of the structure patterns generated via frequent subgraph mining.

The use of a reduced graph representation does have some advantages. First, by collapsing substructures, an entire set of vertices can be compared at once, reducing the graph complexity and marginally decreasing computation time. Second, by changing the substructure size the resolution at which graph structures are compared can be adjusted. The disadvantage of a reduced graph representation is that substructures can only be compared directly to other substructures, and cannot align partial structure matches. As utilized in Frölich et al., this is not as much of a burden since they have defined the best patterns *a priori* using expert knowledge. In the case of the method presented here, however, this is a significant downside, as there is no *a priori* knowledge to guide pattern generation and we wish to retain as much structural information as possible.

3.4 Pattern-based Descriptors

The loss of partial substructure alignment following the use of a reduced graph representation motivated us to find another way of integrating this pattern-based information. Instead of collapsing graph substructures, vertices are simply annotated with additional descriptor labels indicating the vertex's membership in the structure patterns that were previously mined. These pattern-based descriptors are calculated for each vertex and are used by the optimal assignment kernel in the same way that other vertex descriptors are handled. In this way substructure information can be captured in the graph vertices without needing to alter the original graph structure.

Chapter 4

Approximate Alignment Kernel Methods for Protein-chemical Interaction Prediction

The following sections outline algorithms for approximate graph alignment. This method measures the similarity of graph structures whose nodes and edges have been labeled with various features. These features represent different kinds of chemical structure information including atoms and chemical bonds types among others. To compute the similarity of two graphs, the nodes of one graph are aligned with the nodes of the second graph, such that the total overall similarity is maximized with respect to all possible alignments. Vertex similarity is measured by comparing vertex descriptors, and is computed recursively so that when comparing two nodes, the immediate neighbors of those nodes are also compared, and the neighbors of those neighbors, and so on.

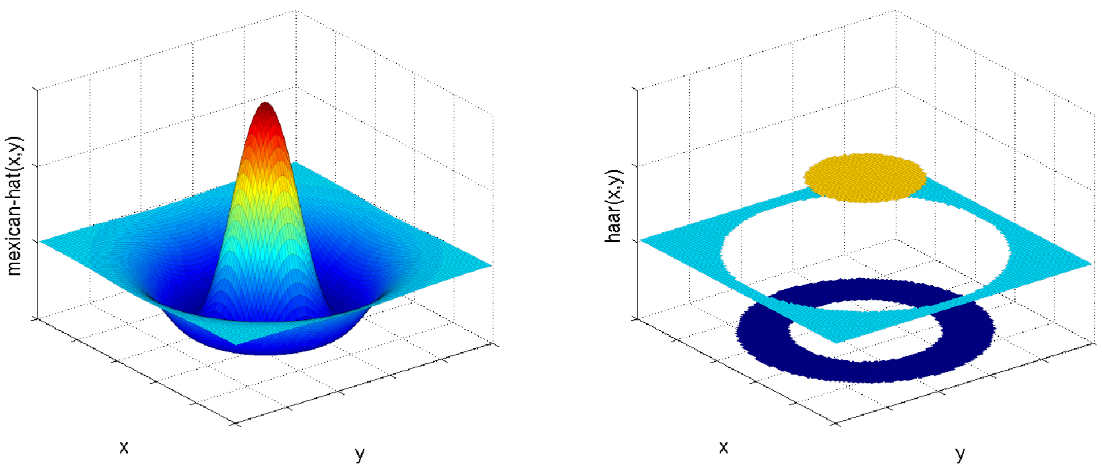

Figure 4.1. Two wavelet functions in three dimensions, Mexican hat and Haar.

4.1 Graph Alignment Kernel

An *alignment* of two graphs G and G' (assuming $|V[G] \leq |V[G']|$) is a 1-1 mapping $\pi : V[G] \to V[G']$. Given an alignment π, define the similarity between two graphs, as measured by a kernel function k_A, below:

$$\begin{aligned} k_A(G, G') := \max_\pi \textstyle\sum_{v \in V[G]} k_n(v, \pi(v)) + \\ \textstyle\sum_{u,v} k_e((u, v), (\pi(u), \pi(v))) \end{aligned} \tag{4.1}$$

The function k_n is a kernel function to measure the similarity of nodes and the function k_e is a kernel function to measure the similarity of edges. Intuitively, equation 4.1 use an additive model to compute the similarity between two graphs by computing the sum of the similarity of nodes and the similarity of edges. The maximal similarity among all possible alignments is defined as the similarity between two graphs.

4.2 Simplified Graph Alignment Kernel

A direct computation of the graph alignment kernel is computationally intensive and is unlikely to be scalable to large graphs. With no surprise, the graph alignment kernel computation is no easier than the subgraph isomorphism problem, a known NP-hard problem [1]. To derive efficient algorithms scalable to large graphs, the graph kernel function is simplified with the following formula:

$$k_M(G, G') = \max_{\pi} \sum_{v \in V[G]} k_a(f(v), f(\pi(v))) \tag{4.2}$$

Where $\pi : V[G] \to V[G']$ denotes an alignment of graph G and G'. $f(v)$ is a set of features associated with a node that not only include node features but also include information about topology of the graph where v belongs to.

Equation 4.2, computes a maximal weighted bipartite graph, which has an efficient solution known as the Hungarian algorithm. The complexities of the algorithm is $O(|V[G]|^3)$. See [17] for further details.

Provided below is an efficient method, based on graph wavelet analysis, to create features to capture the topological structure of a graph.

4.3 Graph Wavelet Analysis

Originally proposed to analyze time series signals, wavelet analysis transforms a series of signals to a set of summaries with different scale. Two of the key insights of wavelet analysis of signals are (i) using localized basis functions and (ii)

[1]Formally, showing a reduction from the graph alignment kernel to the subgraph isomorphism problem is needed. The details of such reduction are omitted due to their loose connection to the main theme of the current paper, which is advanced data mining approach as applied to cheminformatics applications

analysis with different scales. Wavelet analysis offers efficient tools to decompose and represent a function with arbitrary shape [13, 19]. Since invented, wavelet analysis has quickly gained popularity in a wide range of applications outside time series data, such as image analysis and geography data analysis. In all these applications, the level of detail, or *scale* is considered as an important factor in data comparison and compression. Figure 4.1 shows two examples of wavelet functions in a 3D space, the Haar and Mexican Hat.

Intuition. With wavelet analysis as applied to graph representations chemical structure, for each atom, features about the atom and its local environment are collected at different scales. For example, information can be collected about the average charge of an atom and it's surrounding atoms, then assign the average value as a feature to the atom. Information can also be collected about whether an atom belongs to a nearby functional group, whether the surrounding atoms of a particular atom belong to a nearby functional group, and the local topology of an atom to its nearby functional groups.

In summary, conceptually the following two types of insights are gained about the chemicals after applying wavelet analysis to graph represented chemical structure:

- *Analysis with varying levels of scale.* Intuitively, at the finest level, two chemical structures are compared by matching the atoms and chemical bonds in the two structures. At the next level, comparison of two regions is performed (e.g. chemical functional groups). At an even coarser level, small regions may be grouped into larger ones (e.g. pharmacophore), and two chemicals are compared by matching the large regions and the connections among large regions.

- *Non-local connection.* In a chemical structure, two atoms that are not directly connected by a chemical bond may still have some kind of interaction. Therefore when comparing two graphs and their vertices cannot depend only on the *local environment* immediately surrounding an atom, but rather must consider both local and non-local environment.

Though conceptually appealing, current wavelet analysis is often limited to numerical data with regularly structures such as matrices and images. Graphs, however, are arbitrarily structured and may represent innumerable relationships and topologies between data elements. In order to define a reasonable graph wavelet functions, the following two important concepts are introduced:

- h-hop neighborhood
- Discrete wavelet functions

The former, h-hop neighborhood, is essentially used to project graphs from a high dimensional space with arbitrary topology into a Euclidean space suitable for operation with wavelets. The h-hop measure defines a distance metric between vertices that is based on the shortest path between them. The discrete wavelet function then operates on a graph projection in the h-hop Euclidean space to compactly represent the information about the local topology of a graph. It is the use of this compact wavelet representation in vertex comparison that underlies the complexity reduction achieved by this method. Based on the h-hop neighborhood, a discrete wavelet function is used to summarize information in a local region of a graph and create features based on the summarization. These two concepts are discussed in detail below.

h-hop neighborhood In this section the following definitions are introduced.

Definition 4.3.1 *Given a node v in a graph G the h-**hope neighborhood** of v, denoted by $N_h(v)$, is the set of nodes that are (according to the shortest path) exactly h hops away from v.*

For example if $h = 0$, then $N_0(v) = v$ and if $h = 1$, then $N_1(v) = \{u|(u, v) \in E[G]\}$.

Here f_v denotes the feature vector associated with a node v in a graph G. $|f|$ is the feature vector length (number of features in the feature vector). The average feature measurement, denoted by $\overline{f}_j(v)$ for nodes in $N_j(v)$ is

$$\overline{f}_j(v) = \frac{1}{|N_j(v)|} \sum_{u \in N_j(v)} f_u \tag{4.3}$$

Example 4.3.1 *The left part of the Figure 4.2 shows a chemical graph. Given a node v in the graph G, label the shortest distance of nodes to v in the G. In this case $N_0(v) = v$ and $N_1(v) = \{t, u\}$. If the feature vector contains a single feature of atomic number, $\overline{f}_1(v)$ is the average atomic number of atoms that are at most 1-hop away from v. In this case, since $N_1(v) = \{t, u\}$ and $\{t, u\}$ are both carbon with atomic number equal to eight, then $\overline{f}_1(v)$ is equal to eight as well.*

4.3.0.1 Discrete wavelet functions

In order to adapt a wavelet function to discrete structure such as graphs, a wavelet function $\psi(x)$ must be applied to the h-hop neighborhood. Towards that end, a wavelet function $\psi(x)$ (such as the Haar, or Mexican Hat) can be scaled to have support on the domain $[0, 1)$, with integral 0, and partition the function

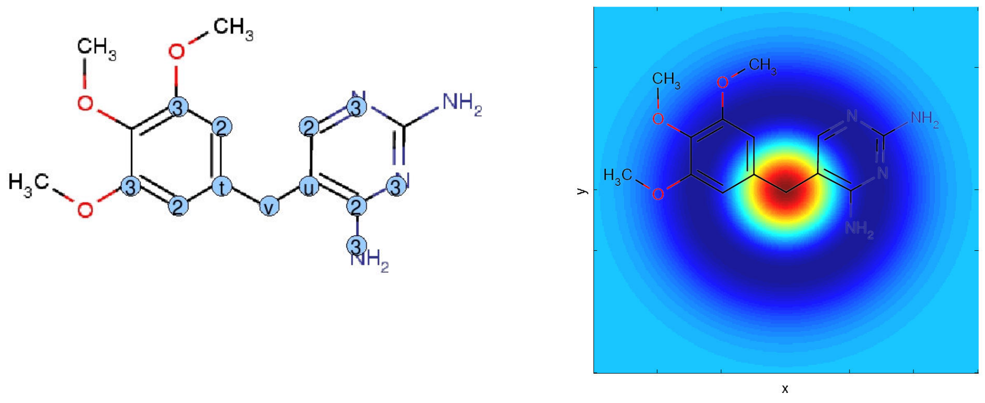

Figure 4.2. A chemical graph and hop distances.

into $h+1$ intervals. Then compute the average, $\psi_{j,h}$, as the average of $\psi(x)$ over the jth interval, $0 \le j \le h$ as below.

$$\psi_{j,h} \equiv \frac{1}{h+1} \int_{j/(h+1)}^{(j+1)/(h+1)} \psi(x)dx \tag{4.4}$$

With neighborhood and discrete wavelet functions, wavelet analysis can be applied to graphs. This analysis is called *wavelet measurements*, denoted by $\Gamma_h(v)$, for a node v in a graph G at scale up to $h > 0$.

$$\Gamma_h(v) = C_{h,v} * \sum_{j=0}^{h} \psi_{j,h} * \overline{f}_j(v) \tag{4.5}$$

where $C_{h,v}$ is a normalization factor with $C(h,v) = (\sum_{j=0}^{h} \frac{\psi_{j,h}^2}{|N_k(v)|})^{-1/2}$

Define $\Gamma^h(v)$ as the sequence of wavelet measurements as applied to a node v with scale value up to h. That is $\Gamma^h(v) = \{\Gamma_1(v), \Gamma_2(v), \ldots, \Gamma_h(v)\}$. Call $\Gamma^h(v)$ the *wavelet measurement vector* of node v. Finally insert the wavelet measurement vector into the alignment kernel with the following formula.

$$k_\Gamma(G, G') = \max_{\pi} \sum_{v \in V[G]} k_a(\Gamma^h(v), \Gamma^h(\pi(v))) \tag{4.6}$$

where $k_a(\Gamma^h(v), \Gamma^h(\pi(v))$ is a kernel function defined on vectors. Two popular choices are linear kernel and radius based function kernel.

Example 4.3.2 *The right part of Figure 4.2 shows a chemical graph overlayed with a wavelet function centered on a specific vertex. It is clear how the wavelet is most intense at the central vertex, hop distance of zero, corresponding to a strongly positive region of the wavelet function. As the hop distance increases the wavelet function becomes strongly negative, roughly at hop distances of one and two. At hop distance greater than two, the wavelet function returns to zero intensity, indicating negligible contribution from vertices at this distance.*

Chapter 5

Feature Approximations for Protein-chemical Interaction Prediction

This chapter discusses approximations for chemical structure features in the context of both structure alignment and matching. In both approaches, chemical features describing elements of chemical structures (such as specific atom properties or membership of atoms in structure fragment features) are diffused throughout the chemical graph, allowing for more relaxed alignment and matching of chemical structures.

5.1 Feature Approximation for Structure Alignment

The work presented in this chapter aims to leverage existing frequent pattern mining algorithms and explore the application of kernel classifiers in building highly accurate graph classification algorithms. Towards that end, a technique is demonstrated called graph pattern diffusion kernel (GPD). In this method, all frequent patterns are first identified from a graph database. Then subgraphs are mapped to graphs in the graph database and nodes of graphs are projected to a high dimensional space with a specially designed function. Finally a graph alignment algorithm is used to compute the inner product of two graphs. This algorithm is tested using a number of chemical structure data sets. The experimental results demonstrate that this method is significantly better than competing methods such as those based on paths, cycles, and other subgraphs.

Here the design of the pattern diffusion kernel is presented. The section begins by first presenting a general framework. It is proved, through a reduction to the subgraph isomorphism problem, that the computational cost of the general framework can be prohibitive for large graphs. The pattern based graph alignment kernel is then presented. Finally a technique is shown called "pattern diffusion" that can significantly improve graph classification accuracy in practice.

5.1.1 Graph Similarity Measurement with Alignment

An *alignment* of two graphs G and G' (assuming $|V[G]| \leq |V[G']|$) is a 1-1 mapping $\pi : V[G] \to V[G']$. Given an alignment π, define the similarity between two graphs, as measured by a kernel function k_A, below:

$$k_A(G, G') = \max_{\pi} \sum_{v} k_n(v, \pi(v)) + \sum_{u,v} k_e((u, v), (\pi(u), \pi(v))) \tag{5.1}$$

The function k_n is a kernel function to measure the similarity of node labels and the function k_e is a kernel function to measure the similarity of edge labels. Equation 5.1 uses an additive model to compute the similarity between two graphs. The maximal similarity among all possible mappings is defined as the similarity between two graphs.

5.1.2 NP-hardness of Graph Alignment Kernel Function

It is no surprise that computing the graph alignment kernel is an NP-hard problem. It is proved this with a reduction from the graph alignment kernel to the subgraph isomorphism problem. In the following paragraphs, assuming there exists an efficient solver of the graph alignment kernel problem, it is shown that the same solver can be used to solve the subgraph isomorphism problem efficiently. Since the subgraph isomorphism problem is an NP-hard problem, with the reduction mentioned before, it is proved that the graph alignment kernel problem is therefore an NP-hard problem as well. Note: this subsection is a stand-alone component of this work, and readers who choose to skip this section should encounter no difficulty in reading the rest of the text.

Given two graphs G and G' (for simplicity, assume nodes and edges in G and G' are not labeled as usually studied in the subgraph isomorphism problem), use a node kernel function that returns a constant 0. Define an edge kernel function $k_e : V[G] \times V[G] \times V[G'] \times V[G'] \to \mathbb{R}$ as

$$k_e((u,v),(u',v')) = \begin{cases} 1 & \text{if } (u,v) \in E[G] \text{ and } (u',v') \in E[G'] \\ 0 & \text{otherwise} \end{cases}$$

With the constant node function and the specialized edge function, the kernel function of two graphs is simplified to the following format:

$$k_A(G,G') = \max_{\pi} \sum_{u,v} k_e((u,v),(\pi(u),\pi(v))) \tag{5.2}$$

The NP-hardness of the graph alignment kernel is established with the following theorem.

Theorem 5.1.1 *Given two (unlabeled) graphs G and G' and the edge kernel function k_e defined previously, G is subgraph isomorphic to G' if and only if $K_a(G, G') = |E[G]|$*

Proof 5.1.1 *If: notice from the definition of k_e that the maximal value of $K_a(G, G')$ is $|E[G]|$. Given $K_a(G, G') = |E[G]|$, it is claimed that there exists an alignment function $\pi : V[G] \to V[G']$ such that for all $(u, v) \in E[G]$, $(\pi(u), \pi(v)) \in E[G']$. The existence of such a function π guarantees that graph G is a subgraph of G'.*

Only if: Given G is a subgraph of G', there is an alignment function $\pi : V[G] \to V[G']$ such that for all $(u, v) \in E[G]$, $(\pi(u), \pi(v)) \in E[G']$. According to Equation 5.2, $K_a(G, G') = |E[G]|$.

Theorem 5.1.1 shows that the graph alignment kernel problem is no easier than the subgraph isomorphism problem and hence is at least NP-hard in complexity.

5.1.3 Graph Node Alignment Kernel

To derive an efficient algorithm scalable to large graphs, the idea is that a function f is used to map nodes in a graph to a high (possibly infinite) dimensional feature space that captures not only the node label information but also the neighborhood topological information around the node. If such a function f is obtained, the graph kernel function may be simplified with the following formula:

$$k_M(G, G') = \max_{\pi} \sum_{v \in V[G]} k_n(f(v), f(\pi(v))) \tag{5.3}$$

Where $\pi : V[G] \rightarrow V[G']$ denotes an alignment of graph G and G'. $f(v)$ is a set of "features" associated with a node.

With this modification, the optimization problem that searches for the best alignment can be solved in polynomial time. To derive a polynomial running time algorithm, a weighted complete bipartite graph is constructed by making every node pair (u,v) $\in V[G] \times V[G']$ incident on an edge. The weight of the edge (u,v) is $k_n(f(v), f(u))$. Figure 5.1, shows a weighted complete bipartite graph for $V[G] = \{v_1, v_2, v_3\}$ and $V[G'] = \{u_1, u_2, u_3\}$. Highlighted edges $(v1, u2)$, $(v2, u1)$, $(v3, u3)$ have larger weights than the rest of the edges (dashed).

With the bipartite graph, a search for the best alignment becomes a search for the maximum weighted bipartite subgraph from the complete bipartite graph. Many network flow based algorithms (e.g. linear programming) can be used to obtain the maximum weighted bipartite subgraph. The Hungarian algorithm is used with complexity $O(|V[G]|^3)$. For details of the Hungarian algorithm see [2].

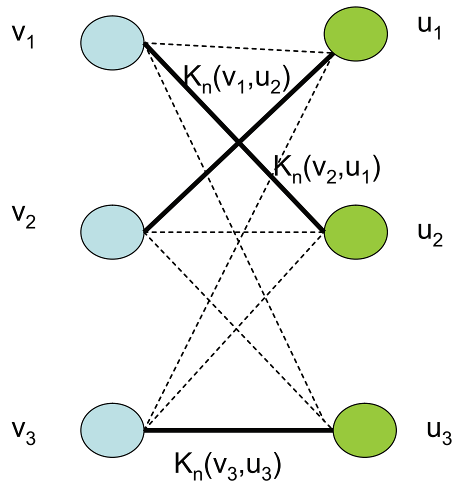

Figure 5.1. A maximum weighted bipartite graph for graph alignment.

Applying the Hungarian algorithm to graph alignment was first explored by [17] for chemical compound classification. In contrast to their algorithm, which utilized domain knowledge of chemical compounds extensively and developed a complicated recursive function to compute the similarity between nodes, a new framework is developed here that maps such nodes to a high dimensional space in order to measure the similarity between two nodes without assuming any domain knowledge. Even in cheminformatics, experiments show that this technique generates similar and sometimes better classification accuracies compared to the method reported in [17].

Unfortunately, using the Hungarian algorithm for assignment, as used by [17] is not a true Mercer kernel. Since the kernel function described here uses this algorithm as well, it is also not a Mercer kernel. Like in [17], however, practically this kernel still performs competitively.

5.1.4 Pattern Diffusion

This section introduces a technique "pattern diffusion" to project nodes in a graph to a high dimensional space that captures both node labeling information and local topology information. This design has the following advantages as a kernel function:

- The design is generic and does not assume any domain knowledge from a specific application. The diffusion process may be applied to graphs with dramatically different characteristics.
- The diffusion process is straightforward to implement and can be computed efficiently.
- It is proved that the diffusion process is related to the probability distribution of a graph random walk (in Appendix). This explains why the simple

process may be used to summarize local topological information.

Below, the pattern diffusion kernel is outlined in three steps.

In the first step, a seed is identified as a starting point for the diffusion. In this design, a “seed” could be a single node, or a set of connected nodes in the original graph. In the experimental study, frequent subgraphs are used for seeds since a seed can easily be compared from one graph to a seed in another graph. However, there is no requirement that frequent subgraphs must be used.

In the second step given a set of nodes S as seed, recursively define f_t in the following way.

The base f_0 is defined as:

$$f_0(u) = \begin{cases} 1/|S| & \text{if } u \in S \\ 0 & \text{otherwise} \end{cases}$$

Given some time t, define f_{t+1} $(t \geq 0)$ with f_t in the following way:

$$f_{t+1}(v) = f_t(v) \times (1 - \frac{\lambda}{d(v)}) + \sum_{u \in N(v)} f_t(u) \times \frac{\lambda}{d(u)} \tag{5.4}$$

In the notation, $N(v)$ is the set of nodes that connects to v directly. $d(v)$ is the node degree of v, or $d(v) = |N(v)|$. λ is a parameter that controls the diffusion rate.

The formula 5.4 describes a process where each node distributes a λ fraction of its value to its neighbors evenly and in the same way receives some value from its neighbors. Call it “diffusion” because the process simulates the way a value is spreading in a network. The intuition is that the distribution of such a value encodes information about the local topology of the network.

To constrain the diffusion process to a local region, one parameter called diffusion time is used, denoted by τ, to control the diffusion process. Specifically the diffusion process is limited to a local region of the original graph with nodes that are at most τ hops away from a node in the seed S. For this reason, the diffusion is referred to as "local diffusion".

Finally, for the seed S, define the mapping function f_S as the limit function of f_t as t approaches to infinity, or

$$f_S = \lim_{t \to \infty} f_t \tag{5.5}$$

5.1.5 Pattern Diffusion Kernel and Graph Classification

This section summarizes the discussion of kernel functions and shows how they are utilized to construct an efficient graph classification algorithm at both the training and testing phases.

5.1.5.1 Training Phase

In the training phase, divide graphs of the training data set $D = \{(G_i, T_i,)\}_{i=1}^{n}$ into groups according to their class labels. For example in binary classification, there are two groups of graphs: positive or negative. For multi-class classification, there are multiple groups of graphs where each group contains graphs with the same class label. The training phase is composed of four steps:

- Obtain frequent subgraphs for seeds. Identify frequent subgraphs from each graph group and union the subgraph sets together as the seed set $\mathcal{S}$.
- For each seed $S \in \mathcal{S}$ and for each graph G in the training data set, use f_S to label nodes in G. Thus the feature vector of a node v is a vector

$L_V = \{f_{S_i}(v)\}_{i=1}^{m}$ with length $m = |\mathcal{S}|$.

- For two graphs G, G', construct the complete weighted bipartite graph as described in section 5.1.3 and compute the kernel $K_a(G, G')$ using Equation 5.3.
- Train a predictive model using a kernel classifier.

5.1.5.2 Testing Phase

In the testing phase, the kernel function is computed for graphs in the testing and training data sets. The trained model is used to make predictions about graph in the testing set.

- For each seed $S \in \mathcal{S}$ and for each graph G in the testing data set, f_S is used to label nodes in G and create feature vectors as done in the training phase.
- Equation 5.3 computes the kernel function $K_a(G, G')$ for each graph G in the testing data set and for each graph G' in the training data set.
- Use kernel classifier and trained models to obtain prediction accuracy of the testing data set

5.2 Feature Approximation for Structure Matching

This section expands on the GPD kernel presented in the previous section, by defining a similar kernel function that uses a matching-based set kernel instead of an alignment kernel. This method is termed a Graph Pattern Matching (GPM) kernel. The advantage of this modification is that the GPM kernel, unlike GPD, is guaranteed to be positive semi-definite, and hence a true Mercer kernel. This algorithm was tested using 16 chemical structure data sets. The experimental results demonstrate that this method outperforms existing state-of-the-art methods with a large margin.

This section presents the design of a graph matching kernel with diffusion. The section begins by first presenting a general framework for graph matching. Then the pattern based graph matching kernel is presented. Finally a technique called "pattern diffusion" is discussed that significantly improves graph classification accuracy in practice.

5.2.1 Graph Matching Kernel

To derive an efficient algorithm scalable to large graphs, a function $\Gamma : V \rightarrow \mathbb{R}^n$ is used to map nodes in a graph to a n dimensional feature space that captures not only the node label information but also the neighborhood topological information around the node. If there is such a function Γ, the following graph kernel may be defined:

$$K_m(G, G') = \sum_{(u,v) \in V[G] \times V[G']} K(\Gamma(u), \Gamma(v)) \tag{5.6}$$

K can be any kernel function defined in the co-domain of Γ. This function K_m is called a *graph matching kernel*. The following theorem indicates that K_m is symmetric and positive semi-definite and hence a real kernel function.

Theorem 5.2.1 *The graph matching kernel is symmetric and positive semi-definite if the function K is symmetric and positive semi-definite.*

Proof sketch: the matching kernel is a special case of the R-convolution kernel and is hence positive semi-definite as proved in [45].

The kernel function can be visualized by constructing a weighted complete bipartite graph: connecting every node pair (u,v) $\in V[G] \times V[G']$ with an edge. The weight of the edge (u,v) is $K(\Gamma(v), \Gamma(v))$. Figure 5.2 shows a weighted complete bipartite graph for $V[G] = \{v_1, v_2, v_3\}$ and $V[G'] = \{u_1, u_2, u_3\}$. Highlighted edges $(v1, u2)$, $(v2, u1)$, $(v3, u3)$ have larger weights than the rest of the edges (dashed).

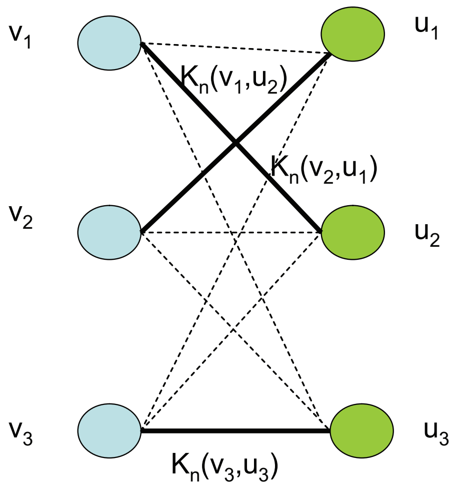

Figure 5.2. The maximum weighted bipartite graph for graph matching.

From the figure it can be seen that if two nodes are quite dissimilar, the weight of the related edge is small. Since dissimilar node pairs usually outnumber similar node pairs, if a linear kernel is used for nodes, kernel function may be noisy and hence lose the signal. In this design, the RBF kernel function is used, as specified below, to penalize dissimilar node pairs.

$$K(X, Y) = e^{\frac{-||X-Y||_2^2}{2}} \tag{5.7}$$

where $||X||_2^2$ is the squared L_2 norm of a vector X.

5.2.2 Graph Pattern Matching Kernel

One way to design the function Γ is to take advantage of frequent patterns mined from a set of graphs. Intuitively if a node belongs to a subgraph F, there is some information about the local topology of the node. Following the intuition, given a node v in a graph G and a frequent subgraph F, a function Γ_F is designed such that

$$\Gamma_F(v) = \begin{cases} 1 & \text{if } u \text{ belongs an embedding of } F \text{ in } G \\ 0 & \text{otherwise} \end{cases}$$

The function Γ_F is called a "pattern membership function" since this function tests whether a node occurs in a specific subgraph feature ("membership to a subgraph").

Given a set of frequent subgraphs $\mathcal{F} = F_1, F_2, \ldots, F_n$, each membership function is treated as a dimension and the function $\Gamma_{\mathcal{F}}$ is defined as below:

$$\Gamma_{\mathcal{F}}(v) = (\Gamma_{F_i}(v))_i^n \tag{5.8}$$

In other words, given n frequent subgraph, the function Γ maps a node v in G to a n-dimensional space, indexed by the n subgraphs, where values of the features indicate whether the node is part of the related subgraph in G.

Example 5.2.1 *In Figure 5.3, it is shown that two subgraph features F_1 and F_2. F_1 have an embedding in Q at $\{q_1, q_2\}$ and F_2 occurs in Q at $\{q_1, q_3\}$. The occurrences are depicted using shadings with different color and orientations. For node q_1, a subgraph F_1 is considered as a feature, and $\Gamma_{F_1}(q_1) = 1$ since q_1 is part of an embedding of F_1 in Q. Also, $\Gamma_{F_1}(q_3) = 0$ since q_3 is not part of an embedding* *of F_1 in Q. Similarly, $\Gamma_{F_2}(q_1) = 1$ and $\Gamma_{F_2}(q_3) = 1$. Hence $\Gamma_{F_1,F_2}(q_1) = (1,1)$ and $\Gamma_{F_1,F_2}(q_3) = (0,1)$. The values of the function Γ_{F_1,F_2} are also illustrated in the same figure using the annotated Q.*

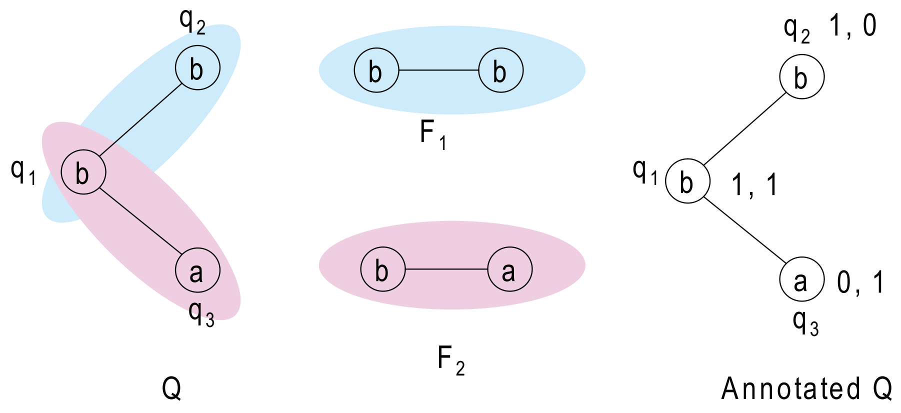

Figure 5.3. Example pattern membership functions for GPM kernel.

5.2.3 Graph Pattern Matching Kernel with Pattern Diffusion

This section introduces a better technique than the pattern membership function to capture the local topology information of nodes. This technique is called "pattern diffusion". It's design has the following advantages:

- It is generic and does not assume any domain knowledge from a specific application. The diffusion process may be applied to graphs with dramatically different characteristics.
- The diffusion process is straightforward to implement and can be computed efficiently.
- It is prove that the diffusion process is related to the probability distribution of a graph random walk. This explains why the simple process may be used to summarize local topological information.

Below, the pattern diffusion kernel is outlined in three steps.

In the first step, a seed is identified as a starting point for the diffusion. In this design, a "seed" could be a single node, or a set of connected nodes in the original graph. In the experimental study, frequent subgraphs are always used for seeds since a seed from one graph can be easily compared to a seed in another graph.

In the second step given a set of nodes S as seed, a diffusion function f_t is recursively defined in the following way.

The base f_0 is defined as:

$$f_0(u) = \begin{cases} 1/|S| & \text{if } u \in S \\ 0 & \text{otherwise} \end{cases}$$

Define f_{t+1} $(t \geq 0)$ with f_t in the following way:

$$f_{t+1}(v) = f_t(v) \times (1 - \frac{\lambda}{d(v)}) + \sum_{u \in N(v)} f_t(u) \times \frac{\lambda}{d(u)} \tag{5.9}$$

In the notation, $N(v) = \{u | (u, v) \text{ is an edge } \}$ is the set of nodes that connects to v directly. $d(v) = |N(v)|$ is the node degree of v. λ is a parameter that controls the diffusion rate.

The formula 5.9 describes a process where each node distributes a λ fraction of its value to its neighbors evenly and in the same way receives some value from its neighbors. It is called "diffusion" because the process simulate the way a value is spreading in a network. The intuition is that the distribution of such a value encodes information about the local topology of the network.

To constrain the diffusion process to a local region, one parameter called diffusion time, denoted by τ, is used to control the diffusion process. Specifically the diffusion process is limited to a local region of the original graph with nodes that are at most τ hops away from a node in the seed S. In this sense, the diffusion should be named "local diffusion".

Finally in the last step, for the seed S, define the mapping function Γ^d_S as the limit function of f_t as t approaches to infinity, or

$$\Gamma^d_S = \lim_{t \to \infty} f_t \tag{5.10}$$

And given a set of frequent subgraph $\mathcal{F} = F_1, F_2, \ldots, F_n$ as seeds, define the pattern diffusion function $\Gamma^d_{\mathcal{F}}$ as:

$$\Gamma^d_{\mathcal{F}}(v) = (\Gamma^d_{F_i}(v))^n_i \tag{5.11}$$

5.2.4 Connections of Other Graph Kernels

5.2.4.1 Connection to Marginalized Kernels

Here the connection of pattern matching kernel function to the marginalized graph kernel [34] is shown, which uses a Markov model to randomly generate walks of a labeled graph.

Given a graph G with nodes set $V[G] = \{v_1, v_2, \ldots, v_n\}$, and a seed $S \subseteq V[G]$, for each diffusion function f_t, construct a vector $U_t = (f_t(v_1), f_t(v_2), \ldots, f_t(v_n))$. According to the definition of f_t, $U_{t+1} = M \times U_t$ where the matrix M is defined as:

$$M(i,j) = \begin{cases} \frac{\lambda}{d(v_j)} & \text{if } i \neq j \text{ and } i \in N(j) \\ 1 - \frac{\lambda}{d(v_i)} & i = j \\ 0 & \text{otherwise} \end{cases}$$

In this representation, compute the stationary distribution ($f_S = \lim_{t\to\infty} f_t$) by computing $M^\infty \times U_0$.

Notice that the matrix M corresponds to a probability matrix corresponding to a Markov Chain since

- all entries are non-negative
- column sum is 1 for each column

Therefore the vector $M^\infty \times U_0$ corresponds to the stationary distribution of the local random walk as specified by M. In other words, rather than using random walk to retrieve information about the local topology of a graph, the stationary distribution is used to retrieve information about the local topology. The experimental study shows that this in fact is an efficient method of graph classification.

5.2.4.2 Connection to Optimal Assignment Kernel

The optimal assignment (OA) kernel [17] carries the same spirit of the graph pattern matching kernel in that OA uses pairwise node kernel function to construct a graph kernel function. OA kernel has been utilized for cheminformatics applications and is found to deliver good results empirically.

There are two major differences between GPM and the OA kernel. (1) OA kernel is not positive semi-definite and hence is not Mercer kernel in a strict sense. Non Mercer kernel functions are used to train SVM model and the problem is that the convex optimizer utilized in SVM will not converge to a global optimal and hence the performance of the SVM training may not be reliable. (2) OA utilizes a complicated recursive function to compute the similarity between nodes, which make the computation of the kernel function runs slowly for large graphs [58].

5.2.5 Pattern Diffusion Kernel and Graph Classification

This section summarizes the discussions presented so far and shows how the kernel function is utilized to construct an efficient graph classification algorithm in both the training and testing phases.

5.2.5.1 Training Phase

In the training phase, graphs of the training data set $D = \{(G_i, T_i,)\}_{i=1}^n$ are divided into groups according to their class labels. For example in binary classification, two groups of graphs: positive or negative. For multi-class classification, graphs are partitioned according to their class label where graphs have the same class labels are grouped together. The training phase is composed of four steps:

- Obtain frequent subgraphs. Identify frequent subgraphs from each graph group and union the subgraph sets together as the seed set $\mathcal{F}$.
- For each graph G in the training data set, use the node pattern diffusion function $\Gamma_{\mathcal{F}}^d$ to label nodes in G. Thus the feature vector of a node v is a vector $L_V = (\Gamma_{F_i}^d(v))_{i=1}^m$ with length $m = |\mathcal{F}|$.
- For two graphs G, G', construct the complete weighted bipartite graph as described in section 5.2.1 and compute the kernel $K_m(G, G')$ using Equation 5.6 and Equation 5.7.
- Train a predictive model using a kernel classifier.

5.2.5.2 Testing Phase

In the testing phase, the kernel function is computed for graphs in the testing and training data sets. The trained model is used to make predictions about graph in the testing set.

- For each graph G in the testing data set, use $\Gamma^d_{\mathcal{F}}$ to label nodes in G and create feature vectors as in the training phase.

- Use Equation 5.6 and Equation 5.7 to compute the kernel function $K_m(G, G')$ for each graph G in the testing data set and for each graph G' in the training data set.

- Use kernel classifier and trained models to obtain prediction accuracy of the testing data set

Chapter 6

Bipartite Feature Selection for Protein-chemical Interaction Prediction

This section describes previous work on structure feature selection for protein-chemical interaction prediction. While this work does not relate to the feature relationship inference approach described here, it does deal with feature selection in a context where feature relationships are known ahead of time. In the previous work, an interaction between a chemical and protein is modeled using the tensor product between the chemical and protein feature vectors. Since each of the tensor product features is related to the originals features a regular, structured way, this structure can be exploited for efficient and accurate feature selection. This work demonstrates the value of using feature relationships (known here *a priori*) in feature selection for chemical-protein interaction prediction.

6.1 Linear Kernels for Tensor Product Feature Selection

The idea of SVM RFE [23] was adopted to select features in the feature tensor product space. Rather than directly apply RFE to select features in the tensor product space, the approach selects domain $\mathcal{A}$ and domain $\mathcal{B}$ features in the original space and hence obtains a subspace of the tensor product space. Consider an object from domain $\mathcal{A}$ represented by a set of features $A = \{a_1, a_2, ...a_n\}$ and an object from domain $\mathcal{B}$ represented by as a set of features $B = \{b_1, b_2, ...b_m\}$, the goal is to select a subset of domain $\mathcal{A}$ features features $A' \subset A$, and a subset of domain $\mathcal{B}$ features $B' \subset B$ to perform fast classification in the feature tensor product space. This goal is formalized as:

$$\arg\max_{\mathbb{A}',\mathbb{B}'} \sum_{i\in\mathbb{A}'} \sum_{j\in\mathbb{B}'} W_{i,j} \tag{6.1}$$

subject to $|A'| = q$ and $|B'| = p$ where q and p are the desired number of features selected from each domain and $W_{i,j}$ is the weight of the feature formed by $a_i \times b_j$ in the tensor product space.

Features describing a cross-domain $\mathcal{A}$-$\mathcal{B}$ interaction are generated by taking the tensor product between the domain $\mathcal{A}$ feature vector and domain $\mathcal{B}$ feature vector. An SVM model is then generated using this combined feature set, and this model gives us the weights corresponding to each $\mathcal{A}$-$\mathcal{B}$ feature pair. These weights become the matrix W and a subset of features is selected that maximizes the sum over the submatrix W'. The remaining features are then used to again train the SVM model and the process repeats until the desired number of features has been selected.

6.2 Iterative Tensor Product Feature Selection with Integer Quadratic Programming

This section shows the connection of the iterative tensor product feature selection problem to the mixed-integer quadratic programming problem (MIQP). The connection is demonstrated by rewriting the formalization of the bipartite feature selection problem as an integer quadratic programming problem.

$$\min_{x} \frac{1}{2} z^T H z \tag{6.2}$$

With $z_i \in \{0,1\}, i = 1,..,n+m$ subject to the constraints $A_1 \cdot z \leq q$ and $A_2 \cdot z \leq p$.The z vector is a column binary vector. Given n domain $\mathcal{A}$ features and m domain $mathcalB$ features, z has length $n+m$. $A_1 = [1,1,..1,0,0,...,0]^T$ is a binary column vector with a leading n number of ones followed by m zeros. $A_2 = [0,0,..0,1,1,...,1]^T$ is a binary column vector with a leading n number of zeros following by m ones. The matrix H corresponds to weights between domain $\mathcal{A}$ and $\mathcal{B}$ pairs, but the weight matrix constructed from the SVM model cannot be used directly since it is dimension $n \times m$. Instead, a matrix must be used that is $(n+m) \times (n+m)$, and embed the $n \times m$ weight matrix twice. The regions of H that correspond to within-domain $\mathcal{A}$-$\mathcal{A}$ or $\mathcal{B}$-$\mathcal{B}$ pairs are empty and add nothing to the minimization problem. The regions corresponding to $\mathcal{A}$-$\mathcal{B}$ pairs are filled with the proper weight from the SVM model. This weight is negated since the QP problem works on minimization while the goal is maximization.

If the original weight matrix between domain $\mathcal{A}$ and domain $\mathcal{B}$ features is a $n \times m$ matrix, then H is a $(n+m) \times (n+m)$ matrix. MATLAB was used for solving the quadratic programming optimizations.

For learning on homogenous data, in order to enforce selection of only a single set of features in a single domain, the MIQP problem can be simplified. Instead of mapping W into H twice, the negated W may be used instead of H and use a single constraint.

6.3 Regularized Logistic Regression for Tensor Product Feature Selection

Given features derived from two domains, $A = f_1^A..f_{m^A}^A$ and $B = f_1^B..f_{m^B}^B$ the goal is to select subsets of those features, $r^A \in \{0,1\}^{n^A}$ and $r^B \in \{0,1\}^{n^B}$ which are bit vectors where each bit represents inclusion/exclusion of a feature. Features from each domain are then combined by taking the tensor product, $A \otimes B$. This section explores the integration of feature selection into the logistic regression problem. The approach described here rests on the manipulation of r^A and r^B in the optimization problem to enforce domain space feature selection.

First, let the original L_1-regularized logistic regression optimization problem be defined as,

$$\arg\min_w \frac{1}{n}\sum_{i=1}^{n} y_i x_i^T w - log(1 + exp(x_i^T w)) + \lambda \sum_{i=1}^{m} |w_i| \tag{6.3}$$

with data, $(x_i, y_i) \in \mathbb{R}^m \times \{-1, 1\}, i = 1..n$, optimization variables $w \in \mathbb{R}^m$,$v \in \mathbb{R}$, and regularization parameter $\lambda \geq 0$. To integrate feature selection into the logistic regression problem, we must change both the basic optimization function as well as add penalization terms. With r^A and r^B as domain feature selection vectors, let $n = n^A * n^B$, $m = m^A * m^B$ and define $s \in \{0,1\}^m$, $s = r^A \otimes r^B$ as the selection vector for the corresponding tensor product space. We will transform this into a diagonal matrix z,

$$z \in \{0,1\}^{m^A} \times \{0,1\}^{m^B} \tag{6.4}$$

With $z_{i,j} = 0, i \neq j$ and $z_{i,j} = s_i, i = j$. Now the logistic regression problem can be rewritten as,

$$\arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} y_i x_i^T w - log(1 + exp(x_i^T z w)) + L_1 \tag{6.5}$$

$$L_1 = \lambda \sum_{i=1}^{m} |w_i| * s_i \tag{6.6}$$

using s and z to enforce proper selection of features in the tensor product space. Finally, regularization terms must be added to control the number of features selected in r^a and r^b. This final problem must optimize for these new variables:

$$\arg\min_{w,r^A,r^B} \frac{1}{n} \sum_{i=1}^{n} y_i x_i^T w - log(1 + exp(x_i^T z w)) + L_1 \tag{6.7}$$

$$L_1 = \lambda_1 \sum_{i=1}^{m} |w_i| * s_i + \lambda_2 \sum_{i=1}^{m^A} |r_i^A| + \lambda_3 \sum_{i=1}^{m^B} |r_i^B| \tag{6.8}$$

Note that, in this formulation, the number of features selected in each domain cannot be fixed to a specific number. Instead, the parameters λ_2 and λ_3 control the penalty for selecting more features. Values for these parameters must be selected so the corresponding terms contribute to the minimization process, yet do not dominate it.

This problem formulation is intuitive, but requires several parameters and the use of mixed data types (binary and real-valued) that must be optimized. Instead, an alternative formulation is adopted to remove the binary $r \in \{0,1\}$ vectors and replace them with $r \in \mathbb{R}^m$. We then set $w = r^A \otimes r^B$ and optimize only r^A and r^B. This problem is written as,

$$\arg\min_{r^A, r^B} \frac{1}{n} \sum_{i=1}^{n} y_i x_i^T w - log(1 + exp(x_i^T w)) + L_1 \tag{6.9}$$

$$L_1 = \lambda_1 \sum_{i=1}^{m^A} |r_i^A| + \lambda_2 \sum_{i=1}^{m^B} |r_i^B| \tag{6.10}$$

Where $w = r^A \otimes r^B$.This form of the problem is mathematically more attractive, though perhaps less intuitive. Both models have been implemented and this simpler model found to provide better performance, and hence this model was used in experimental studies.

6.4 Coordinate Descent for Regularized Logstic Regression

To solve the convex optimization problem, we implemented an algorithm that sequentially optimizes each single variable using a line search, and hence refer to it as a coordinate descent method. To optimize a feature r_j^A the gradients of the loss and penalty terms are calculated as,

$$\nabla r_j^A = \frac{1}{n} \sum_{i}^{n} \frac{e^{-y_i (x_i^T z w)} * (-y_i \sum_k^{m^B} x_{i,l} * w_l * r_k^B)}{1 + e^{-y_i (x_i^T z w)}} + \lambda_1 * \frac{r_j^A}{\sqrt{{r_j^A}^2 + \epsilon}} \tag{6.11}$$

where $l = (j-1) * m^B + k$, giving the index for w corresponding the the j'th r^A feature and k'th r^B feature. The equations are similar for a feature r_k^B. For learning on homogenous data the implementation is altered so that optimization is done to features in only one domain and mirrored in the other (identical) domain.

Chapter 7

Similarity Boosting Studies on Genome-wide Protein-chemical Interaction Prediction

The SimBoost method for genome-wide chemical-protein interaction prediction will be reviewed and then compared to competing methods and evaluated according to prediction accuracy, precision, and recall. SimBoost is also compared to Local Models in terms of tolerance to noise in the interaction data. This chapter discusses the relevant data sources, evaluation metrics, and competing methods. Experimental results are presented as well.

7.1 Background

Some of the most recent work in genome-wide protein chemical interaction prediction has focused on ensemble methods built from local models that make predictions about a small region of interaction space. This approach is well motivated, since it is unlikely that a single model can generalize to make predictions regarding many diverse protein families and chemotypes. There are several issues with this current work on interaction prediction, and the research proposed in this section aims to address each of these issues. Boosting is proposed as a generable ensemble learning framework, under which two modifications are proposed: 1) adjust sample reweighting based on sample similarity as well as misclassification, and 2) weight model predictions based on sample similarity as well, so that models make predictions only for samples most similar to previous samples that were correctly classified. This section first explores the connection between local models and boosting, then outlines the technical details required to modify boosting for the genome-wide interaction prediction setting.

The approach proposed in [5] using local models strongly resembles the use of ensemble methods in which a number of weakly performing classifiers are organized in such a way that their predictions can be combined into a strongly performing classifier. In the previous approach, a local model for each unknown protein-chemical interaction is built using only known samples that share a protein or chemical with the unknown interaction. This method is analogous to an ensemble approach where each model makes a prediction for only a single sample, and is built using only the most relevant data to that sample. For genome-wide protein-chemical interaction prediction, local models are useful because the chemical-protein interaction space is large and built from a diverse number of protein families as well as chemotypes. Partitioning that space into more homogenous segments and building specialized models for each segment is a much more reasonable approach than attempting to build a single model to make predictions predictions across all interactions. The drawback of using local models is that each model is *too* specialized, and broader similarities between samples are ignored during both learning and prediction. Further, known ensemble learning approaches have not been explored in this context.

Recent work in genome-wide protein chemical interaction prediction has focused on ensemble methods built from local models that make predictions about a small region of interaction space. This approach is well motivated, since it is unlikely that a single model can generalize to make predictions regarding many diverse protein families and chemotypes. There are several issues with this current work on interaction prediction, and the research described in this section aims to address each of these issues. Boosting is proposed as a generable ensemble learning framework, under which two modifications are made: 1) adjust sample reweighting based on sample similarity as well as misclassification, and 2) weight model predictions based on sample similarity as well, so that models make predictions only for samples most similar to previous samples that were correctly classified.

The boosting algorithm, following AdaBoost [26] is as follows:

Boosting works by constructing a series of models, each weighted so that it favors accurate predictions regarding samples that have been misclassified by previous samples. The final predictions are then made by a voting arrangement where each model contributes a vote weighted by itś overall prediction accuracy. Here the similarity between boosting and local models can be seen, in that they both rely on a series of specialized models that each make predictions favoring only specific samples. The difference between boosting and local models is that boosting incorporates all individual model predictions into an overall set of predictions, while local models do not weight samples in response to misclassification rates. The drawback to boosting is that is does not pay attention to sample similar-

Algorithm 1 AdaBoost

```
Require: data $(x_1, y_1), ..., (x_n, y_n)$
    class labels $x_i \in X, y_i \in Y = \{-1, +1\}$
1: First, initialize sample weights $w_i = \frac{1}{n}, i = 1, ..., n$.
2: for $t = 1$ to $T$: do
3:     Fit a classifier $G_t(x) : X \to \{-1, +1\}$ to the training data with weights $w_i$.
4:     Compute the sample-weighted error:
       $err_t = \sum_{i=1}^{n} w_i I[y_i \neq G_t(x_i)]$
5:     Compute the classifier weight, using $err_t$ as computed in the previous step:
       $\alpha_t = \frac{1}{2} log \frac{1 - err_t}{err_t}$
6:     Finally, update the sample weights:
       $w_i \leftarrow \frac{w_i \; exp(\alpha_t I[y_i \neq G_t(x_i)])}{Z_t}$
       where $Z_t$ is a normalization factor.
7: end for
8: The final classifier is output as:
   $G(x) = sign(\sum_{t=1}^{T} \alpha_t G_t(x))$
```

ity when computing sample weights, and also does not restrict model predictions to only the relevant samples. The remaining material in this section describes boosting variations and relevant details.

7.2 Methodology

Boosting is a well known ensemble learning framework that is usable with many types of classifiers, and which can be robust to overtraining and provide high generalization ability. Boosting works by constructing a series of models, each weighted so that it favors accurate predictions regarding samples that have been misclassified by previous samples. The final predictions are then made by a voting arrangement where each model contributes a vote weighted by itś overall prediction accuracy. Here the similarity between boosting and local models can be seen, in that they both rely on a series of specialized models that each make predictions favoring only specific samples. The difference between boosting and local models is that boosting incorporates all individual model predictions into an overall set of predictions, while local models do not weight samples in response to misclassification rates. The drawback to boosting is that is does not pay attention to sample similarity when computing sample weights, and also does not restrict model predictions to only the relevant samples.

7.2.0.3 Similarity-based Sample Weighting

In order to modify boosting to take advantage of sample similarity, the sample weighting step must be modified to incorporate similarity alongside miss classification. By averaging the sample weights according to similarity, the effect of misclassified samples is reduced. Given n samples, let K be the n by n matrix of normalized sample similarities such that $K(i, j)$ is the similarity between samples i and j, where $0 \leq K(i, j) \leq 1$ and K is symmetric, with $K(i, j) = 1$ indicating strong similarity and $K(i, j) = 0$ indicating weak similarity. The sample weight update equation can then be redefined as:

$$w'_i \leftarrow \frac{\sum_{j=1}^{n} K(x_i, x_j)\ w'_j}{Z_t}\ exp(-\alpha_t I[y_i \neq G_t(x_i)]) \tag{7.1}$$

So that w'_i is the similarity-weighted sum of all sample weights, and Z-t is a normalization factor. Note that when K is the identity matrix the equation for w'_i reduces to the previous reweighting equation for w_i.

The choice of K can be made arbitrarily, computed, or learned. In the case where samples are known to be organized into discrete groups $C_1, ..., C_m$, and K can be chosen so that for each C_i and a pair of samples (a, b), $K(a, b) = 1$ if a, b are members of C_i and zero otherwise, with each a, b belong to only one C_i. In this case each sample in the group will be weighted the same.

If explicit feature vectors for samples are available, the similarity matrix K can be computed according to a distance metric such as Euclidean distance, or they are not available a similarity function such as a kernel function can be used. Using different similarity/kernel functions will embed samples into different vector spaces and hence induce different sample weightings.

7.2.0.4 Similarity-based Prediction Weighting

The boosting equation for building the final classifier uses a weighted sum of the local classifiers, but not only does each classifier contribute equally to each prediction, but each classifier is also weighted by its *global* accuracy on the entire data set. Both of these cases contradict the intuition that each local model is specialized for a subset of samples and thus should make predictions regarding only those samples.

Like the modifications to boosting for sample weighting, modifying the boosting prediction function uses a matrix K of similarities between samples, $0 \leq K(i,j) \leq 1$ and $K(i,j) = K(j,i)$, for all $i,j = 1,...,n$. A kernel $K(x,x_i)$ between some unknown sample x and previous samples x_i is also required. The loss function terms can be redefined to parameterize err_t and α_t so that the final classifier $G(x)$ incorporates the similarity between a new sample x and previously well classified samples:

$$err'_t(x) = \sum_{i=1}^{n} K(x, x_i)\ w_i\ I[y_i \neq G_t(x_i)] \tag{7.2}$$

$$\alpha'_t(x) = \frac{1}{2} ln \frac{1 - err'_t(x)}{err'_t(x)} \tag{7.3}$$

$$G'(x) = sign(\sum_{t=1}^{T} \alpha'_t(x) G_t(x))) \tag{7.4}$$

where the term $err'_t(x)$ weights model predictions to favor models that perform well on samples most similar to the unknown sample x. The normal loss function is weighted by the similarity term $K(x, x_k)$ in order to minimize the contribution of samples that are dissimilar to x. Note that when $K(x, x_k) = 1$ for all $k = 1, ..., n$, this equation reduces to the original err_t. The w_i here is the same w_i from the original equations, computed during the last boosting iteration.

7.3 Data Sets

This section describes the characteristics of the data sets used to evaluate general classification performance, as well as tolerance to interaction label noise.

7.3.1 Noise Tolerance Data Sets

In order to measure the tolerance of the SimBoost and Local Model classifiers to noise in the training labels, a gold standard data set of high quality with known positive and negative interactions is required. The authors of Local Models evaluated their method on such a collection of data [5] and have made this data set available along with the protein and chemical similarities used in their evaluation studies. In order to compare our method to Local Models we have elected to use this data as well. The data is not ideal for an overall evaluation since it is divided into 4 distinct classes of interactions involving enzymes, nuclear receptors, g-protein coupled receptors (GPCRs) and ion channels. The sizes and characteristics of these data sets are given in table 2. Chemical compounds for this data were taken from the DRUG and COMPOUND areas of the KEGG LIGAND database [33] and protein amino acid sequences were taken from the KEGG GENES database. Chemical similarity was determined using graph alignment and the protein similarities determined using sequence alignment. Further details regarding data are available in the Local Models paper.

Table 7.1. Number of total, positive and negative samples in each data set.

Data Set	# Samples	# Positives	# Negatives	# Proteins	# Chemicals
Nuc. Receptors	1404	90	1314	26	54
GPCRs	21185	635	20550	95	223
Ion Channels	42840	1476	41364	204	210
Enzymes	295480	2926	292554	664	445
Total	360909	5127	355782	989	932
Selected	3200	1600	1600	861	502

It is evident that the interaction space is very sparse, that is, very few positive interactions compared to negative interactions. Many classifiers are dependent on the class balance of a data set, and given sparse data can easily devolve into a trivial classifier predicting all samples as members of the dominant class. To combat this difficulty, some classifier parameters can be tuned to avoid this behavior. The parameter tuning process often requires another level of cross-validation, and hence can be time consuming to find the best parameters to ensure a fair comparison between methods. In order to avoid the complications that a class bias introduce, we have randomly sampled a set of negatives equal to the number of positives. This random selection was done 5 times generating 5 subsets of each data set, each with a different random selection of negative protein-chemical pairs. In order to measure tolerance to noise, the interaction labels for each of the data subsets were set randomly in increasing amounts ranging from 0% (no noise) to 100% (all labels are random) in increments of 10%, for 11 total experiments for each subset. Each of the data subsets was divided into half for training, and the other half for testing, and various levels of noise was introduced into the training data, creating 11 experimental results for each of the 5 subsets in each data set. The results are averaged over the 5 random subsets at each noise level, for all four data sets and both methods.

7.3.2 General Performance Data Sets

In order to evaluate interaction prediction on some typical interaction data, we have randomly sampled a number of interactions from the DrugBank [66] repository and assembled them into several data sets. A number of sets ranging from 100-400 samples and 100-200 features were collected and the general performance of SimBoost and related methods was evaluated. In each dataset, a balanced number of positive and negative samples were randomly selected. Because Drugbank offers only a list of known positive interactions and does not provide a list of known negative interactions, a set of synthetic negative samples was created by first selecting protein-chemical pairs known to interact, and then substituting another protein/chemical for the original one, such that feature similarity between them is low.

For each dataset, we created 10 randomized trials. For each trial, 10-fold cross-validation was used to evaluate each method's performance, requiring 10 experiments each with one 10% segment of the data used for testing with the remaining 90% used for training. This or a total of 100 sets experimental results per method per data set, which are averaged into the final measures of accuracy, sensitivity, and specificity. For classifiers requiring parameter optimization, such as SVM, internal 5-fold cross-validation on the training data is used to select the best parameter. Details on parameter selection for various methods are given below.

Features were extracted using the training data during each experiment. For chemicals, frequent subgraph features [28] were generated, and frequent k-mers were generated for proteins. In these experiments, $k = 4$ was used to generate the k-mers, which were then filtered by frequency. The frequency threshold for both subgraphs and k-mers was chosen so that the desired number of features was obtained, as well as a balanced number of protein/chemical features. For instance, if 200 features is desired, a frequency threshold would be chosen such that 100 frequent subgraphs were obtained as well as 100 frequent k-mers. SimBoost and SVM both use a linear kernel to compute similarity between samples based on these features.

Tables 3 and 4 show the number of samples and features in each data set, as well as the number of unique proteins and chemicals.

7.4 Experimental Protocols

In order to establish the utility of SimBoost on protein-chemical interaction data, a series of interaction prediction data sets were constructed as described above. Experimental evaluation of this data was performed using the Weka software. The SimBoost and Local Models methods were implemented as Weka modules in Java and compared to other Weka implementations of competing methods.

Table 7.2. Number of samples and features of each data set.

Index	# Samples	# Features
1	100	100
2	150	100
3	150	150
4	200	100
5	200	200
6	300	150
7	300	300
8	400	100
9	400	150
10	400	200

Table 7.3. Number of unique chemicals and proteins in each data set.

Index	# Unique Chem.	# Unqiue Prot.
1	36	28
2	45	34
3	52	43
4	49	37
5	76	60
6	64	51
7	113	80
8	48	40
9	70	52
10	87	65

This section discusses some details on experimental methodology regarding parameter selection for various classifiers, evaluation metrics, and competing methods.

7.4.1 Parameter Evaluation

SimBoost is dependent on two parameters, in addition to the features/kernels selected: the base learner used, and the number of iterations/models. A good starting base learner would be naïve Bayes, due to its lack of a parameter that will need tuning and strong theoretical performance, but other base learners will be evaluated as well. The parameter that controls number of iterations/models will have significant impact on performance as well, and a range of values should be evaluated on a number of data set sizes in order to establish a correlation between the number of data samples and number of models needed to effectively discriminate between samples. For SimBoost and AdaBoost, the number of iterations was fixed at 10, and both used naïve Bayes as the base classifier. The naïve Bayes method has no parameters to optimize. For the SVM parameter selection, as mentioned above, within each experimental cross-validation fold, a smaller internal 5-fold cross-validation experiment was performed to select the best C parameter.

7.4.2 Evaluation Metrics

For the general performance studies, comparisons will be made primarily on the basis of: interaction prediction accuracy, sensitivity, and specificity. These statistics will be recorded and averaged over all cross-validation trials in these experiments. Accuracy is defined as $(TP + TN)/S$ where TP is number of true positives, TN is number of true negatives and S is the total number of testing samples. Precision $(TP/(TP+FP))$ and recall $(TP/(TP+FN))$ are defined also in terms of FP, number of false positives and FN, the number of false negatives.

For the label noise tolerance studies, accuracy will be reported as well.

7.4.3 Comparison Methods

For general performance comparisons, the proposed SimBoost method will be compared to several competing methods. First, it will be compared to the typical boosting algorithm as a baseline comparison. It will also be compared to a single-model (non-ensemble) version of the base classifier used. Finally, SimBoost is compared to popular global classifiers such as SVM, as well as the local classifiers proposed in previous work [5].

The noise tolerance of SimBoost will be evaluated in comparison to the Local Models, using the same data set as well as similarity kernel used by the Local Models authors in their evaluation.

7.5 Label Noise Tolerance Results

The following tables show the performance of SimBoost and Local Models for the gold standard data sets, as the amount of noise in the interaction labels is increased.

For the Nuclear Receptor data set, the results are ambiguous as the accuracy and AUC are unstable as the amount of label noise is increased. This effect may be due to the very small size of the data set. The second, larger data set with GPCRs is much more clean and stable, showing a clear deterioration of the Local Models approach as label noise is introduced. This is contrasted with a very slow decline in the performance of the SimBoost method, which has strong performance up to 70 and 80% label noise. Both methods perform around 50% accuracy when the amount of noise is increased to 100%, as expected. The results for the Ion Channel data set are not as clear as the GPCR results, largely because the performance for Local Models starts off much lower and so the decline in performance as noise is introduced is not as steep. The results for the Enzyme data is a mix between those from the other data sets. The accuracy for Local Models is lower than SimBoost, and declines accordingly, while the accuracy for SimBoost declines relatively slowly while most of the noise is introduced, and then drops sharply at the highest levels of label noise.

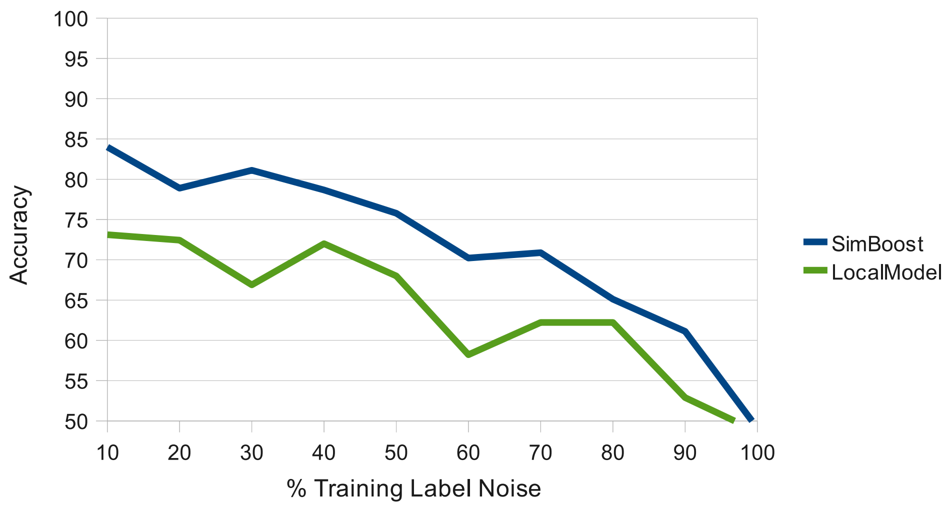

Figure 7.1. Prediction accuracy for Nuclear Receptor data in response to training label noise.

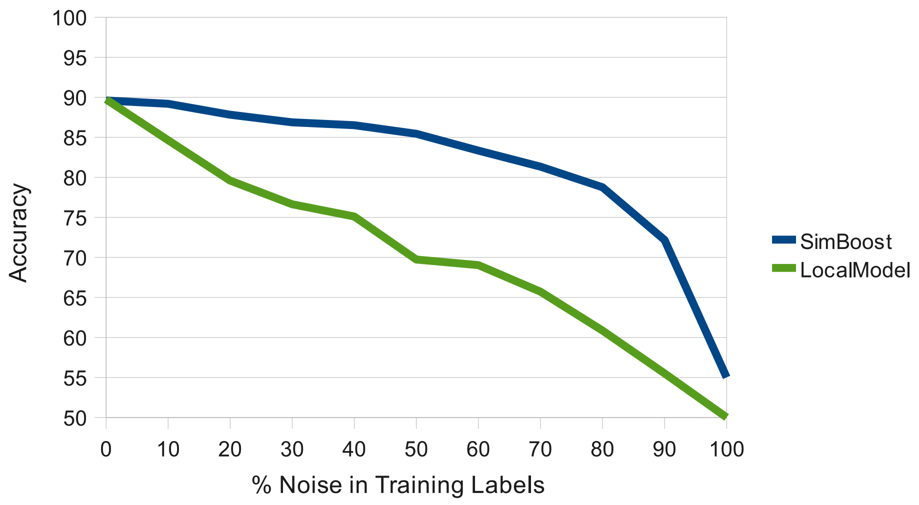

Figure 7.2. Prediction accuracy for GPCR data in response to training label noise.

Table 5 shows the performance of SimBoost, Local Models, and several other methods at the highest level of training label noise tested, 90%. Showing that in all cases, the accuracy for SimBoost is the best. It is notable that on two of that

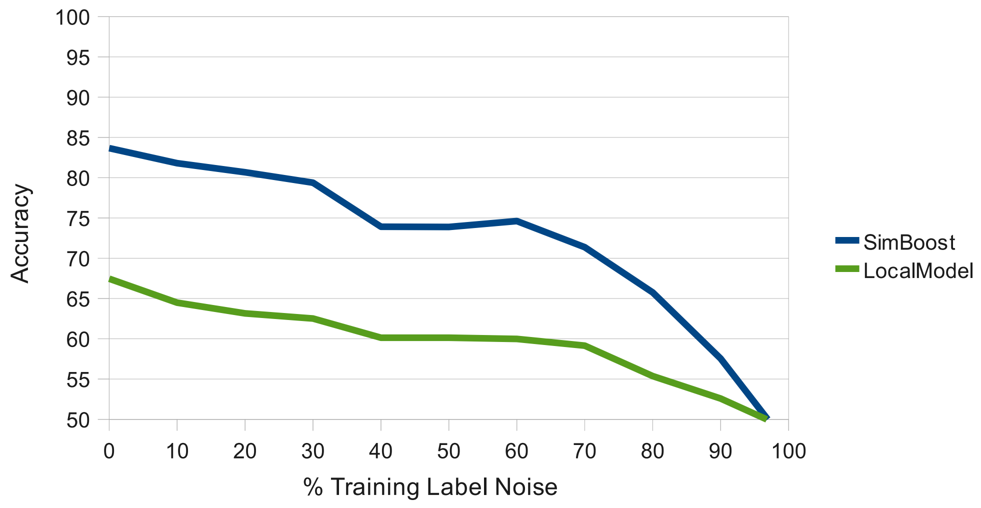

Figure 7.3. Prediction accuracy for Ion Channel data in response to training label noise.

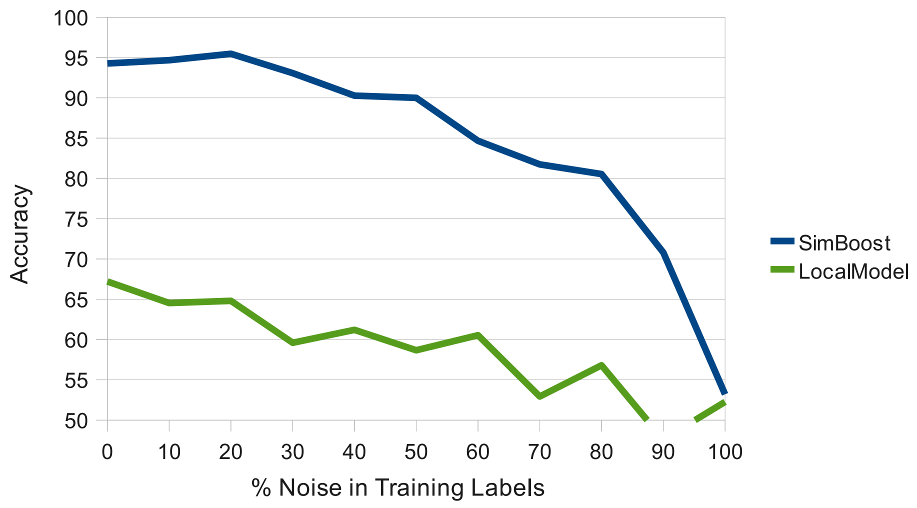

Figure 7.4. Prediction accuracy for Enzyme data in response to training label noise.

data sets, SimBoost is able to achieve 70% accuracy when 90% of the training data labels are noise.

Table 7.4. Accuracy of several methods at 90% training label noise. Best result for each data set is marked in bold.

Set	SimBst	AdaBst	SVM	Loc.Model
Nuc. Receptor	**61.11**	55.55	56.22	52.89
GPCR	**72.18**	61.30	57.01	55.51
Ion Channel	**57.55**	54.79	53.58	52.58
Enzyme	**70.80**	59.07	51.47	47.60

7.6 General Performance Results

The following tables and figures present results for experiments comparing the general performance of SimBoost to a variety of competing methods. Table 7.5 presents a comparison of classification accuracy for each of the methods across the ten datasets.

Table 7.5. Accuracy for 10 drugbank datasets comparing SimBoost to competing methods, along with average accuracy for each method. Results marked in bold are the best for each data set.

Set	SimBst	AdaBst	Nai.Bayes	SVM	Loc.Model
1	**65.3**	57.1	61.1	60.0	61.2
2	**59.9**	54.8	58.8	54.9	58.2
3	**59.9**	54.6	57.3	56.4	56.3
4	**61.1**	59.7	58.1	58.9	59.4
5	**59.7**	56.7	58.1	56.3	56.5
6	**57.5**	55.5	56.4	57.3	56.2
7	**58.1**	57.3	57.3	53.3	56.4
8	**57.5**	56.4	56.6	57.3	56.9
9	57.1	56.1	57.2	**58.3**	57.0
10	58.5	57.9	56.2	**58.6**	56.9
Avg.	**59.5**	56.6	57.7	57.2	57.5

The results obtained show that the SimBoost method is significantly better than AdaBoost at the 1% level in 6 of the 10 datasets, and significant at the 10% level for one more. Significance was determined using 2-way ANOVA. SimBoost is also more accurate than naïve Bayes at the 1% level in 9 of the datasets, and worse in one. Finally SimBoost claims a higher accuracy than SVM with a 1%

Table 7.6. Precision for 10 drugbank datasets comparing SimBoost to competing methods, with averages for each method. Results marked in bold are the best for each data set.

Set	SimBst	AdaBst	Nai.Bayes	SVM	Loc.Model
1	**66.6**	58.6	60.8	60.7	60.7
2	**62.1**	55.0	59.7	55.6	58.7
3	**61.6**	55.7	58.1	57.3	57.0
4	**61.9**	60.3	57.4	59.2	58.7
5	**60.0**	56.6	57.6	56.4	56.4
6	57.2	55.7	55.7	**57.4**	55.7
7	**58.1**	57.5	56.5	52.9	55.7
8	**57.6**	56.9	56.2	56.6	56.6
9	56.9	55.8	56.7	**58.6**	56.7
10	**59.0**	58.2	56.0	58.9	56.6
Avg.	**60.1**	57.0	57.4	57.3	57.2

confidence level for 6 of the data sets, and is worse than SVM in two of the sets.

Results for precision are similar to those for accuracy, although compared to AdaBoost, SimBoost is has significantly better precision for only 4 of the sets at 1% and one set at 10%. In the other cases, SimBoost is better but not significantly so. Again, SimBoost is significantly better than naïve Bayes at the 1% level for 9 of the 10 data sets. Compared to SVM, SimBoost is significantly better in 6 of the datasets at 1% and one set at 10%; SVM has better precision in two of the data sets.

The results for comparing methods in terms of recall are somewhat different than for accuracy and precision. SimBoost is significantly better than AdaBoost in 8 of the datasets, all at the 1% level, and worse in only two. Naïve Bayes, however, was better than SimBoost in recall for all data sets. The results for SVM are somewhat mixed: SimBoost is again better in 4 of the data sets at 1% significance, and better in another data set at the 5% level, although SimBoost is worse for three of the data sets.

Table 7.7. Recall for 10 drugbank datasets comparing SimBoost to competing methods, with averages for each method. Results marked in bold are the best for each data set.

Set	SimBst	AdaBst	Nai.Bayes	SVM	Loc.Model
1	65.6	53.0	**67.4**	58.8	63.0
2	55.9	52.0	58.9	55.7	**61.4**
3	58.6	51.1	60.5	55.2	**61.1**
4	58.7	59.8	63.9	61.4	**64.3**
5	61.0	55.3	**64.2**	54.7	62.6
6	61.3	53.6	**65.9**	56.0	63.4
7	60.0	53.9	**63.8**	51.7	62.7
8	60.5	54.5	**63.5**	62.1	62.7
9	56.7	59.4	**62.0**	56.6	59.9
10	58.8	56.8	61.1	59.6	**61.9**
Avg.	59.7	54.9	**63.1**	57.1	62.3

Table 7.8. Number of data sets where SimBoost performs significantly better (at the 5% confidence level using ANOVA), for each comparison method and statistical measure.

Statistic	AdaBoost	NaïveBayes	SVM	LocalModel
Accuracy	6	9	7	9
Precision	4	9	7	9
Recall	8	10	4	10

Overall, p-values generated by performing 2-way ANOVA across all datasets indicates that the SimBoost method is better than compared methods with greater than 99% confidence (p-values were less than 10^{-10}).

Chapter 8

Clustered Boosting Studies on Genome-wide Protein-chemical Interaction Prediction

This chapter describes in detail the experimental studies performed to evaluate the clustered boosting method described previously. This chapter begins with a discussion of background and methodology is provided. Next, the data sources and representations for chemicals, proteins, and interactions is presented. Following that, the experimental protocols used are discussed along with the feature selection, parameter selection, and evaluation measures. Finally, the experimental results are presented and discussed. General performance of clustered boosting on genome-wide protein-chemical interaction prediction is evaluated compared to competing methods. The performance of clustered boosting is evaluated on both balanced and unbalanced selections of data. The response of clustered boosting to variations in the input parameters is also examined. A comparison of clustered boosting to SimBoost is performed in terms of accuracy and computation time.

8.1 Background

Similarity boosting performs an implicit specialization of base models using kernel similarity. The specialization of each model is due to the smooth reweighting of samples according to misclassification and spatial proximity over boosting iterations. While this approach produces a model with some attractive qualities, in particular the tolerance to label noise, it's limitation is in the computational effort required to perform a summation for every sample over kernel similarities between all other samples.

In contrast to this implicit specialization process, it is also possible to explicitly specialize models. Instead of using real value sample weights to specialize models toward a specific subset of samples, a binary function can be used to indicate inclusion/exclusion of samples in one of an ensemble of models. A computationally efficient approach is to assign samples into disjoint partitions, and assign a model to each partition. If samples are allowed to be included in more than one model, the total number of samples used to train with will increase. By assigning each sample to only one partition at a time, the total number of samples used to train over all models will be equal to the number of samples used to train a single global model. This approach may admit some computational gains over using a global model if the base model used has time complexity worse than linear with respect to the number of training samples.

Clustering is a well-studied area of machine learning designed to partition samples into disjoint regions. One of the earliest and most straightforward clustering methods is k-means. In k-means clustering a number of cluster means or centroids are selected and samples are assigned to the closest cluster. Cluster means are then updated to reflect the center of mass for the assigned samples. The sample assignment is performed again and the process continues for a set number of iterations or until convergence.

Clustering is an unsupervised method, and can be used for classification by assigning the same class labels to unknown samples as the majority class of labels found in the corresponding clusters. In the case of genome-wide protein-chemical interaction prediction imay not be reasonable to assume that the interaction space could be easily partitioned into disjoint regions of identical labels. More often the case is that non-interacting samples appear throughout the entire interaction space although the distribution may be uneven. Positively interacting samples then accrue in certain regions of enrichment, overlapping to various degrees with negative samples.

While it may not be possible to directly partition the genome-wide interaction space into disjoint regions of identically labeled samples, it is not unreasonable that a supervised model could be trained to accurately predict the class labels for a cluster of interaction samples. An ensemble of models may then be learned, each corresponding to a cluster of samples. In this way the ensemble of models is directly specialized, each to a different region of the interaction space. This ensemble can then be used to make predictions for unknown samples by following the boosting framework and building a weighted additive model from the ensemble. This approach leads to the clustered boosting method described in the following sections.

Algorithm 2 K-Means Clustered Learning

```
Require: training set $\{(x_i, y_i)\}_{i=1}^n$,
    base learner $h$
    number of clusters $C$
1: Initialize cluster means $\mu_1, ..., \mu_C$
2: while Clusters have not converged do
3:     Assign samples to the closest cluster for each $c = 1, ..., C$:
       $S_c = \{x_i : ||x_i - \mu_c|| \leq ||x_i - \mu_j|| \text{ for all } j = 1, ..., C\}$
4:     for $c = 1$ to $C$ do
5:         Train a base learner $h_c$ on the set $\{(x_i, y_i) | x_i \in S_c\}$
6:         Update the cluster mean as:
           $\mu_c = \frac{1}{\sum_{x_i \in S_c} I(h_c(x_i) = y_i)} \sum_{x_i \in S_c} x_i I(h_c(x_i) = y_i)$
7:     end for
8: end while
9: Return clusters and models.
```

8.2 Methodology

This section describes new work on boosting with clustering for genome-wide protein chemical interaction prediction.

One option to incorporate the decision function into the clusters is the use of supervised or semi-supervised clustering. This section describes a learning algorithm that finds a set of clustered models by iterative moving cluster centroids toward regions of better predictive performance. The key insight is to cluster samples such that the samples in each cluster can be accurately predicted by a model learned just for that cluster. The major algorithmic difference is that in the cluster update step, cluster centroids are moved to the region of space enriched with accurately predicted samples. A k-means algorithm based on this approach is outlined.

Given random initial clusters, clustered learning begins with a set of models that are similar, but with some slight variation due to the random assignments.

Through iteration, those slight differences are magnified by retraining the model on data it was better at predicting. In this way clustered learning progressively refines the clusters to maximize the accuracy of each model.

This formulation bears a resemblance to AdaBoost, with a series of models constructed in each iteration. This connection can be made clear by defining a classification algorithm based on clustered learning, but combining each of the base models in the same fashion as AdaBoost. Some modifications are required for this approach. First, since the center of mass for clusters changes in every iteration along with the cluster base model, clusters may not converge, and hence a limit on the number of iterations is required. Another issue is weighting of the individual cluster models and how their predictions are combined to create a unified model.The clustered boosting classifier with these modifications is given as follows.

The final model also bears a resemblance to tree boosting, in that it is an additive model where each base model is also additive. In both methods, the inner summation is used as a convenience for selection of a lead node in the case of tree boosting, or a cluster in the case of clustered boosting.

One practical issue is that in some circumstances clusters may contain zero samples. This can arise if a cluster centroid is surrounded closely by other cluster centroids. When this situation occurs the cluster is removed. However a threshold for removal is required since the cluster samples may be used for cross-validation to select parameters for a cluster classifier. In this case the number of samples in the cluster must exceed the number of cross-validation folds, and hence the threshold with which to eliminate a cluster must be high enough so this case will not occur. This threshold is not meant as a model parameter to tune in order to

Algorithm 3 Clustered Boosting

```
Require: training set $\{(x_i, y_i)\}_{i=1}^n$,
    base learner $h$
    number of clusters $C$
    number of iterations $T$
 1: Initialize cluster means $\mu_1^0, ..., \mu_C^0$
 2: for $t = 1$ to $T$ do
 3:   Assign samples to the closest cluster for each $c = 1, ..., C$:
      $S_c^t = \{x_i : ||x_i - \mu_c^{t-1}|| \leq ||x_i - \mu_j^{t-1}||$ for all $j = 1, ..., C\}$
 4:   for $c = 1$ to $C$ do
 5:     Train a base learner $h_c^t$ on the set $\{(x_i, y_i)|x_i \in S_c^t\}$
 6:     Calculate the well-classified, kernel-weighted center of mass:
        $\hat{\mu_c^t} = \frac{1}{\sum\limits_{x_i \in S_c^t} I(h_c^t(x_i) = y_i)} \sum\limits_{x_i \in S_c^t} x_i I(h_c^t(x_i) = y_i) K(\mu_c^t, x_i)$
 7:     Move the cluster centroid to the center of mass:
        $\mu_c^t = \hat{\mu_c^t}$
 8:     Calculate the classifier error rate for the new clusters:
        $\epsilon_c^t = \frac{1}{|S_c^t|} \sum\limits_{x_i \in S_c^t} I(h_c^t(x_i) = y_i)$
 9:     Calculate the classifier weight for each cluster:
        $\alpha_c^t = \frac{1}{2} log \frac{1 - \epsilon_c^t}{\epsilon_c^t}$
10:   end for
11: end for
12: Final model is given as:
    $F(x) = sign(\sum\limits_{t=1}^{T} \sum\limits_{c=1}^{C} \alpha_C^t h_c^t(x_i) I(x \in S_c^t)$
```

adjust model performance, but rather a mechanism for handling an error case.

8.3 Data Sources

Data for the experimental studies presented here is obtained from a variety of sources. Three kinds of data are required: chemical structures, protein sequences, and protein-chemical interactions. The data sources used to obtain this information are described.

The authors of the local models approach evaluated their method on a collection of data [5] and have made this data set available along with the protein and chemical similarities used in their evaluation studies. In order to compare our method to local models we have elected to use this data as well. Yamanishi et al. originally assembled this data set in 2008 [71], and the authors stated they recovered the interactions from KEGG BRITE, BRENDA, SuperTarget, and DrugBank and use this data as a 'gold standard' reference set. In order to perform a comparison using this data set, we have randomly selected a balanced set of 3200 interaction samples, with exactly 1600 positives and 1600 negatives. The sizes and characteristics of these data sets are given in the previous chapter.

Chemical structures were obtained in SDF format from KEGG DRUG and COMPOUND databases. Protein sequences were recovered from KEGG GENES database. For the preliminary studies, explicit features were not available for some of the data and in this case the pre-computed similarities were provided with the data used in these experiments. For the preliminary studies, chemical similarity was determined using graph alignment and the protein similarities determined using sequence alignment. Further details regarding data are available in the Local Models paper [5].

8.4 Extracted Features

For the gold standard studies extraction of general protein and chemical features was performed. The PROFEAT [44] server was used to computer a large number of features for both proteins and chemicals. PROFEAT offers an extensive selection of protein features, and we generated over 1000 different features of various types. The following table outlines and describes these features, along with a table describing protein classes as used by some features calculations.

Table 8.1. Features Calculated by PROFEAT

Group	Name
1	Amino acid composition
2	dipeptide composition
3.1	Normalized Moreau-Broto autocorrelation
3.2	Moran autocorrelation
3.3	Geary autocorrelation
4.1	Composition
4.2	Transition
4.3	Distribution
5.1	Sequence-order-coupling numbers
5.2	Quasi-sequence-order
6	pseudo-amino acid composition
7	amphiphilic pseudo-amino acid composition
8	topological descriptors at atomic level
9	total amino acid properties

Table 8.2. Amino Acid Attributes and Classes

Property	Class 1	Class 2	Class 3
Hydrophobicity	polar	neutral	hydrophobic
Normalized van der Waals volume	0-2.78	2.95-4.0	4.03-8.08
Polarity	4.9-6.2	8.0-9.2	10.4-13.0
Polarizibility	0-1.08	0.128-0.186	0.219-0.409
Charge	positive	neutral	negative
Secondary structure	helix	strand	coil
Solvent accessibility	buried	exposed	immediate

Generation of protein features with PROFEAT used the default settings. For chemical features, PROFEAT generates only the topological descriptors.

8.5 Experimental Protocols

In order to establish the utility of clustered boosting on protein-chemical interaction data, a series of interaction prediction data sets were constructed as described above. Experimental evaluation of this data was performed using the Weka software. The clustered boosting and local models methods were implemented as Weka modules in Java and compared to other Weka implementations of competing methods. This section discusses some details on experimental methodology regarding parameter selection for various classifiers, evaluation metrics, and competing methods.

For each dataset, we created 10 randomized trials. For each trial, 10-fold cross-validation was used to evaluate each method's performance, requiring 10 experiments each with one 10% segment of the data used for testing with the remaining 90% used for training. This process was repeated on 5 trials with randomly ordered samples for a total of 50 sets experimental results per method per data set, which are averaged into the final measures of accuracy, precision, recall, and F-measure. For classifiers requiring parameter optimization, such as SVM, internal 10-fold cross-validation on the training data is used to select the best parameter.

8.6 Parameter Selection

Clustered boosting is dependent on two parameters, in addition to the features/kernels selected and the base classifier used: the number of clusters, and the number of iterations. As shown in previous work, naïve Bayes is a reasonable base classifier, due to its lack of a parameter that will need tuning and strong theoretical performance, but other base learners have been evaluated as well. The parameters that control number of clusters/iterations will have significant impact on performance as well. For clustered boosting, the number of clusters and iterations was selected using grid search over results from 10-fold cross-validation on the training data. The parameters were both selected from the set $\{5, 15, 25\}$. For AdaBoost, the number of iterations was selected in the same way using 10-fold cross-validation on the training data, with the parameter also chosen from the set $\{5, 15, 25\}$. In many comparisons the naïve Bayes as the base classifier, however SVM is also used in the general performance comparison. The naïve Bayes method has no parameters to optimize. SVM parameter selection is performed in the same was as described for clustered boosting and AdaBoost, with the C parameter selected from the set $\{10^{-1}, 10^{0}, 10^{1}\}$.

8.7 Feature Selection

Feature selection using training data was performed in order to filter out features that have low utility. Features were ranked using information gain (IG) between each feature vector and the class labels. A low cutoff score of 0.01 was used to eliminate the most uninformative features. The distribution of IG scores calculated for all features is heavily skewed toward low scores, with 556 features having zero calculated IG. The cutoff of 0.01 was chosen to eliminate these low utility features while retaining as many features with some utility as possible. The following figure shows the IG score distributions. Note that score rankings and distributions for each specific experiment will deviate from these results slightly as they are calculated for the training data only.

8.8 Evaluation Metrics

For the general performance studies, comparisons are made primarily on the basis of: interaction prediction accuracy, sensitivity, and specificity. These statis-

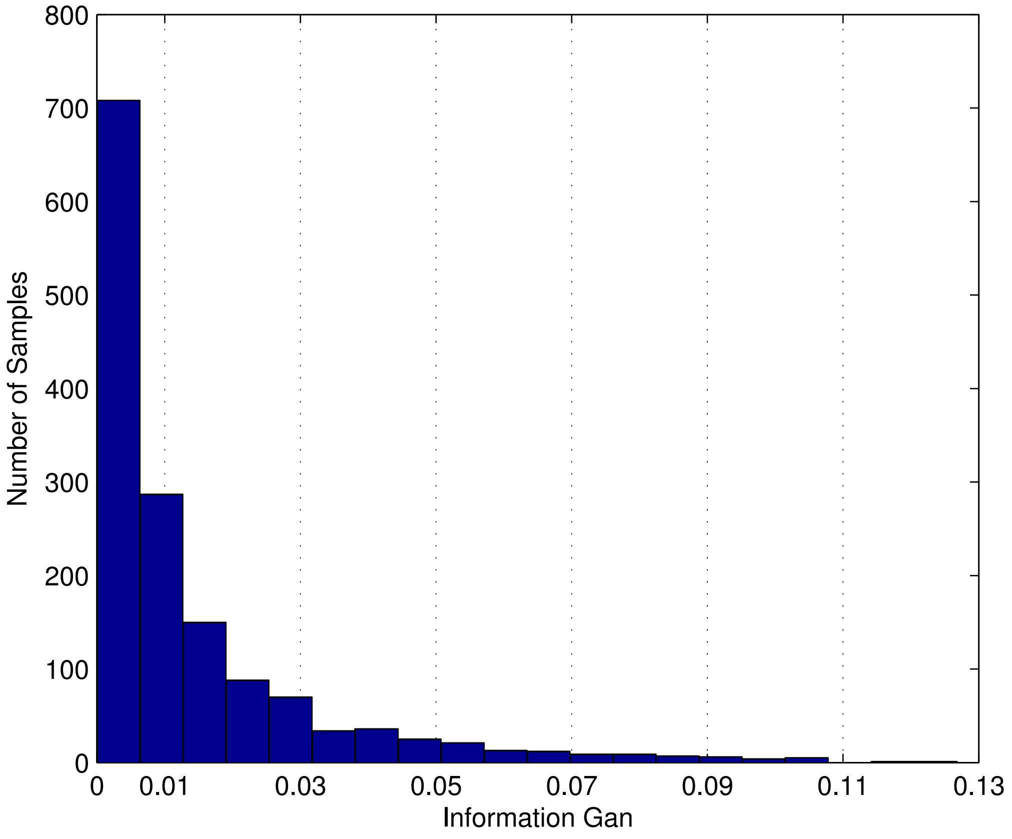

Figure 8.1. Distribution of information gain scores calculated for features using entire data set.

tics will be recorded and averaged over all cross-validation trials in these experiments. Accuracy is defined as $(TP+TN)/S$ where TP is number of true positives, TN is number of true negatives and S is the total number of testing samples. Precision $(TP/(TP+FP))$ and recall $(TP/(TP+FN))$ are defined also in terms of FP, number of false positives and FN, the number of false negatives. The F-measure is reported in most studies as well, which is the harmonic mean of precision and recall: $(2*P*R)/(P+R)$, where P and R are precision and recall, respectively.

8.9 Comparison Methods

For general performance comparisons, clustered boosting will be compared to several competing methods. First, it is compared to the typical boosting algorithm as a baseline comparison. It will also be compared to a single-model (non-ensemble) version of the base classifier used. Finally, clustered boosting is compared to the local models approach described by Bleakley and Yamanishi previously [5].

8.10 Results

8.10.1 General Performance Comparisons with Clustered Boosting

In this study, the clustered boosting methods are compared to competing methods for genome-wide protein-chemical interaction prediction. These methods are evaluated using a random class balanced sample of 3200 interactions from the combined gold standard protein-chemical interaction data set.

Clustered boosting is first compared to AdaBoost, SVM, and LocalModels, with each meta classifier using SVM as well. For accuracy, precision, and F-measure, clustered boosting performs better than all other methods with significance at the 5% level according to 2-way ANOVA. Clustered boosting is also significantly better in terms of recall than AdaBoost and SVM, while local models perform significantly better. There is always a tradeoff between precision and recall, and it is evident that the local models approach trades an increase in total positive samples found for a decrease in the true positive rate.

Clustered boosting is also compared to competing methods using AdaBoost as the base classifier. In these experiments, many of the results are similar to those obtained using SVM as the base classifier, with the exception that the local models approach does not appear to make the same tradeoff between precision and accuracy, and instead produces a more balanced classification profile. Clustered boosting is significantly better than all other methods in all measured statistics, with the exception of recall compared to AdaBoost. In this case clustered boosting is better, but not significantly so.

Table 8.3. Comparison of clustered boosting to competing methods for genome-wide protein chemical interaction prediction, with SVM as a base model.

	Methods			
Statistic	ClustBoost	AdaBoost	SVM	LocalModel
Accuracy	**0.75**	0.73 ●	0.73 ●	0.65 ●
F-measure	**0.76**	0.74 ●	0.74 ●	0.71 ●
Precision	**0.74**	0.73 ●	0.73 ●	0.61 ●
Recall	0.78	0.75 ●	0.76 ●	**0.86** ○

○, ● statistically significant improvement or degradation

Table 8.4. Comparison of clustered boosting to competing methods for genome-wide protein chemical interaction prediction, with naïve bayes as a base model.

	Methods			
Measure	ClustBoost	AdaBoost	NaïveBayes	LocalModel
Accuracy	**0.71**	0.69 ●	0.66 ●	0.64 ●
F-measure	**0.72**	0.71 ●	0.66 ●	0.63 ●
Precision	**0.71**	0.69 ●	0.67 ●	0.65 ●
Recall	**0.73**	0.72	0.65 ●	0.62 ●

○, ● statistically significant improvement or degradation

8.10.2 Performance on Unbalanced Data

In this study, clustered boosting is compared to competing methods as the ratio of positive to negative samples is decreased. Naïve bayes was used as a base learner in these studies because of the faster computation time. Additionally, these results were generated using a slightly smaller random selection of 2000 samples from the complete set of protein-chemical interactions.

The results for accuracy show that clustered boosting continues to perform significantly better than competing methods as the percentage of positive samples included decreases. The results for F-measure show somewhat more variation. The clustered boosting method consistently outperforms each of the competing methods, although the performance increase is not always significant. Clustered boosting is significantly better than local models in all cases, and it is significantly better than both AdaBoost and naïve bayes in 3 of the 5 data sets.

Table 8.5. F-measure comparison of clustered boosting to competing methods as the percentage of positive samples is decreased, with naïve bayes as the base model.

% Positives	ClustBoost	AdaBoost	NaïveBayes	LocalModel
50%	**0.71**	0.70 ●	0.69 ●	0.66 ●
40%	**0.63**	0.62	0.62	0.58 ●
30%	**0.56**	0.55	0.55 ●	0.48 ●
20%	**0.46**	0.41 ●	0.44 ●	0.38 ●
10%	**0.32**	0.22 ●	0.30	0.24 ●

○, ● statistically significant improvement or degradation

Table 8.6. Accuracy comparison of clustered boosting to competing methods as the percentage of positive samples is decreased, with naïve bayes as the base model.

% Positives	ClustBoost	AdaBoost	NaïveBayes	LocalModel
50%	**70.76**	67.98 ●	66.56 ●	64.78 ●
40%	**69.82**	68.03 ●	65.08 ●	64.14 ●
30%	**72.38**	69.40 ●	65.54 ●	65.20 ●
20%	**76.41**	72.56 ●	65.59 ●	67.94 ●
10%	**86.13**	82.52 ●	75.35 ●	76.94 ●

○, ● statistically significant improvement or degradation

8.10.3 Parameter Evaluation for Clustered Boosting

The performance of clustered boosting is dependent on several parameters chosen during model construction. In order to analyze the response of clustered boosting to changes in these parameters, we have performed several studies measuring predictive performance as each parameter is indepedently varied. The number of clusters as well as the number of iterations are both important parameters in clustered boosting. The number of clusters determines the total number of models used to cover the sample interaction space, and by extension, determines the average number of samples used to train each model. The number of iterations determines how much opportunity each of the cluster models has to move toward the nearest group of well-classified samples. If the number of iterations is too low, the randomly selected clusters will not have the chance to move to optimal locations. If the number of iterations is too large, cluster models may stop moving and the contribution to the final model will be overwhelmed by only the models generated from the most recent clusters.

8.10.3.1 Response to Change in Number of Iterations

These results were obtained using a half-size data set of 1600 samples. The number of iterations for clustered boosting was varied between 5 and 100, and in each experiment the number of clusteres was determined using 10-fold cross-validation on the training data. The results detailing the response of clustered boosting to the number of iterations chosen shows a minimal amount of variation. The various statistics do not fluctuate much over the parameter range, and do not vary significantly. One trend, however, is that the performance at the lowest number of iterations is generally worse.

Table 8.7. Comparison of clustered boosting models as the number of iterations is changed, with the number of clusters determined by 10-fold cross-validation on the training data and using naïve bayes as the base model.

	Number of Iterations					
Statistic	5	10	15	25	50	100
Accuracy	0.68	0.69	0.69	0.69	0.69	0.69
F-measure	0.68	0.69	0.69	0.69	0.70	0.70
Precision	0.68	0.69	0.70	0.69	0.69	0.69
Recall	0.69	0.69	0.70	0.69	0.70	0.70
Train Time (s)	298.99	521.09	734.35	1298.95	2803.06	3479.53

8.10.3.2 Response to Change in Number of Clusters

As in the previous results, these results were obtained using a half size data set of 1600 samples. The number of clusters for clustered boosting was varied between 5 and 100 as well, and similarly the number of iterations was chosen using 10-fold cross-validation on the training data for each experiment. The results of these experiments are more clear than those obtained when varying the number of iterations. The performance according to each of the statistics gradually increases as the number of clusters increases. The training time results reveal, however, that the gradual performance boost using more clusters comes at a cost of increased computation time. In general, increasing the number of clusters should reduce the amount of data each model must learn, however as cluster centroids are adjusted the number of data points assigned to each cluster may become imbalanced. In the case of large numbers of clusters there is additional overhead as well.

8.10.4 Comparison to Similarity Boosting

For this experiment the results were obtained using a small, balanced sample of 100 interactions. Naïve bayes was used as a base model. The results com-

Table 8.8. Comparison of clustered boosting models as the number of clusters is changed, with the number of iterations determined by 10-fold cross-validation on the training data and using naïve bayes as the base model.

	Number of Clusters					
Statistic	5	10	15	25	50	100
Accuracy	0.66	0.66	0.68	0.68	0.71	0.71
F-measure	0.67	0.67	0.68	0.69	0.71	0.71
Precision	0.65	0.66	0.68	0.69	0.72	0.73
Recall	0.69	0.68	0.69	0.69	0.71	0.69
Train Time (s)	704.71	714.18	849.15	831.50	1453.18	1798.46

paring clustered boosting and similarity boosting show that performance is not significantly different, although clustered boosting performs slightly better. The training time results indicate a large difference however. It is clear even for small data sets that clustered boosting is faster by at least an order of magnitude.

Table 8.9. Comparison of SimBoost and clustered boosting with naïve bayes as base model for genome-wide protein-chemical interaction prediction with on a small, balanced selection of samples.

Dataset	ClustBoost	SimBoost
Accuracy	59.00	58.60
Train Time (s)	0.25	3.57

○, ● statistically significant improvement or degradation

Chapter 9

Discussion and Conclusions

Ensemble learning uses a series of models to make predictions about predictions about samples and combines those predictions in a meaningful way. It is hopeless to use multiple copies of the exact same models and therefore specialization of each model is required. A central issue in ensemble learning then is exactly how this specialization is done. In classic boosting the specialization is done according to sample misclassification. This approach is practical for many types of data, however in the case of protein-chemical interaction prediction it is limited by the fact that misclassified interaction samples may not share similarities suitable for prediction and generalization using a single model. For this reason it is more useful to specialize base models by sample similarity and spatial distribution. The similarity and clustered boosting methods described here are two attempts at addressing the limitation of traditional boosting for genome-wide interaction prediction.

SimBoost is a method for genome-wide protein-chemical interaction prediction that utilizes protein and chemical similarity to reduce the model's sensitivity to mislabeled training samples. This method has been evaluated in a number of protein-chemical interaction experiments and compared to competing methods.

The results gathered show that in the protein-chemical interaction cases evaluated, the SimBoost approach to interaction prediction provides a degree of tolerance to noise in the interaction labels that is not found in local model approaches. Overall, the SimBoost method shows competitive performance on a range of data sets. It's performance on the gold-standard data sets indicate that it is also tolerant to noise in the data class labels, a situation common in real world data sets.

Clustered boosting is an ensemble learning method for protein-chemical interaction prediction that explicitly specializes base models to disjoint regions of the interaction space. When applied to genome-wide prediction studies, clustered boosting consistently outperforms competing methods. Clustered boosting has been evaluated on balanced and unbalanced data selected from established protein-chemical interactions.

Clustered boosting approaches genome-wide interaction prediction from the same point of view as similarity boosting: sample similarity and nearness can be used to specialized models and improve generalization ability. The difference between the two approaches is that similarity boosting uses global sample reweighting to implicitly specialize base models according to the similarity. Clustered boosting, on the other hand, explicitly partitions the sample space and iteratively adjusts the models in each partition toward groups of well classified samples.

References

[1] D. K. Agrafiotis, W. Cedeno, and V. S. Lobanov. On the use of neural network ensembles in QSAR and QSPR. *Journal of chemical information and computer sciences*, 42:903–911, 2002.

[2] R. Ahuja, T. Magnanti, and J. Orlin. Network flows. *SIAM Review*, 37 No.1, 1995.

[3] A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl 1):i38–i46, 2005.

[4] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–42, 2000.

[5] K. Bleakley and Y. Yamanishi. Supervised prediction of drug target interactions using bipartite local models. *Bioinformatics*, 25:2397–2403, 2009.

[6] K. Borgwardt and H. Kriegel. Shortest-path kernels on graphs. In *Proceedings of the International Conference on Data Mining (ICDM)*, 2005.

[7] A. Chatr-aryamontri, A. Ceol, L. M. Palazzi, G. Nardelli, M. V. Schneider, L. Castagnoli, and G. Cesareni. Mint: the molecular interaction database. *Nucleic Acids Res*, 2007.

[8] J. Chen, S. J. Swamidass, Y. Dou, and P. Baldi. Chemdb: a public database of small molecules and related chemoinformatics resources. *Bioinformatics*, 21:4133–4139, 2005.

[9] K. K. Chohan, S. W. Paine, and N. J. Waters. Quantitative structure activity relationships in drug metabolism. *Current Topics in Medicinal Chemistry*, 6:1569–1578, 2006.

[10] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, and D. L. e. a. Grier. Description of several chemical structure file formats used by computer programs developed at molecular design limited. *Journal of Chemical Information and Computer Sciences*, 32:244–255, 1992.

[11] Daylight theory manual, 2011.

[12] J. A. de Sousa and J. Gasteiger. Prediction of enantiomeric excess in a combinatorial library of catalytic enantioselective reactions. *Journal of Combinatorial Chemistry*, 7:298–301, 2005.

[13] A. Deligiannakis and N. Roussopoulos. Extended wavelets for multiple measures. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.

[14] M. Deshpande, M. Kuramochi, and G. Karypis. Frequent sub-structure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[15] DRAGON. http://www.talete.mi.it.

[16] H. Fröhlich, J. Wegner, F. Sieker, and A. Zell. Kernel functions for attriubted molecular graphs - a new similarity-based approach to ADME prediction in classification. *QSAR & Combinatorial Science*, 25(4):317–326, 2006.

[17] Fröohlich, J. Wegner, F. Sieker, and A. Zell. Kernel functions for attributed molecular graphs - a new similarity-based approach to ADME prediction in classification. *QSAR & Combinatorial Science*, 2006.

[18] Z. Gao, H. Li, H. Zhang, X. Liu, L. Kang, X. Luo, W. Zhu, K. Chen, X. Wang, and H. Jiang. Pdtd: a web-accessible protein database for drug target identification. *BMC Bioinformatics*, 9(1):104, 2008.

[19] M. Garofalakis and A. Kumar. Wavelet synopses for general error metrics. *ACM Transactions on Database Systems (TODS)*, 30(4):888–928, 2005.

[20] T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Sixteenth Annual Conference on Computational Learning Theory and Seventh Kernel Workshop*, 2003.

[21] B. GD, D. I, W. C, O. BF, P. T, and H. CW. Bind–the biomolecular interaction network database. *Nucleic Acids Res.*, 29(1):242–5, 2001.

[22] S. Gomez, W. Noble, and A. Rzhetsky. Learning to predict protein-protein interactions from protein sequences. *Bioinformatics*, 19(15):1875–1881, 2003.

[23] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002 January.

[24] C. Hansch, A. Kurup, R. Garg, and H. Gao. Chem-bioinformatics and QSAR: A review of QSAR lacking positive hydrophobic terms. *Chemical Reviews*, 101:619–672, 2001.

[25] C. Hansch, A. Leo, S. B. Mekapati, and A. Kurup. QSAR and ADME. *Bioorganic & Medicinal Chemistry*, 12:3391–3400, 2004.

[26] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer-Verlag, 2001.

[27] T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *Proc. of the International Conference on Machine Learning*, 2004.

[28] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, pages 549–552, 2003.

[29] L. Jacob, B. Hoffmann, V. Stoven, and J.-P. Vert. Virtual screening of gpcrs: an in silico chemogenomics approach. Technical Report HAL-00220396, French Center for Computational Biology, 2008.

[30] L. Jacob and J.-P. Vert. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, 24(19), 2008.

[31] F. JL, M. M, M. S, S. K, and S. R. Genome scale enzyme-metabolite and drug-target interaction predictions using the signature molecular descriptor. *Bioinformatics*, 24(2):225–33, 2007.

[32] D. K, de Matos P, E. M, H. J, Z. M, M. A, A. R, D. M, G. M, and A. M. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic Acids Res*, 36:344–50, 2008.

[33] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, , and M. Hirakawa. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Research*, 34:D354–357, 2006.

[34] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

[35] A. Kovatcheva, A. Golbraikh, S. Oloff, Y. D. Xiao, W. Zheng, P. Wolschann, G. Buchbauer, and A. Tropsha. Combinatorial QSAR of ambergris fragrance compounds. *J. Chem. Inf. Comput. Sci.*, 44:582–595, 2004.

[36] S. KP, G. GA, H. MP, B. NE, C. HA, N. S, B. S, S. JP, M. J, S. M, F. P, T. NJ, S. SL, and C. PA. Chembank: a small-molecule screening and cheminformatics resource database. *Nucleic Acids Res*, 36:351–9, 2008.

[37] R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Journal of Bioinformatics and Computational Biology*, 3 (3):527–550, 2005.

[38] H. Kubinyi. Quantitative relationships between chemical-structure and biological-activity. *Drug Discovery Today*, 2:457–467, 1997.

[39] T. Kudo, E. Maeda, and Y. Matsumoto. An application of boosting to graph classification. In *NIPS*, 2004.

[40] L. L, B.-E. K, B. PH, R. JJ, H. JR, N. JA, P. ME, and M. SO. Biodrugscreen: a computational drug design resource for ranking molecules docked to the human proteome. *Nucleic Acids Res*, 38:765–73, 2010.

[41] R. L, S. SJ, S. H, and B. P. Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110, 2005.

[42] A. R. Leach, B. K. Shoichet, and C. E. Peishoff. Prediction of protein-ligand interactions. docking and scoring: Successes and gaps. *Journal of medicinal chemistry*, 49(20):5851–55, 2006.

[43] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: a string kernel for svm protein classification. In *Proceedings of the Pacific Symposium on Biocomputing 2002, World Scientific, Singapore*, 2002.

[44] Z. Li, H. Lin, L. Han, L. Jiang, X. Chen, , and Y. Chen. Profeat: A web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. *Nucleic Acids Res*, 34:32–7, 2006.

[45] S. Lyu. Mercer kernels for object recognition with local features. In *IEEE Computer Vision and Pattern Recognition*, pages 223–229, 2005.

[46] P. Mahe and J. Vert. Graph kernels based on tree patterns for molecules. Technical Report HAL:ccsd-00095488, Ecoles des Mines de Paris, September 2006.

[47] S. Martin, D. Roe, and J.-L. Faulon. Predicting protein-protein interactions using signature products. *Bioinformatics*, 21(2):218–226, 2005.

[48] A. McNaught. The iupac international chemical identifier:inchi. *Chemistry International (IUPAC)*, 28(6), 2006.

[49] P. Murray-Rust, H. S. Rzepa, and M. Wright. Development of chemical markup language (cml) as a system for handling complex chemical content. *New J. Chem.*, pages 618–634, 2001.

[50] N. Nagamine and Y. Sakakibara. Statistical prediction of protein chemical interactions based on chemical structure and mass spectrometry data. *Bioinformatics*, 23:2004–2012, 2007.

[51] Y. Okuno, J. Yang, K. Taneishi, H. Yabuuchi, and G. Tsujimoto. GLIDA: GPCR-ligand database for chemical genomic drug discovery. *Nucleic Acids Res.*, 2006(9).

[52] Oyama, Satoshi, Manning, and C. D. Using feature conjunctions across examples for learning pairwise classifiers. In *15th European Conference on Machine Learning (ECML2004)*, 2004.

[53] P. P, K. S, O. M, B. B, D.-K. I, F. G, M. C, M. P, S. V, M. HW, R. A, and F. D. The mips mammalian protein-protein interaction database. *Bioinformatics*, 21(6):832–834, 2005.

[54] G. V. Paolini, R. H. B. Shapland, W. P. van Hoorn, J. S. Mason, and A. L. Hopkins. Global mapping of pharmacological space. *Nature Biotechnology*, 24:805–815, 2006.

[55] Pubchem. http://pubchem.ncbi.nlm.nih.gov.

[56] R. Put, Q. S. Xu, D. L. Massart, and Y. V. Heyden. Multivariate adaptive regression splines (MARS) in chromatographic quantitative structure-retention relationship studies. *Journal of Chromatography*, 1055(A):11–19, 2004.

[57] M. Shen, C. Beguin, A. Golbraikh, J. P. Stables, H. Kohn, and A. Tropsha. Application of predictive QSAR models to database mining: identification and experimental validation of novel anticonvulsant compounds. *J. Med. Chem.*, 47:2356–2364, 2004.

[58] A. Smalter, J. Huan, and G. Lushington. Structure-based pattern mining for chemical compound classification. In *Proceedings of the 6th Asia Pacific Bioinformatics Conference*, 2008.

[59] H. Strombergsson and G. Kleywegt. A chemogenomics view on protein-ligand spaces. *BMC Bioinformatics*, 10(Suppl 6):S13, 2009.

[60] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston. A classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences*, 43:1947–1958, 2003.

[61] S. W. Tamas Horvath, Thomas Gartner. Cyclic pattern kernels for predictive graph mining. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[62] D. Tao, X. Li, X. Wu, W. Hu, and S. J. Maybank. Supervised tensor learning. *Journal of Knowledge and Information Systems*, 13, 2007.

[63] I. V. Tetko. Neural network studies. 4. introduction to associative neural networks. *Journal of chemical information and computer sciences*, 42:717–728, 2002.

[64] S. V. N. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In *In Advances in Neural Information Processing Systems*, 2006.

[65] D. Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, 28(31), 1988.

[66] D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, Z. Chang, and J. Woolsey. Drugbank: a comprehensive resource for in silico drug discovery and exploratin. *Nucleic Acids Res.*, 2006(1).

[67] C. X, J. ZL, and C. YZ. Ttd: Therapeutic target database. *Nucleic Acids Res*, 20(1):412–5, 2002.

[68] I. Xenarios, E. Fernandez, L. Salwinski, X. J. Duan, M. J. Thompson, E. M. Marcotte, and D. Eisenberg. Dip: The database of interacting proteins: 2001 update. *Nucleic Acids Res*, 29:239–241, 2001.

[69] Y. Y. Xiao and M. R. Segal. Prediction of genomewide conserved epitope profiles of hiv-1: Classifier choice and peptide representation. *Statistical Applications in Genetics and Molecular Biology*, 4, 2005.

[70] Y. Xue, H. Li, C. Y. Ung, C. W. Yap, and Y. Z. Chen. Classification of a diverse set of tetrahymena pyriformis toxicity chemical compounds from molecular descriptors by statistical learning methods. *Chem. Res. Toxicol.*, 19 (8), 2006.

[71] Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kanehisa. Prediction of drug target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13):i232–i240, 2008.

Appendices

Table 9.1. Appendix: Topological descriptiors generated by PRO-FEAT

Name
Number of Atoms
Number of Heavy atoms
Number of H atoms
Number of B atoms
Number of C atoms
Number of N atoms
Number of O atoms
Number of F atoms
Number of P atoms
Number of S atoms
Number of Cl atoms
Number of Br atoms
Number of I atoms
Number of Bonds
Number of non-H Bonds
Number of rings
Molecular weight
Average molecular weight
Number of H-bond donnor
Number of H-bond acceptor
Number of 3-member rings
Number of 4-member ings
Number of 7-member rings
Number of 5-member non-aromatic rings
Number of 6-member non-aromatic rings
Number of 5-member aromatic rings
Number of 6-member aromatic rings
Number of heterocyclic rings
Number of N heterocyclic rings
Number of O heterocyclic rings
Number of S heterocyclic rings

Table 9.2. Appendix: Topological descriptiors generated by PRO-FEAT (cont'd)

Name
Fingerprint for primary carbocation
Fingerprint for secondary carbocation
Fingerprint for tertiary carbocation
Fingerprint for organohalide
Fingerprint for amonium ion
fingerprint for primary amonium
fingerprint for secondary amonium
fingerprint for tertiary amonium
fingerprint for nitro
fingerprint for nitrile
fingerprint for diazo
fingerprint for phenol(Ph-OH)
fingerprint for primary alcohol
fingerprint for second alcohol
fingerprint for tertiary alcohol
fingerprint for Ph-O-Ph
fingerprint for ether(R-O-R)
fingerprint for aldehyde(R-CHO)
fingerprint for ketone(R-CO-R)
fingerprint for carboxylic acid(R-COOH)
fingerprint for carboxylate ion(R-COO(-))
fingerprint for acyl cation(R-CO(+))
fingerprint for ester(R-COOR)
fingerprint for Acid anhydride(R-CO-O-COR)
fingerprint for Alkoxide ion(R-O(-))
fingerprint for peroxide(R-O-O-R)
Fingerprint for epoxide(c-O-c ring)
Fingerprint for diol(C(OH)-C(OH)-)
Fingerprint for organosilicon
Fingerprint for organoarsenical
Fingerprint for thiol(R-SH)
Fingerprint for thiophenol(Ph-SH)
Fingerprint t for R-S-R
Fingerprint for Ph-S-Ph
Fingerprint for sulfonic acid
Fingerprint for thioketone(R-C=S)
Fingerprint t for phosphonic acid
Fingerprint for phosphinic acid
Fingerprint for organophophosphate ester

Table 9.3. Appendix: Topological descriptiors generated by PROFEAT (cont'd)

Name
Fingerprint for carboxylic thioester
Fingerprint for sulfate ester
Fingerprint for thiophosphate ester
Fingerprint for amide
Fingerprint for alpha-amino acid
Fingerprint for hydroxynitrile
Fingerprint for oxime
Fingerprint for nitrate ester
Fingerprint for acid halide(RCOX)
Number of rotable bonds
Schultz molecular topological index
Gutman molecular topological index
Topological charge index G1
Topological charge index G2
Topological charge index G3
Topological charge index G4
Topological charge index G5
Mean topological charge index J1
Mean topological charge index J2
Mean topological charge index J3
Mean topological charge index J4
Mean topological charge index J5
Global topological charge index J
Wiener index
Mean Wiener index
Harary index
Gravitational topological index
Molecular path count of length 1
Molecular path count of length 2
Molecular path count of length 3
Molecular path count of length 4
Molecular path count of length 5
Molecular path count of length 6

Table 9.4. Appendix: Topological descriptiors generated by PRO-FEAT (cont'd)

Name
Total path count
Xu index
Modified Xu Index
Balaban Index J
Platt Number
First Zagreb Index(M1)
Second Zagreb Index(M2)
First Modified Zagreb Index
Second Modified Zagreb Index
Quadratic index(Q)
0th edge connectivity index
Edge connectivity index
Extened edge connectivity inndex
0th Kier-Hall connectivity index
1th Kier-Hall connectivity index
Mean Randic Connectivity index
2th Kier-Hall connectivity index
Simple topological index by Narumi
Harmonic topological index by Narumi
Geometric topological index by Narumi
Arithmetic topological index by Narumi
0th valence connectivity index
1th valence connectivity index
2th valence connectivity index
0th order delta chi index
1th order delta chi index
2th order delta chi index
Pogliani index

Table 9.5. Appendix: Topological descriptiors generated by PROFEAT (cont'd)

0th Solvation connectivity index
1th Solvation connectivity index
2th Solvation connectivity index
1th order Kier shape index
2th order Kier shape index
3th order Kier shape index
1th order Kappa alpha shape index
2th order Kappa alpha shape index
3th order Kappa alpha shape index
Molecular Flexibility Index
Topological radius
Topological diameter
Graph-theoretical shape coefficient
Eccentricity
Average atom eccentricity
Mean eccentricity deviation
Average distance degree
Mean distance degree deviation
Unipolarity
Rouvary index
Centralization
Variation
Dispersion
Log of PRS INDEX
RDSQ ondex
RDCHI index
Optimized 1th connectivity index
Logp from connectivity
BCUT descriptors
Moreau-Broto Autocorrelation descriptors
Moran Autocorrelation descriptors
Geary Autocorrelation descriptors