Development and evaluation of machine learning algorithms for biomedical applications

Turki Talal Turki
New Jersey Institute of Technology

ABSTRACT

DEVELOPMENT AND EVALUATION OF MACHINE LEARNING AND DATA MINING ALGORITHMS FOR BIOMEDICAL APPLICATIONS

by
Turki Talal Turki

Gene network inference and drug response prediction are two important problems in computational biomedicine. The former helps scientists better understand the functional elements and regulatory circuits of cells. The latter helps a physician gain full understanding of the effective treatment on patients. Both problems have been widely studied, though current solutions are far from perfect. More research is needed to improve the accuracy of existing approaches.

This dissertation develops machine learning and data mining algorithms, and applies these algorithms to solve the two important biomedical problems. Specifically, to tackle the gene network inference problem, the dissertation proposes (i) new techniques for selecting topological features suitable for link prediction in gene networks; (ii) a graph sparsification method for network sampling; (iii) combined supervised and unsupervised methods to infer gene networks; and (iv) sampling and boosting techniques for reverse engineering gene networks. For drug sensitivity prediction problem, the dissertation presents (i) an instance selection technique and hybrid method for drug sensitivity prediction; (ii) a link prediction approach to drug sensitivity prediction; (iii) a noise-filtering method for drug sensitivity prediction; and (iv) transfer learning approaches for enhancing the performance of drug sensitivity prediction. Substantial experiments are conducted to evaluate the effectiveness and efficiency of the proposed algorithms. Experimental results demonstrate the feasibility of the algorithms and their superiority over the existing approaches.

DEVELOPMENT AND EVALUATION OF MACHINE LEARNING ALGORITHMS FOR BIOMEDICAL APPLICATIONS

by
Turki Talal Turki

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

May 2017

APPROVAL PAGE

DEVELOPMENT AND EVALUATION OF MACHINE LEARNING AND DATA MINING ALGORITHMS FOR BIOMEDICAL APPLICATIONS

Turki Talal Turki

Dr. Jason T. L. Wang, Dissertation Co-Advisor Professor of Computer Science, NJIT	Date
Dr. Zhi Wei, Dissertation Co-Advisor Associate Professor of Computer Science, NJIT	Date
Dr. Ali Mili, Committee Member Professor of Computer Science, NJIT	Date
Dr. David Nassimi, Committee Member Associate Professor of Computer Science, NJIT	Date
Dr. Mengchu Zhou, Committee Member Distinguished Professor of Electrical and Computer Engineering, NJIT	Date

BIOGRAPHICAL SKETCH

Author:	Turki Talal Turki
Degree:	Doctor of Philosophy
Date:	May 2017

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2017
- Master of Science in Computer Science,
  Polytechnic Institute of New York University, Brooklyn, NY, 2012
- Bachelor of Science in Computer Science,
  King Abdulaziz University, Jeddah, Saudi Arabia, 2008

Major: Computer Science

Presentations and Publications:

T. Turki, Z. Wei, and J. T. L. Wang, "Transfer Learning Approaches to Improve Drug Sensitivity Prediction in Multiple Myeloma Patients," Accepted at IEEE Access.

T. Turki and J. T. L. Wang, "Reverse Engineering Gene Regulatory Networks Using Sampling and Boosting Techniques," Accepted at the 13th International Conference on Machine Learning and Data Mining, New York, NY, 2017.

Y. Abduallah, T. Turki, K. Byron, Z. Du, M. Cervantes-Cervantes, and J. T. L. Wang, "MapReduce Algorithms for Inferring Gene Regulatory Networks from Time-Series Microarray Data Using an Information-Theoretic Approach," BioMed Research International, vol. 2017, Article ID 6261802, 8 pages, 2017.

T. Turki and Z. Wei, "A Noise-Filtering Approach for Cancer Drug Sensitivity Prediction," in the Proceedings of the Neural Information Processing Systems Workshop on Machine Learning for Health (NIPS ML4HC), Barcelona, Spain, 2016.

T. Turki and Z. Wei, “A Link Prediction Approach to Cancer Drug Sensitivity Prediction,” International Conference on Intelligent Biology and Medicine (ICIBM), Houston, Texas, USA, 2016. Accepted for inclusion as a special issue in BioMed Central (BMC) Systems Biology.

T. Turki, J. T. L. Wang, and I. Rajikhan, “Inferring Gene Regulatory Networks by Combining Supervised and Unsupervised Methods,” in the Proceedings of the 15th International Conference on Machine Learning and Applications (ICMLA), Anaheim, California, 2016.

T. Turki and Z. Wei, “Learning Approaches to Improve Prediction of Drug Sensitivity in Breast Cancer Patients,” in the Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Orlando, FL, 2016.

T. Turki and J. T. L. Wang, “A Learning Framework to Improve Unsupervised Gene Network Inference,” in the Proceedings of the 12th International Conference on Machine Learning and Data Mining, New York, NY, pp. 28-42, 2016.

T. Turki and Z. Wei, “A Greedy-Based Oversampling Approach to Improve the Prediction of Mortality in MERS Patients,” in the Proceedings of the 10th Annual IEEE International Systems Conference, Orlando, FL, 2016.

T. Turki and J. T. L. Wang, "A New Approach to Link Prediction in Gene Regulatory Networks," in the Proceedings of the 16th International Conference on Intelligent Data Engineering and Automated Learning, Wroclaw, Poland, 2015.

T. Turki and Z. Wei, “IPRed: Instance Reduction Algorithm Based on the Percentile of the Partitions,” in the Proceedings of the 26th Modern AI and Cognitive Science Conference, Greensboro, NC, 2015.

T. Turki and U. Roshan, "MaxSSmap: A GPU program for divergent short read mapping to genomes with the maximum scoring subsequence," BioMed Central Genomics (BMC Genomics), 2014.

T. Turki, M. Amimul Ihsan, N. Turki, J. Zhang, U. Roshan, and Z. Wei, “Top-k Parametrized Boost,” in the Proceedings of the Second International Conference on Mining Intelligence and Knowledge Exploration, Cork, Ireland, 2014.

T. Turki and U. Roshan, “Weighted Maximum Variance Dimensionality Reduction,” in the Proceedings of the 6th Mexican Conference on Pattern Recognition, Cancun, Mexico, 2014.

To my parents, my wife, and my family.

ACKNOWLEDGMENT

My sincere thanks to my co-advisors Professors Jason T. L. Wang and Zhi Wei for their support and continual guidance. Without their guidance and knowledge, the task of completing my PhD studies would have been an infinitely harder task. Their insights into solving real biological problems and building models that are useful for biologists inspired my work on adapting machine learning algorithms to solve real biological problems. I am fortunate to have worked with them. They have provided me with excellent training.

Special thanks are expressed to Professors Ali Mili, David Nassimi, and Mengchu Zhou for participating in my defense committee; their time and effort are appreciated.

The first courses I took when I joined the PhD program were Discrete Mathematics and Algorithms with Dr. David Nassimi. His teaching style and explanations greatly sharpened my thinking.

I have also greatly benefited from Dr. Wang's classes on data mining and bioinformatics, which introduced me to link prediction and network inference.

I thank King Abdulaziz University and Saudi Arabian Cultural Mission for their financial and academic support.

I am also thankful to my wife for her patience and support throughout my PhD studies. Our lives have been enhanced with the birth of our son Talal, who makes us stronger.

Last and most important, nobody has had as much impact on my life as my wonderful parents, who have always encouraged and supported me throughout my entire life. I have been lucky and blessed in my life.

TABLE OF CONTENTS

**TABLE OF CONTENTS
(Continued)**

Chapter **Page**

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

Table **Page**

LIST OF TABLES
(Continued)

LIST OF TABLES
(Continued)

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

CHAPTER 1

BACKGROUND

1.1 Gene Network Inference

Inference of gene regulatory networks (GRNs) from steady-state and time-series gene expression data is an important biological problem that has received increasing attention for advancing genomics research. Many computational approaches to solving this problem have been developed to correctly infer gene regulatory networks from gene expression data. However, existing approaches are not sufficiently accurate and robust to convert gene expression data to regulatory relationships between transcription factors and target genes.

The first part of this dissertation proposes several computational approaches for making accurate and robust network inference. Chapter 2 develops a supervised link prediction approach that combines gene expression data with topological features extracted from a partially known gene regulatory network, and uses machine learning algorithms for training and predicting unseen links in the gene regulatory network. Chapter 3 introduces a learning framework that receives as input a network constructed by an unsupervised method and employs a graph sparsification technique for network sampling and principal component analysis for feature selection to obtain better quality training data for regulatory link prediction. Chapter 4 develops a data cleaning algorithm that takes as input a network constructed by an unsupervised method, and produces a better quality training set by incorporating techniques from linear algebra and machine learning. Then, the training set is passed to machine learning and deep learning algorithms for training and performing predictions on a test set. Chapter 5 proposes several methods for regulatory link prediction using sampling and boosting techniques. These methods include (i) an upward extension of AdaBoost, called Boost I, that takes as input a training set corresponding to a partially observed gene regulatory network and iteratively boosts the performance of a weighted decision tree during training and classifying remaining unobserved links; (ii) Boost I+U which is the same as Boost I except that Boost I+U adopts an additional undersampling technique for improving prediction performance; (iii) Boost I+O which is the same as Boost I+U except that Boost I+O uses an oversampling technique instead of an undersampling technique; (iv) a different extension of AdaBoost, called Boost II, that iteratively boosts the performance of a weighted decision tree for training and testing; (v) Boost II+U which is the same as Boost I+U except that Boost II+U uses a different boosting technique; (vi) Boost II+O which also works the same as Boost I+O except that Boost II+O uses a different boosting technique; (vii) Boost III which is a new boosting technique that is different from Boost I and Boost II; (viii) Boost III+U which is the same as Boost II+U except that Boost III is a different boosting technique; and (ix) Boost III+O which works the same as Boost II+O except that Boost III is different from Boost II.

To evaluate the performance of the proposed approaches and compare them against existing approaches, comprehensive experiments were conducted on many datasets. Experimental results showed that the proposed approaches are robust and outperform the existing approaches.

1.1.1 Link Prediction in Gene Regulatory Networks

Link prediction is an important data mining problem that finds many applications in social network analysis. One of the methods for solving the link prediction problem is to extract features from a given partially observed network and incorporate these features into a classifier. The links (i.e., edges) between entities (i.e., nodes or vertices) in the given partially observed network are labeled [1, 2]. One then uses the classifier built from the given partially observed network to predict the presence of links for unobserved pairs of entities in the network. Liben-Nowell and Kleinberg [3] showed that topological features can be used to increase the accuracy of link prediction in social network analysis.

Several authors have developed supervised methods for GRN inference [4-6]. For example, Gillani *et al*. [7] presented CompareSVM, which uses support vector machines (SVMs) to predict the regulatory relationship between a transcription factor (TF) and a target gene where the regulatory relationship is represented by a directed edge (link), and both the TF and target gene are nodes in a gene network. SIRENE [4, 8] is another supervised method, which splits the network inference problem into many binary classification problems using one SVM for each TF. The trained SVM classifiers are then used to predict which genes are regulated. The final step is to combine all SVM classifiers to produce a ranked list of TF-gene interactions in decreasing order, and to construct a network based on the ranked list. Cerulo *et al*. [5] developed a SVM-based method for GRN inference, which uses positive and unlabeled data for training the SVM classifier. Ernst *et al*. [9] developed a similar semi-supervised approach for GRN inference.

1.1.2 Network Inference Through Link Prediction

Network inference through link prediction is a major research topic in computational social science [1, 10, 11] and biomedicine [12-14]. For example, computational biologists develop different methods for reconstructing gene regulatory networks (GRNs) using high throughput genomics data. Maetschke *et al*. [15] categorized the existing GRN reconstruction algorithms into three groups: unsupervised, supervised and semi-supervised. While supervised algorithms are capable of achieving the highest accuracy among all the network inference methods, these algorithms require a large number of positive and negative training examples, which are difficult to obtain in many organisms [15-17]. Unsupervised algorithms infer networks based solely on gene expression profiles and do not need any training example; however, the accuracy of the unsupervised algorithms is low as they often create missing and spurious links [15].

GRNs inferred by unsupervised methods use time-series gene expression data. These methods include BANJO (Bayesian Network Inference with Java Objects) [18], TimeDelay-ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks) [19], tlCLR (Time-Lagged Context Likelihood of Relatedness) [20, 21], DFG (Dynamic Factor Graphs) [22], BPDS (Boolean Polynomial Dynamical Systems) [23], MIDER [24], Jump3 [25], ScanBMA [26], and Inferelator [27]. BANJO models networks as a first-order Markov process; it searches through all possible networks, seeking the network with the best score. TimeDelay-ARACNE infers networks from time-series data using mutual information from information theory.

The tlCLR method also uses mutual information and depends on ordinary differential equations to model time-series data. DFG models experimental noise as a fitted Gaussian and then infers networks based on an assumed underlying, idealized gene expression pattern. Jump3 uses a non-parametric procedure based on decision trees to reconstruct GRNs. ScanBMA is a Bayesian inference method that incorporates external information to improve the accuracy of GRN inference. Inferelator uses ordinary differential equations that learn a dynamical model for each gene using time-series data. Recent extensions of Inferelator incorporate prior knowledge into the tool, and are resilient to noisy inputs.

1.1.3 Network Inference via Supervised and Unsupervised Methods

Current biotechnology has allowed researchers in various fields to obtain immense amounts of experimental information, ranging from macromolecular sequences, gene expression data to proteomics and metabolomics. In addition to large-scale genomic information obtained through such methods as third generation DNA sequencing, newer technology, such as RNA-seq and ChIP-seq, has allowed researchers to fine tune the analysis of gene expression patterns [28]. More information on interactions between transcription factors and DNA, both qualitative and quantitative, is increasingly emerging from microarray data.

Although microarrays alone do not provide direct evidence of functional connections among genes, the attachment of transcription factors (TFs) and their binding sites (TFBSs), located at specific gene promoters, influences transcription and modulates RNA production from that particular gene, thus establishing a first level of functional interaction. Since the TFs are gene-encoded polypeptides and the target TFBSs belong to different genes, analyses of TFs-TFBSs interactions could uncover gene networks and may even contribute to elucidate unknown GRNs [29]. Besides contributing to infer and understand these interactions, determining GRNs also aims to provide explanatory models of such connections [30]. GRNs could be the basis to infer more complex networks, encompassing gene, protein, and metabolic spaces, as well as the entangled and often overlooked signaling pathways that interconnect them [14, 31, 32].

Maetschke *et al*. [33] categorized GRN inference methods into three groups: supervised, unsupervised and semi-supervised. While supervised algorithms are capable of achieving the highest accuracy among all the GRN inference methods, these algorithms require a large number of positive and negative training examples, which are difficult to obtain in many organisms [5, 34, 35]. Unsupervised algorithms infer GRNs based solely on gene expression profiles and do not need any training example; however, the accuracy of the unsupervised algorithms is low [33].

1.1.4 Networks Inference Using Sampling and Boosting Techniques

Gene regulation is a series of processes that control gene expression and its extent. The connections among genes and their regulatory molecules, usually transcription factors, and a descriptive model of such connections, are known as gene regulatory networks (GRNs). Elucidating GRNs is crucial to understand the inner workings of the cell and the interactions among genes. Furthermore, GRNs could be the basis to infer more complex networks, encompassing gene, protein, and metabolic spaces, as well as the entangled and often over-looked signaling pathways that interconnect them [36-40].

Existing GRN inference methods can be broadly categorized into two groups: unsupervised and supervised [41]. Unsupervised methods infer GRNs based solely on gene expression data. The accuracy of these methods is usually low. By contrast, supervised methods use machine learning algorithms and training data to achieve higher accuracy. These methods work as follows. We represent a GRN by a directed graph in which each node is a gene or transcription factor, and a directed link or edge from node A to node B indicates that gene A regulates the expression of gene B. The training data contains a partially known network with known present edges and absent edges between nodes. These known present edges are positive training examples, and the known absent edges are negative training examples. We train a machine learning algorithm using the training data and apply the trained model to predict the remaining unknown edges in the network. With the predicted present and absent edges, we are able to infer or construct a complete GRN.

GRNs are always sparse graphs. The ratio between the number of gene interactions (i.e., edges or links) and the number of genes (i.e., nodes) falls between 1.5

and 2.75 regardless of the differences in phylogeny, phenotypic complexity, life history, and the total number of genes in an organism [42]. Thus, all GRNs have relatively few present edges and a lot of absent edges. This means there are few positive examples and a lot of negative examples when modeling the GRN inference problem as the link prediction problem described above. This poses an imbalanced classification problem in which the positive class (i.e., the minority class) is much smaller than the negative class (i.e., the majority class). However, existing supervised GRN inference methods [8] [43] do not take into consideration the imbalanced datasets, and hence their performance is unsatisfactory.

1.2 Cancer Drug Sensitivity Prediction

The problem of cancer drug sensitivity prediction has attracted considerable recent attention from various domains such as computational biology, machine learning, and data mining. As a result, many computational approaches have been proposed to predict correctly the response of cancer to drugs using genomic information with the associated drug values, where both are modeled as a single data matrix. However, many of these approaches are not accurate and robust enough to connect genomic information to drug values and cancer response. This dissertation presents computational approaches to make accurate and robust cancer drug sensitivity predictions.

The second part of this dissertation proposes different computational approaches to improve the prediction performance of cancer drug sensitivity prediction. Specifically, Chapter 6 presents: (i) instance selection approach to select the most important samples (i.e., examples), which are then provided to the standard machine learning algorithm to train and perform prediction on new samples (i.e., test set); (ii) an oversampling approach to generate synthetic samples, which are provided as input along with the original samples to a machine learning algorithm for training and performing prediction on a testing set; (iii) a hybrid approach that performs a majority vote on predictions obtained on the test set based on models generated using machine learning algorithms trained on different selected samples and genes. Chapter 7 introduces: (i) a link prediction approach to cancer drug sensitivity prediction; (ii) an algorithm employing the link prediction approach and a modified version of Query by Committee to select samples and provide them as input to a machine learning algorithm for training and predicting a test set; (iii) an extended algorithm to the previous one that performs an additional step, gene selection using statistical leverage scores. Chapter 8 develops a noise filtering approach derived from numerical linear algebra and information retrieval techniques to select the most important samples and pass them to a standard machine learning algorithm for training and performing prediction on the test set. Chapter 9 proposes: (i) a transfer learning approach that changes the representation of auxiliary data from the related task to a new representation that is closer to the target training set and incorporating the data with changed representation and the target training set, where both are passed to a standard machine learning algorithm for training and performing prediction on testing; (ii) an extended transfer learning approach to the previous approach, which includes a modified version of Adaboost to boost the performance on the test set.

To compare the proposed prediction algorithms against existing approaches, experimental evaluations of all approaches were performed using the Area Under the ROC Curve (AUC) on test sets corresponding to real clinical trial datasets of cancer patients. Experimental results demonstrate the stability and superiority of the proposed approaches over the existing approaches in terms of statistical significance and accuracy.

1.2.1 Approaches to Cancer Drug Sensitivity Prediction

Cancer is a major public health problem in the world and the second leading cause of deaths in the Unites States of America [44]. Patients respond differently to cancer chemotherapy owing to genetic heterogeneity, tumor heterogeneity, and environmental factors, which make cancer drug discovery very difficult [45-48]. Cancer, because it tends to be a progressive threat, has attracted attention of researchers from various domains for identifying novel cancer genes and for cancer drug discovery [49-52]. Costello *et al.* [53] assessed 44 drug sensitivity prediction algorithms based on profiling datasets (i.e., genomic, proteomic, and epigenomic data) of patients in breast cancer cell lines. The training set consist of 35 cell lines (i.e., instances), where each cell line was associated with 28 drugs response (output) values. The test set consist of 18 cell lines (i.e., instances). The task of each algorithm was to predict the response (i.e., ranking 28 drugs from most sensitive to most resistant) for each cell line of the test set. The 44 algorithms were assigned to six categories: (i) kernel methods, (ii) nonlinear regression, (iii) sparse linear regression (SLR), (iv) partial least-squares or principal component regression, (v) ensemble/model selection and (vi) other (those methods not falling cleanly into the previous five categories). Georgii *et al*. [53], the top-performing team, presented Bayesian multitask multiple kernel learning (MKL) that integrates multiple profiling datasets and enhanced data representations into probabilistic nonlinear regression model to learn and predict drug sensitivity for all drugs simultaneously. Wan *et al*. [53], the second-best performing team, employed random forest regression trees to address drug sensitivity prediction task. Prediction algorithms were evaluated using weighted probabilistic *c*-index (wpc-index) [53]. The other teams were not statistically significantly different according to the performed analysis on the algorithms.

Geeleher *et al*. [54] proposed an approach to drug sensitivity prediction performing homogenization and filtering on input data (baseline gene expression levels with drug response values and in vivo tumor gene expression). A Learning algorithm is applied to the baseline gene expression levels in the cell lines with associated drug response values, to learn a model. The resulting learned model is then applied to the baseline tumor expression data from the clinical trial, to yield drug sensitivity predictions.

Several problems are associated with the previous supervised approaches: (i) The poor quality of cell lines, especially when cell lines are not screened against all the compounds [55]; (ii) The lack of sufficiently large and representative cancer cell lines, as they provide the basis to improve the accuracy of prediction algorithms. However, this requires larger infrastructures and associated with higher costs of screening size [56].

1.2.2 Cancer Drug Sensitivity Prediction Through Link Prediction

Cancer has a significant global impact on public health; it is the second leading cause of death in the United States of America [44]. Cancer patients respond differently to potential drugs (i.e., chemotherapy) due to environmental causes, tumor heterogeneity, and genetic factors, making cancer drug discovery difficult [45-48]. The increasing number of deaths associated with cancer has attracted the attention of researchers from numerous domains, such as computational biology, machine learning, and data mining [49-52]. Costello *et al.* [53] assessed the performance of 44 drug sensitivity prediction algorithms based on profiling datasets (i.e., genomic, proteomic, and epigenomic data) in breast cancer cell lines. The training set consists of 35 cell lines, in which each cell line is associated with 28 drug responses. The test set consists of 18 cell lines. The task of each prediction algorithm is to learn a model from the training cell lines and perform predictions on the test set. The predictions correspond to a ranking of the 28 drugs—from the most sensitive to the most resistant for each cell line on the test set. The top-performing approach [53] improved the performance by integrating several profiling datasets with improved representation with a probabilistic nonlinear regression model. The second-best performing approach employed random forest regression to make predictions on the test set. The prediction algorithms were evaluated using the weighted probabilistic *c*-index (wpc-index) and resampled Spearman correlations [53]. The remaining prediction algorithms were not statistically different.

Geeleher *et al*. [54] proposed the following approach to drug sensitivity in which the input data are baseline expressions with drug IC_{50} values in cell lines and in vivo tumor gene expressions. The raw microarray data for the cell lines and clinical trials are processed separately and then combined and homogenized. The homogenized expression data consist of cell line expression data (i.e., baseline gene expression levels in the cell lines) and clinical trial expression data (i.e., baseline tumor expression data from the clinical trial). A learning algorithm is applied to the cell line expression data with the associated drug IC_{50} values for cell lines to learn a model. The resulting model is applied to clinical trial expression data to yield drug sensitivity predictions.

Two problems associated with the previous drug sensitivity prediction algorithms contribute to the degradation of the performance: (1) the poor quality of cell lines, especially when cell lines are not screened against all compounds [55]; and (2) the failure to adopt a new feature representation, because new feature representations provide a basis for improving the performance of learning algorithms [57-59].

1.2.3 Filtering Noisy Cancer Cell Lines for Drug Sensitivity Prediction

Cancer has a significant impact on public health worldwide and is the second leading cause of death in the US [60]. In 2016, the American Cancer Society [61] predicted that 1,685,210 new cancer cases will be diagnosed, resulting in 595,690 deaths attributable to cancer in the US. Many of these cancer patients respond differently to the same cancer drug (i.e., during chemotherapy). These response differences are attributable to environmental (i.e., external) factors such as tobacco, infectious organisms, and an unhealthy diet, as well as to genetic (i.e., internal) factors such as inherited genetic mutations, hormones, or immune conditions, and cancer cell heterogeneity, all of which make cancer drug discovery very difficult [45, 46, 48, 62]. Because of the significant numbers of deaths associated with cancer, its study has attracted the attention of researchers from numerous domains including computational biology, machine learning, and data mining [49, 52, 63].

Many existing drug sensitivity prediction algorithms do not take sample quality into consideration [53, 54] , which degrades their performance in the real world. Cell lines of poor quality exist, especially when cell lines are not screened against all of the compounds [55]. These existing approaches fail to remove the poor cell lines, which correspond to noisy samples in machine learning terms, and this failure leads to the degraded performance of machine learning algorithms [58].

1.2.4 Improving Drug Sensitivity Prediction via Transfer Learning

Cancer has a significant impact on public health worldwide and is the second leading cause of death in the US [60]. In 2016, the American Cancer Society predicts that 1,685,210 new cancer cases will be diagnosed, resulting in 595,690 deaths attributable to cancer in the US. Many of these cancer patients respond differently to the same cancer drug during chemotherapy. These response differences are attributable to not only environmental (i.e., external) factors such as tobacco, infectious organisms and an unhealthy diet, but also genetic (i.e., internal) factors such as inherited genetic mutations, hormones, immune conditions, and cancer cell heterogeneity, all of which make cancer drug discovery very difficult [45-49]. Because of the significant numbers of deaths associated with cancer, its study has attracted the attention of researchers from numerous domains including computational biology, machine learning, and data mining [50, 51, 63-65].

Traditional machine learning approaches to drug sensitivity prediction have been adopted to improve the performance of prediction algorithms. For example, Riddick *et al*. [66] presented an approach that employs random forests as a learning algorithm trained on gene expression signatures of selected cancer cell lines and the corresponding drug IC_{50} values (i.e., labels), to induce (i.e., learn) a model. The learned model is then applied to gene expression signatures of cancer cell lines in the test set, to yield drug sensitivity predictions. Geeleher *et al*. [54] proposed an approach to drug sensitivity prediction that works as follows. The input data consisted of baseline expressions with drug IC_{50} values in cell lines and in vivo tumor gene expression. The raw microarray data for the cell lines and clinical trials were processed separately and then combined and homogenized. The homogenized expression data consisted of cell lines expression data (i.e., baseline gene expression levels in the cell lines) and clinical trial expression data (i.e., baseline tumor expression data from clinical trials). A learning algorithm was applied to the training set (cell lines expression data along with the associated drug IC_{50} values for those cell lines), to learn a model. The resulting model was applied to the clinical trial expression data in the test set, to yield drug sensitivity predictions.

Costello *et al*. [53] assessed the performance of 44 drug sensitivity prediction algorithms based on genomic, proteomic, and epigenomic profiling data for 53 breast cancer cell lines. The training set consisted of several profiling data for 35 cell lines, where each cell line was associated with responses of 28 drugs. The test set consisted of profiling data for 18 cell lines. The drug response data (also called the ground truth) were hidden for evaluation purposes. The goal of each prediction algorithm is to induce (i.e., learn) a model from the training set and, then perform predictions on the test set. The predicted drug responses corresponded to a ranked list of the most sensitive (to be ranked first) to the most resistant (to be ranked last) cell lines for each drug across all the 18 cell lines in the test set. The top-performing approach worked by integrating several profiling data with improved representation combined with a probabilistic nonlinear regression model [53]. The second-best performing approach employed random forest regression to learn a model from profiling data of the training set and perform prediction on the test set. The algorithms' predictions were evaluated against the ground truth using a weighted probabilistic c-index (wpc-index) to report the final team rankings and resampled

Spearman correlations for verifying the consistency between team rankings [53]. The remaining prediction algorithms were not statistically different.

The previous approaches work well only under the common assumption: the training set and test set are in the same feature space and with the same distribution. However, this assumption does not hold in real-world applications [67]. As an example, consider the task of predicting drug sensitivity in multiple myeloma (referred to as the target task) where we have limited training data (called target training data). On the other hand, there exist an abundance of labeled auxiliary data in the task of predicting drug sensitivity in patients with a cancer type (referred to as the related task), where the auxiliary data are in a different feature space or come from a different distribution. In addition, collecting additional training data to improve the accuracy of prediction algorithms of the target task requires larger infrastructures and is associated with higher costs of screening size [56]. Therefore, there is a need to create high-performance prediction algorithms trained with more easily obtained data from a related task. This methodology is referred to as transfer learning [68].

CHAPTER 2

A NEW APPROACH TO LINK PREDICTION IN GENE REGULATORY NETWORKS

2.1 Introduction

Link prediction is an important data mining problem that has many applications in different domains such as social network analysis and computational biology. For example, biologists model gene regulatory networks (GRNs) as directed graphs where nodes are genes and links show regulatory relationships between the genes. By predicting links in GRNs, biologists can gain a better understanding of the cell regulatory circuits and functional elements. Existing supervised methods for GRN inference work by building a feature-based classifier from gene expression data and using the classifier to predict links in the GRNs.

Feature extraction is crucial in building efficient classifiers for link prediction [3, 69, 70]. This chapter presents a new supervised approach for link prediction in GRNs. The proposed approach employs both gene expression data and topological features extracted from the GRNs, in combination with three machine learning algorithms including random forests, support vector machines and neural networks. Experimental results on different datasets demonstrate the good performance of the proposed approach and its superiority over the existing methods.

2.2 Proposed Approach

2.2.1 Feature Extraction

Let $G=(V,E)$ be the directed graph that represents the topological structure of a gene regulatory network (GRN) where E is the set of edges or links and V is the set of nodes or vertices in G. The goal is to build a classifier that includes topological features alone or combined with gene expression data. There are totally sixteen topological features, which are described in detail below.

Node Degree. In considering node degrees, each directed edge $e=(u,v)\in E$ has four topological features, $indeg(u)$, $outdeg(u)$, $indeg(v)$, and $outdeg(v)$, which are defined as the number of edges entering u, leaving u, entering v, and leaving v, respectively.

Normalized Closeness Centrality. Normalized closeness centrality measures the closeness between a node and all other nodes in the graph G. For each node or vertex $v\in V$, the normalized closeness centrality $C(v^{in})$ is defined as

$$C(v^{in})=\frac{|V|-1}{\sum_{i\neq v} d(i,v)} \tag{2.1}$$

where $d(i,v)$, $i\neq v$, is the distance from $i\in V$ to $v\in V$, and $|V|$ is the number of vertices in the graph G [71]. The distance from i to v is the number of edges on the shortest path from i to v. If no such path exists, then the distance is set equal to ∞. Since G is a directed graph, the distance from i to v is not necessarily the same as the distance from v to i. The normalized closeness centrality $C(v^{out})$ is defined as

$$C(v^{out})=\frac{|V|-1}{\sum_{v\neq i} d(v,i)} \tag{2.2}$$

where $d(v,i)$, $v \neq i$, is the distance from $v \in V$ to $i \in V$. In considering normalized closeness centrality, each directed edge $e = (u,v) \in E$ has four topological features, $C(u^{in})$, $C(u^{out})$, $C(v^{in})$, and $C(v^{out})$.

Eccentricity. The eccentricity of a vertex $v \in V$ is the maximum distance between v and any other vertex $i \in V$ [72]. For each vertex $v \in V$, the eccentricity $\epsilon(v^{in})$ is defined as

$$\epsilon(v^{in}) = \max_{i \in V} d(i,v) \tag{2.3}$$

The eccentricity $\epsilon(v^{out})$ is defined as

$$\epsilon(v^{out}) = \max_{i \in V} d(v,i) \tag{2.4}$$

In considering eccentricity, each directed edge $e = (u,v) \in E$ has four topological features, $\epsilon(u^{in})$, $\epsilon(u^{out})$, $\epsilon(v^{in})$, and $\epsilon(v^{out})$.

Betweenness Centrality. Betweenness centrality measures the centrality of a vertex in the graph G [73]. For each vertex $v \in V$, the betweenness centrality of v, denoted $Between(v)$, is defined as

$$Between(v) = \sum_{i \neq v \neq j} \frac{\sigma_{i,j}(v)}{\sigma_{i,j}} \tag{2.5}$$

where $\sigma_{i,j}$ is the total number of shortest paths from vertex i to vertex j and $\sigma_{i,j}(v)$ is the total number of shortest paths from vertex i to vertex j that pass through v [71, 73]. In considering betweenness centrality, each directed edge $e = (u,v) \in E$ has two topological features, $Between(u)$ and $Between(v)$.

Eigenvector Centrality. Eigenvector centrality is another centrality measure where vertices in the graph have different importance. A vertex connected to a very important vertex is different from a vertex that is connected to a less important one. This concept is incorporated into eigenvector centrality [74]. For each vertex $v \in V$, the eigenvector centrality of v, denoted $Eigen(v)$, is defined as [75].

$$Eigen(v) = \frac{1}{\lambda} \sum_{i \in V} a_{v,i} Eigen(i) \tag{2.6}$$

where λ is a constant and $\mathbf{A} = (a_{v,i})$ is the adjacency matrix, i.e., $a_{v,i} = 1$ if vertex v is linked to vertex i, and $a_{v,i} = 0$ otherwise. The above eigenvector centrality formula can be rewritten in the matrix form as

$$\mathbf{Ax} = \lambda \mathbf{x} \tag{2.7}$$

where $\mathbf{x}$ is the eigenvector of the adjacency matrix $\mathbf{A}$ with the eigenvalue λ. In considering eigenvector centrality, each directed edge $e = (u, v) \in E$ has two topological features, $Eigen(u)$ and $Eigen(v)$.

2.2.2 Feature Vector Construction

Given n genes where each gene has p expression values. The gene expression profiles of these genes is denoted by $G \subseteq R^{n \times p}$, which contains n rows, each row corresponding to a gene, and p columns, each column corresponding to an expression value [8]. To train a classifier, the regulatory relationships among some genes have to be known. Suppose these regulatory relationships are stored in a matrix $H \subseteq R^{m \times 3}$. H contains m rows, where each row shows a known regulatory relationship between two genes, and three columns. The first column shows a transcription factor (TF). The second column shows a target gene. The third column shows the label, which is $+1$ if the TF is known to regulate the target gene or -1 if the TF is known not to regulate the target gene.

The matrix H represents a partially observed or known gene regulatory network for the n genes. If the label of a row in H is $+1$, then the TF in that row regulates the target gene in that row, and hence that row represents a link or edge of the network. If the label of a row in H is -1, then there is no link between the corresponding TF and target gene in that row.

Given a pair of genes g_1 and g_2 where the regulatory relationship between g_1 and g_2 is unknown, the goal is to use the trained classifier to predict the label of the gene pair. The predicted label is either $+1$ (i.e., a link is predicted to be present between g_1 and g_2) or -1 (i.e., a link is predicted to be missing between g_1 and g_2). Using biological terms, the present link means g_1 (transcription factor) regulates g_2 (target gene) whereas the missing link means g_1 does not regulate g_2.

To perform training and predictions, a feature matrix $D \subseteq R^{k \times 2p}$ is constructed with k feature vectors based on the gene expression profiles G. For a pair of genes g_1 and g_2, their feature vector d is created, which is stored in the feature matrix D, denoted by D_d and defined as

$$D_d = [g_1^1, g_1^2, \ldots, g_1^p, g_2^1, g_2^2, \ldots, g_2^p] \tag{2.8}$$

where $g_1^1, g_1^2, \ldots, g_1^p$ are the gene expression values of g_1, and $g_2^1, g_2^2, \ldots, g_2^p$ are the gene expression values of g_2.The above feature vector definition has been used by the existing supervised network inference methods [4, 5, 7, 8]. In the rest of this paper the above technique for constructing feature vectors is referred to as Ge, indicating that it is based on gene expression data only.

In addition, another feature matrix $D^{'} \subseteq R^{k \times 16}$ is constructed. Each feature vector d in the feature matrix $D^{'}$, denoted by $D_d^{'}$, is defined as

$$D_d^{'} = [t_1, t_2] \tag{2.9}$$

$$t_1 = indeg(g_1), C(g_1^{in}), \epsilon(g_1^{in}), outdeg(g_1), C(g_1^{out}), \epsilon(g_1^{out}), Between(g_1), Eigen(g_1) \tag{2.10}$$

$$t_2 = indeg(g_2), C(g_2^{in}), \epsilon(g_2^{in}), C(g_2^{out}), \epsilon(g_2^{out}), Between(g_2), Eigen(g_2), outdeg(g_2) \tag{2.11}$$

This feature vector construction technique is referred to as To, indicating that it is based on the sixteen topological features proposed in the paper.

Finally, the third feature matrix $D^{''} \subseteq R^{k \times (2p+16)}$ is constructed. Each feature vector d in the feature matrix $D^{''}$, denoted by $D_d^{''}$, contains both gene expression data and topological features, and is defined as

$$D_d^{''} = [g_1^1, g_1^2, ..., g_1^p, g_2^1, g_2^2, ..., g_2^p, t_1, t_2] \tag{2.12}$$

This feature vector construction technique is referred to as All, indicating that it is based on all the features described in the paper.

2.3 Experiments and Results

This section conducts a series of experiments to evaluate the performance of the approach and compare it with the existing methods for gene regulatory network (GRN) inference [76]. Below, datasets used in the study is described, experimental methodology, and the experimental results.

2.3.1 Datasets

GeneNetWeaver [77] is used to generate the datasets related to yeast and E. coli. We first built five different networks are first built and taken from yeast, where the networks contained 50, 100, 150, 200, 250 genes (or nodes) respectively. For each network, three files of gene expression data are generated. These files were labeled as knockouts, knockdowns and multifactorial, respectively. A knockout is a technique to deactivate the expression of a gene, which is simulated by setting the transcription rate of this gene to zero [7, 77]. A knockdown is a technique to reduce the expression of a gene, which is simulated by reducing the transcription rate of this gene by half [7, 77].Multifactorial perturbations are simulated by randomly increasing or decreasing the activation of the genes in a network simultaneously [77].

Table 2.1 presents details of the yeast networks, showing the number of nodes (edges, respectively) in each network. The edges or links in a network form positive examples. In addition, the same number of negative examples is randomly picked where each negative example corresponds to a missing link in the network. The networks and gene expression profiles for E. coli were generated similarly. Table 2.2 presents details of the networks generated from E. coli.

Table 2.1 Yeast Networks Used in the Experiments

Network	Directed	#Nodes	#Edges	#Pos examples	#Neg examples
Yeast 50	Yes	50	63	63	63
Yeast 100	Yes	100	281	281	281
Yeast 150	Yes	150	333	333	333
Yeast 200	Yes	200	517	517	517
Yeast 250	Yes	250	613	613	613

Table 2.2 E. Coli Networks Used in the Experiments

Network	Directed	#Nodes	#Edges	#Pos examples	#Neg examples
E. coli 50	Yes	50	68	68	68
E. coli 100	Yes	100	177	177	177
E. coli 150	Yes	150	270	270	270
E. coli 200	Yes	200	415	415	415
E. coli 250	Yes	250	552	552	552

2.3.2 Experimental Methodology

This section considers nine classification algorithms, denoted by RF+All, NN+All, SVM+All, RF+Ge, NN+Ge, SVM+Ge, RF+To, NN+To, SVM+To, respectively. Table 2.3 lists these algorithms and their abbreviations. RF+All (RF+Ge, RF+To, respectively) represents the random forest algorithm combined with all features including both gene expression data and topological features (RF combined with only gene expression data, RF combined with only topological features, respectively). NN+All (NN+Ge, NN+To, respectively) represents the neural network algorithm combined with all features (NN combined with only gene expression data, NN combined with only topological features, respectively). SVM+All (SVM+Ge, SVM+To, respectively) represents the support vector machine algorithm combined with all features (SVM combined with only gene expression data, SVM combined with only topological features, respectively). SVM+Ge is adopted by the existing supervised network inference methods [4, 5, 7, 8].

Table 2.3 Nine Classification Algorithms and Their Abbreviations

Abbreviation	Classification algorithm and features
RF + All	Random Forests with all features
NN + All	Neural Networks with all features
SVM + All	Support Vector Machines with all features
RF + Ge	Random Forests with gene expression features
NN + Ge	Neural Networks with gene expression features
SVM + Ge	Support Vector Machines with gene expression features
RF + To	Random Forests with topological features
NN + To	Neural Networks with topological features
SVM + To	Support Vector Machines with topological features

Software used in this work included: the random forest package in R [78], the neuralnet package in R [79], and the SVM with linear kernel in the LIBSVM package [80]. We used R to write some utility tools for performing the experiments, and employed the package, igraph, to extract topological features from a network [81] .

The performance of each classification algorithm was evaluated through 10-fold cross validation. The size of each fold was approximately the same, and each fold contained the same number of positive and negative examples. On each fold, the *balanced error rate* [76, 82] (BER) of a classification algorithm was calculated where the BER is defined as

$$BER = \frac{1}{2} \times \left(\frac{FN}{TP + FN} + \frac{FP}{FP + TN} \right) \tag{2.13}$$

FN is the number of false negatives (i.e., present links that were mistakenly predicted as missing links).TP is the number of true positives (i.e., present links that were correctly predicted as present links). FP is the number of false positives (i.e., missing links that were mistakenly predicted as present links). TN is the number of true negatives (i.e., missing links that were correctly predicted as missing links). For each algorithm, the mean BER, denoted MBER, over 10 folds was computed and recorded. The lower MBER an algorithm has, the better performance that algorithm achieves. Statistically significant performance differences between classification algorithms were calculated using Wilcoxon signed rank tests [76, 83] .As in [84, 85], p-values below 0.05 are considered to be statistically significant.

2.3.3 Experimental Results

Table 2.4 shows the MBERs of the nine classifications on the fifteen yeast datasets used in the experiments. For each dataset, the algorithm having the best performance (i.e., with the lowest MBER) is in boldface. Table 2.5 shows, for each yeast dataset, the p-values of Wilcoxon signed rank tests between the best algorithm, represented by '-', and the other algorithms. A p-value in boldface ($p \leq 0.05$) indicates that the corresponding result is significant. It can be seen from Table 2.4 that random forests performed better than support vector machines and neural networks. In particular, random forests combined with all features (i.e., RF+All) performed the best on 10 out of 15 yeast datasets. For the other five yeast datasets, RF+All was not statistically different from the best algorithms according to Wilcoxon signed rank tests ($p > 0.05$); cf. Table 2.5.

Table 2.6 shows the MBERs of the nine classification algorithms on the fifteen E. coli datasets used in the experiments. Table 2.7 shows, for each E. coli dataset, the p-values of Wilcoxon signed rank tests between the best algorithm, represented by '-', and the other algorithms. It can be seen from Table 2.6 that random forests combined with topological features (i.e., RF+To) performed the best on 6 out of 15 E. coli datasets. For the other nine E. coli datasets, RF+To was not statistically different from the best algorithms according to Wilcoxon signed rank tests ($p > 0.05$); cf. Table 2.7.

Table 2.4 MBERs of Nine Classification Algorithms on Fifteen Yeast Datasets

Dataset	RF+ All	NN+ All	SVM+ All	RF+ Ge	NN+ Ge	SVM+ Ge	RF+ To	NN+ To	SVM+ To
Yeast 50 knockouts	16.6	**15.0**	20.0	18.3	15.8	20.0	17.7	16.3	19.4
Yeast 50 knockdowns	**16.6**	17.5	20.0	19.1	22.7	16.9	17.7	18.8	19.4
Yeast 50 multifactorial	**16.1**	17.5	20.8	15.5	24.7	17.2	17.7	16.6	19.4
Yeast 100 knockouts	12.8	13.7	17.3	14.4	20.0	16.7	**11.6**	22.2	14.5
Yeast 100 knockdowns	14.2	14.9	15.2	14.1	29.4	14.1	**11.8**	17.5	14.5
Yeast 100 multifactorial	12.0	17.8	18.0	12.3	21.3	18.4	**11.4**	20.0	14.5
Yeast 150 knockouts	**5.10**	10.7	14.1	5.40	10.4	13.6	6.10	15.4	11.0
Yeast 150 knockdowns	**5.00**	10.5	10.4	5.80	14.7	10.9	5.80	16.0	11.0
Yeast 150 multifactorial	**4.10**	12.4	13.9	**4.10**	15.1	16.1	5.80	10.8	11.0
Yeast 200 knockouts	**1.90**	4.00	5.30	**1.90**	4.90	5.60	2.50	13.7	3.70
Yeast 200 knockdowns	**1.90**	5.10	5.80	**1.90**	6.30	10.5	2.70	9.80	3.70
Yeast 200 multifactorial	**1.90**	6.80	10.1	**1.90**	4.70	14.3	2.50	5.50	3.70
Yeast 250 knockouts	**3.80**	8.40	7.60	4.10	5.50	7.20	7.40	10.6	10.4
Yeast 250 knockdowns	**4.00**	9.70	7.40	**4.00**	6.20	7.30	8.10	7.70	10.4
Yeast 250 multifactorial	4.00	8.50	8.50	**3.90**	6.00	9.00	7.50	11.3	10.4

Table 2.5 P-Values of Wilcoxon Signed Rank Tests between the Best Algorithm, Represented by '-', and the Other Algorithms for Each Yeast Dataset

Dataset	RF+ All	NN+ All	SVM+ All	RF+ Ge	NN+ Ge	SVM+ Ge	RF+ To	NN+ To	SVM+ To
Yeast 50 knockouts	0.75	-	0.28	0.44	0.67	0.07	0.46	0.78	0.27
Yeast 50 knockdowns	-	0.79	0.46	0.17	0.13	0.52	0.58	0.86	0.33
Yeast 50 multifactorial	-	0.7	0.28	0.89	0.07	0.68	0.58	0.68	0.33
Yeast 100 knockouts	0.34	0.44	**0.05**	0.14	**0.01**	**0.02**	-	**0.04**	0.15
Yeast 100 knockdowns	0.22	0.24	0.16	0.22	**0.00**	**0.03**	-	0.08	0.12
Yeast 100 multifactorial	0.46	0.13	**0.02**	0.27	**0.01**	**0.04**	-	**0.00**	0.12
Yeast 150 knockouts	-	**0.01**	**0.02**	1.00	**0.01**	**0.01**	0.10	**0.01**	**0.02**
Yeast 150 knockdowns	-	**0.02**	**0.02**	0.17	**0.03**	**0.01**	0.17	**0.00**	**0.02**
Yeast 150 multifactorial	-	**0.01**	**0.01**	-	**0.01**	**0.01**	0.17	**0.01**	**0.02**
Yeast 200 knockouts	-	**0.01**	**0.01**	-	**0.01**	**0.02**	0.50	**0.01**	0.17
Yeast 200 knockdowns	-	**0.04**	**0.01**	-	**0.01**	**0.02**	0.10	**0.01**	0.17
Yeast 200 multifactorial	-	**0.04**	**0.01**	-	**0.06**	**0.00**	0.50	**0.01**	0.17
Yeast 250 knockouts	-	**0.03**	0.20	0.17	0.06	0.10	0.17	0.09	**0.02**
Yeast 250 knockdowns	-	**0.01**	**0.10**	-	0.06	**0.05**	0.07	0.20	**0.05**
Yeast 250 multifactorial	1.00	**0.05**	0.11	-	**0.01**	**0.03**	0.18	**0.05**	**0.03**

Table 2.6 MBERs of Nine Classification Algorithms on Fifteen E. Coli Datasets

Dataset	RF+ All	NN+ All	SVM+ All	RF+ Ge	NN+ Ge	SVM+ Ge	RF+ To	NN+ To	SVM+ To
E. coli 50 knockouts	14.5	5.70	9.80	18.8	22.0	21.5	**5.00**	11.6	18.4
E. coli 50 knockdowns	14.5	9.50	10.7	19.1	19.0	16.7	**5.00**	10.8	18.4
E. coli 50 multifactorial	15.3	10.3	8.20	15.7	22.9	18.0	**5.00**	10.3	18.4
E. coli 100 knockouts	10.3	9.50	14.2	12.0	14.4	17.7	**5.60**	6.00	13.6
E. coli 100 knockdowns	10.6	11.6	14.2	11.4	14.1	18.0	**5.40**	9.80	13.6
E. coli 100 multifactorial	9.80	10.4	11.4	9.80	12.4	13.6	**7.10**	9.90	13.6
E. coli 150 knockouts	**2.40**	4.40	2.90	**2.40**	8.80	5.90	2.70	5.90	3.30
E. coli 150 knockdowns	**2.20**	4.40	2.70	**2.20**	8.10	3.70	2.70	3.80	3.30
E. coli 150 multifactorial	**2.20**	3.80	2.40	**2.20**	8.70	2.40	2.70	3.30	3.30
E. coli 200 knockouts	5.50	5.50	5.10	5.50	**4.60**	**4.60**	6.40	11.0	6.00
E. coli 200 knockdowns	5.30	5.00	5.80	6.10	**4.40**	5.50	6.40	8.50	6.00
E. coli 200 multifactorial	5.00	4.20	3.90	4.90	**2.80**	3.40	6.40	8.00	6.00
E. coli 250 knockouts	**6.60**	10.3	7.00	7.70	8.80	8.00	6.30	12.7	10.0
E. coli 250 knockdowns	5.30	9.30	5.90	**5.10**	6.70	7.50	6.20	11.9	10.0
E. coli 250 multifactorial	5.40	11.1	8.00	**5.20**	11.0	9.70	6.30	10.6	10.0

Table 2.7 P-Values of Wilcoxon signed Rank Tests between the Best Algorithm, Represented by '-', and the Other Algorithms for Each E. Coli Dataset

Dataset	RF+ All	NN+ All	SVM+ All	RF+ Ge	NN+ Ge	SVM+ Ge	RF+ To	NN+ To	SVM+ To
E. coli 50 knockouts	0.06	1.00	0.46	**0.03**	**0.00**	**0.01**	-	**0.10**	**0.04**
E. coli 50 knockdowns	0.06	0.07	0.46	**0.01**	**0.01**	**0.01**	-	0.46	**0.04**
E. coli 50 multifactorial	0.06	0.49	0.46	0.08	**0.01**	**0.04**	-	**0.04**	**0.04**
E. coli 100 knockouts	0.08	0.04	**0.01**	**0.01**	**0.00**	**0.01**	-	0.68	**0.04**
E. coli 100 knockdowns	0.02	0.16	**0.01**	**0.01**	**0.01**	**0.01**	-	0.27	**0.04**
E. coli 100 multifactorial	0.67	0.22	0.17	0.67	0.11	0.06	-	0.79	0.17
E. coli 150 knockouts	-	0.10	0.25	-	**0.02**	0.06	0.65	**0.04**	0.65
E. coli 150 knockdowns	-	**0.04**	0.17	-	0.06	0.06	1.00	0.1.00	1.00
E. coli 150 multifactorial	-	0.06	1.00	-	0.10	**1.00**	1.00	0.100	1.00
E. coli 200 knockouts	0.91	0.50	0.46	0.91	0.68	-	0.41	**0.03**	0.46
E. coli 200 knockdowns	0.89	0.50	0.27	0.75	-	0.46	0.46	0.11	0.68
E. coli 200 multifactorial	0.67	0.09	0.14	0.67	-	0.28	0.13	0.13	0.20
E. coli 250 knockouts	-	**0.02**	0.23	0.65	**0.00**	0.12	0.71	0.12	**0.05**
E. coli 250 knockdowns	0.17	**0.02**	**0.02**	-	**0.02**	**0.02**	0.10	**0.02**	**0.01**
E. coli 250 multifactorial	1.00	**0.02**	**0.01**	-	**0.01**	**0.01**	0.72	**0.01**	**0.02**

These results show that using random forests with the proposed topological features alone or combined with gene expression data performed well. In particular, the RF+All algorithm achieved the best performance on 14 out of all 30 datasets. This is far better than the SVM+Ge algorithm used by the existing supervised network inference methods [4, 5, 7, 8] which achieved the best performance on one dataset only (i.e., the E. coli 200 knockouts dataset in Table 2.6).

It is worth pointing out that, for a fixed dataset size (e.g., 200), the SVM+To algorithm always yielded the same mean balanced error rate (MBER) regardless of which technique (knockout, knockdown or multifactorial) was used to generate the gene expression profiles. This happens because these different gene expression profiles correspond to the same network, and SVM+To uses only the topological features extracted from the network without considering the gene expression data. On the other hand, due to the randomness introduced in random forests and neural networks, RF+To and NN+To yielded different MBERs even for the same network.

2.4 Summary

This chapter presents a new approach to network inference through link prediction with topological features. The experimental results showed that using the topological features alone or combined with gene expression data performs better than the existing network inference methods that use only gene expression data. This work assumes that there are exactly the same number of positive examples (i.e., links that are present) and negative examples (i.e., links that are missing) in the datasets. In many biological networks, however, negative datasets (majority class) are usually much larger than positive datasets (minority class).

CHAPTER 3

A LEARNING FRAMEWORK TO IMPROVE UNSUPERVISED GENE NETWORK INFERENCE

3.1 Introduction

Network inference through link prediction is an important data mining problem that finds many applications in computational social science and biomedicine. For example, by predicting links, i.e., regulatory relationships, between genes to infer gene regulatory networks (GRNs), computational biologists gain a better understanding of the functional elements and regulatory circuits in cells. Unsupervised methods have been widely used to infer GRNs; however, these methods often create missing and spurious links. This chapter proposes a learning framework to improve the unsupervised methods. Given a network constructed by an unsupervised method, the proposed framework employs a graph sparsification technique for network sampling and principal component analysis for feature selection to obtain better quality training data, which guides three classifiers to predict and clean the links of the given network. The three classifiers include neural networks, random forests and support vector machines. Experimental results on several datasets demonstrate the good performance of the proposed learning framework and the classifiers used in the framework.

3.2 The Learning Framework

3.2.1 Graph Sparsification

The input of the proposed learning framework is a weighted directed graph $G = (V, E)$ that represents the topological structure of (a subgraph of) the gene regulatory network

(GRN) constructed by Inferelator based on a time series gene expression dataset. E is the set of edges or links, and V is the set of vertices or nodes in G,where each link represents a regulatory relationship and each node represents a gene. Each edge $e=(u,v)\in E$ is associated with a weight, denoted by $W(e)$, where $0<W(e)\leq 1$.

The proposed graph sparsification method, named GeneProbe (reminiscent of LinkProbe [86] for social network analysis), takes as input the graph G and two genes of interest: an origin or regulator gene, and a destination or regulated gene. GeneProbe creates six sets of genes, described below, and produces as output an inference subgraph that contains all genes in the six sets and all edges in E that connect the genes in the six sets.

Two sets of k-backbone genes. These include one set of k-backbone hub genes and one set of k-backbone authority genes. The k-backbone hub genes include all genes whose weighted outgoing degree is greater than or equal to a user-specified positive real value $k_{hub}\in\mathbb{R}^{+}$. The weighted outgoing degree of a gene or node u is defined as the sum of edge weights for all outgoing edges of u. Likewise, the k-backbone authority genes include all genes whose weighted incoming degree is greater than or equal to a user-specified value $k_{authority}\in\mathbb{R}^{+}$. The weighted incoming degree of a node u is defined as the sum of edge weights for all incoming edges of u. (In the study presented here, k_{hub} = 15 and $k_{authority}$ = 10.) Intuitively few ``highly social" individuals who would represent ``social hubs/authorities" for inference across geographical regions are selected. The genes most likely to be regulators (with the largest weighted outgoing degrees) are selected as the ``hubs" of the network $G$ for inclusion in the inference subgraph. Furthermore, the genes most likely to be regulated genes (with the largest weighted incoming degrees) are selected as the ``authorities" of the network G for inclusion in the inference subgraph.

Formally, let $W-out(u)$ ($W-in(u)$, respectively) denote the weighted outgoing (incoming, respectively) degree of node *u*. Then

$$W-out(u) = \sum_{e \in E-out(u)} W(e) \tag{3.1}$$

$$W-in(u) = \sum_{e \in E-in(u)} W(e) \tag{3.2}$$

where $W(e)$ is the weight of edge e, and $E-out(u)$ ($E-in(u)$, respectively) denotes the set of edges leaving (entering, respectively) u. GeneProbe retrieves all genes $u \in G$ where $W-out(u) \geq k_{hub}$ and $W-in(u) \geq k_{authority}$.

Two sets of d -local genes. These include one set of d -local genes for the origin and one set of d -local genes for the destination. The d -local genes are the genes adjacent to each of the two genes of interest, i.e., the origin and destination, with incident edge weights greater than or equal to a user-specified positive real value $d \in \mathbb{R}^+$. (In the study presented here, d = 0.95.)Intuitively, the d -local genes represent the genes most likely to be regulated by and most likely to regulate the two genes of interest.

Two sets of random walk metropolis genes. These include one set of random walk metropolis genes for the origin and one set of random walk metropolis genes for the destination. The random walk metropolis (RWM) genes provide a stochastic path from the genes of interest back to a k -backbone gene (if possible). The RWM does not differentiate between k -backbone hub and k -backbone authority genes. All of the genes encountered along the RWM path are added to the inference subgraph. For the origin or regulator gene, the random walk is a walk along outgoing edges towards the k-backbone, whereas the random walk for the destination or regulated gene is a backtrack to the k-backbone along incoming edges. Each step along the random walk metropolis is selected based on a randomized chance until a k-backbone gene is reached (or a maximum number of tries is exceeded).

The randomized chance at each step along the random walk for the regulator gene (i.e., origin) can be characterized as follows. Given a current gene u, a random edge from the list of outgoing edges of gene u is selected. Let w represent the gene at the end of the randomly selected outgoing edge. The random walk will proceed from gene u to gene w if a randomly selected number between 0 and 1 is less than or equal to the minimum of 1 and the weighted outgoing degree of w divided by the weighted outgoing degree of u. That is, w is accepted as the next state with the probability of less than or equal to an acceptance rate α_{out}. Otherwise, another random outgoing edge of gene u is selected and similar calculations are performed. This move can be formalized in Equation (3.3). $P(u \rightarrow w)$ is the probability that a random walk proceeds from u to w where

$$\alpha_{out} = P(u \rightarrow w) = min\left\{1, \frac{W-out(w)}{W-out(u)}\right\} \tag{3.3}$$

This process is repeated until a maximum number of tries is reached (or a k-backbone gene is reached). Note that given enough chances in a connected gene regulatory network, the random walks will always reach a k-backbone gene. It logically follows that a setting that includes few k-backbone genes will likely generate many RWM genes and vice versa.

The randomized chance at each step along the random walk for the regulated gene (i.e., destination) can be characterized as follows. Given a current gene v, a random edge from the list of incoming edges of gene v is selected. Let w represent the gene at the end of the randomly selected incoming edge. The random walk will backtrack from gene v to gene w if a randomly selected number between 0 and 1 is less than or equal to the minimum of 1 and the weighted incoming degree of w divided by the weighted incoming degree of v. That is, w is accepted as the next state with the probability of less than or equal to an acceptance rate α_{in}. Otherwise, another random incoming edge of gene v is selected and similar calculations are performed. This move is formalized in Equation (3.4).

$$\alpha_{in} = P(w \leftarrow v) = min\left\{1, \frac{W-in(w)}{W-in(v)}\right\} \tag{3.4}$$

where $P(w \leftarrow v)$ is the probability that a random walk moves backward from v to w. This process is repeated until a maximum number of tries is reached (or a k-backbone gene is reached). Figure 3.1 illustrates an inference subgraph.

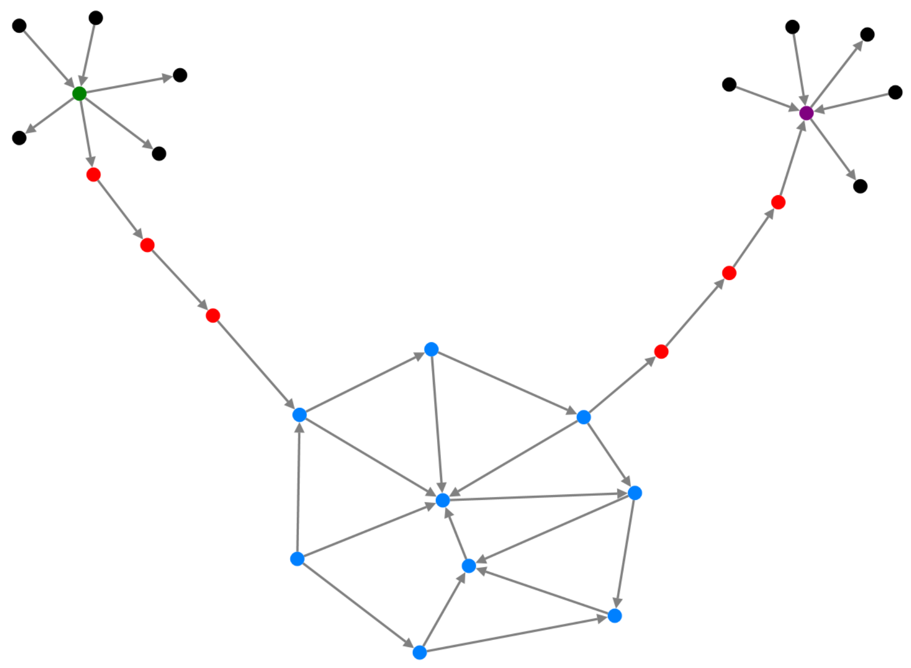


Figure 3.1 Example of an inference subgraph containing an origin (green), a destination (purple), d -local genes (black), k -backbone genes (blue) and random walk metropolis genes (red).

3.2.2 Feature Selection

An inference subgraph may still have missing and spurious links. We select a more reliable sample from the inference subgraph where the weight of each edge in the sample is greater than or equal to 0.5. For each pair of genes u , v in the sample graph, a feature vector B is created by concatenating the gene expression profiles of u and v, as in [4, 16]. That is,

$$B=[u^1,u^2,\ldots,u^p,v^1,v^2,\ldots,v^p] \tag{3.5}$$

where $u^1, u^2, \dots, u^p$ are the gene expression values of u, and $v^1, v^2, \dots, v^p$ are the gene expression values of v. Each gene expression value is a feature.

We employ principal component analysis (PCA) to reduce the dimensionality of the feature vectors of a sample graph [87]. Specifically, the feature vectors are combined into a $2p \times N$ matrix X where $2p$ is the total number of features and N is the number of links in the sample graph. Let the rank of the matrix X be r where the rank represents the maximum number of uncorrelated column vectors in X [88]. We represent X through singular value decomposition (SVD) as

$$X = U \cdot S \cdot V^T \tag{3.6}$$

Both U and V are orthogonal matrices. Each column of U is one of the eigenvectors of the covariance matrix $X \cdot X^T$ where X^T is the transpose of X. Each column of V is one of the eigenvectors of the matrix $X^T \cdot X$. The $r \times r$ matrix S contains eigenvalues of X on the diagonal line of S.

In this case, each column vector of the matrix X representsa (present or missing) link of the sample graph. That is, each link of the sample graph is a vector in the $2p-$dimensional Euclidean space. The dot product between two column vectors reflects the extent to which the two corresponding links share similar feature occurrences. Thus, the dot product can be used to get pairwise link distances. Let M contain pairwise link distances, i.e., M_{ij} is the dot product distance between link L_i and link L_j. Then M can be derived by:

$$M = X^T \cdot X \tag{3.7}$$

which can be generalized as:

$$M = (U \cdot S \cdot V^T)^T \cdot (U \cdot S \cdot V^T) \tag{3.8}$$
$$= (V \cdot S \cdot U^T) \cdot (U \cdot S \cdot V^T) \tag{3.9}$$
$$= V \cdot S^2 \cdot V^T \tag{3.10}$$
$$= (V \cdot S) \cdot (V \cdot S)^T \tag{3.11}$$

The new representation of the matrix M shows that the pairwise link comparison matrix M can be obtained through the dot product of $(V \cdot S)$ and $(V \cdot S)^T$. That is, the ith row of the $N \times r$ matrix $(V \cdot S)$ is an r-dimensional vector representing the ith link in the sample graph. This result indicates that after performing projection transformation with respect to the matrix V, pairwise distances are kept between link-vectors as in the original setting.

SVD reduces the dimensionality (from $2p$ to r where $r < min(2p, N)$) of a link-vector. However, the reduced r-dimensional link-vector might still contain redundant dimensions. In practice, the dimensionality of these link-vectors could be further reduced without losing characteristics of the link-vectors in the original $2p-$dimensional Euclidean space. Specifically, based on the optimal solution of the squared-error criterion of PCA [87], an r-dimensional vector could be projected onto a k-dimensional subspace, $k < r$, spanned by the eigenvectors corresponding to the k largest eigenvalues of the covariance matrix $X \cdot X^T$. As a result, X_k can be obtained, which is an approximation of the original matrix X, by keeping the k largest eigenvalues of the covariance matrix $X \cdot X^T$ and replacing the remaining eigenvalues with zeros. Then, Equation (3.6) can be rewritten as

$$X_k = U_k \cdot S_k \cdot V_k^T \tag{3.12}$$

Therefore, the $N \times k$ matrix $(V_k \cdot S_k)$ replaces the matrix $(V \cdot S)$ in Equation (3.11), where the ith row of the matrix $(V_k \cdot S_k)$ is a k dimensional vector that represents the ith link in the sample graph.(In the study presented here, k = 10.)

3.2.3 The Link Prediction Algorithm

After explaining the concepts of graph sparsification and feature selection, the proposed learning framework (i.e., link prediction algorithm) is now described as how it works. The main assumption here is that the network G constructed by Inferelator is not accurate, and there are many missing and spurious links in G. A missing link or edge e_m refers to a regulatory relationship that exists in the ground truth but is not inferred by Inferelator, and hence $e_m \notin G$. A spurious link e_s refers to a regulatory relationship that does not exist in the ground truth, but is inferred by Inferelator, and hence $e_s \in G$. The goal here is for the link prediction algorithm to detect these missing and spurious links, so as to clean them. To achieve this goal, the algorithm predicts whether there is a link between two nodes and uses the predicted outcome to replace the result obtained from Inferelator if the predicted outcome differs from Inferelator's result.

Let $G = (V, E)$ be the gene regulatory network (GRN) constructed by Inferelatorbased on a time series gene expression dataset. The proposed algorithm first creates two subgraphs $G_+ = (V_+, E_+)$ and $G_- = (V_-, E_-)$ where V_+ (V_-, respectively) contains incident nodes of the edges in E_+ (E_-, respectively),the weight of each edge in E_+ (E_-, respectively) is greater than or equal to (less than, respectively) the median of the weights of the edges in E, $E_+ \cap E_- = \varnothing$ and $E_+ \cup E_- = E$. Thus, the edges in G_+ are likely to be positive instances and the edges in G_- are likely to be negative instances. Note, however, that in practice these two subgraphs G_+ and G_- have low quality data,i.e., they contain many missing and spurious links.

The proposed link-prediction algorithm consists of five steps.

Step 1: Suppose the algorithm aims to predict whether there is a link from node/gene u to node/gene v. There are two cases to consider. In case 1, the gene pair (u,v) is in G_+. Then the algorithm creates an inference subgraph $I_+ \subseteq G_+$ by invoking GeneProbe and using G_+, the origin u and the destination v as input. In addition, the algorithm randomly selects a pair of genes x, y in G_-, and creates an inference subgraph $I_- \subseteq G_-$ by invoking GeneProbe and using G_-, the origin x and the destination y as input. In case 2, the gene pair (u,v) is in G_-. Then the algorithm creates an inference subgraph $I_- \subseteq G_-$ by invoking GeneProbe and using G_-, the origin u and the destination v as input. In addition, the algorithm randomly selects a pair of genes x, y in G_+, and creates an inference subgraph $I_+ \subseteq G_+$ by invoking GeneProbe and using G_+, the origin x and the destination y as input. Without loss of generality, case 1 is assumed to hold and case 1 will be used to describe the following steps. Thus, the algorithm creates dual graph sparsifications I_+ and I_- in step 1.

Step 2: Create a sample graph $I_+^{'} \subseteq I_+$. where $I_+^{'}$ does not contain the testing gene pair (u,v), and the weight of each edge in $I_+^{'}$ is greater than or equal to 0.5. We consider the edges in $I_+^{'}$ to have higher quality and are more likely to be positive instances. Suppose there are K edges in $I_+^{'}$. We then randomly select K edges from I_- and use the randomly selected edges to form a sample graph $I_-^{'} \subseteq I_-$. In training the three classifiers including neural networks, random forests and support vector machines, the edges in $I_+^{'}$ are used as positive training examples, and use the edges in $I_-^{'}$ as negative training examples. The dual sample graphs $I_+^{'}$ and $I_-^{'}$ together form the training dataset.

Step 3: Construct a feature vector for the testing gene pair (u,v) by concatenating the gene expression values of u and v, as shown in Equation (3.5). Also, construct a feature vector for each gene pair (p,q) in the training dataset by concatenating the gene expression values of p and q.

Step 4: Reduce the dimensionality of the feature vectors constructed in step 3 using principal component analysis (PCA), as described in Section 3.2.2.

Step 5: Use the training examples (reduced feature vectors obtained from step 4) to train three classifiers including neural networks, random forests and support vector machines. Use the trained models to predict whether there is a link from gene u to gene v.

3.3 Experiments and Results

We conducted a series of experiments to evaluate the performance of the proposed learning framework and the three classifiers used in the framework. Below, the datasets and experimental methodology used in the study are described, and then the experimental results are presented.

3.3.1 Datasets

We adopted the five time-series gene expression datasets available in the DREAM4 100-gene *insilico* network inference challenge [21, 77, 89, 90]. Each dataset contains 10 times series, where each time series has 21 time points, for 100 genes. Each gene has $(10 \times 21) = 210$ gene expression values. Each link consists of two genes, and hence is represented by a 420-dimensional feature vector; cf. Equation (3.5). Through principal component analysis, each reduced feature vector has only 10 dimensions.

Each time-series dataset is associated with a gold standard file, where the gold standard represents the ground truth of the network structure for the time-series data. Each link in the gold standard represents a true regulatory relationship between two genes. For a given time-series dataset, Inferelator [27] constructs a directed network, in which each link has a weight and represents an inferred regulatory relationship between two genes.

Table 3.1 presents details of the five networks, true and inferred, used in the experiments. The table shows the numbers of true present and missing links in each gold standard, and the numbers of inferred present and missing links in each network constructed by Inferelator. Each network contains 100 nodes or genes, which form 9,900 ordered gene pairs totally.

Table 3.1 Networks Used in the Experiments

	Net1	Net2	Net3	Net4	Net5
Directed	Yes	Yes	Yes	Yes	Yes
Nodes	100	100	100	100	100
True present links	176	249	195	211	193
True missing links	9,724	9,651	9,705	9,689	9,707
Inferred present links	6,232	6,066	6,186	5,930	6,180
Inferred missing links	3,668	3,834	3,714	3,970	3,720

For each network, four sets of testing data are created. Each testing dataset contains 50 randomly selected links from the gold standard. Among the 50 links, 25 are true present links and 25 are true missing links in the gold standard. The label (+1 or -1) of each selected link is known, where +1 represents a true present link and -1 represents a true missing link. These testing data were excluded from the training datasets used to train the classifiers studied in the paper. There were 20 testing datasets totally.

3.3.2 Experimental Methodology

Three classification algorithms are considered, namely neural networks (NN), random forests (RF) and support vector machines (SVM). Software used in this work included: the neuralnet package in R [79], the random forest package in R [78], and the SVM program with the polynomial kernel of degree 2 in the LIBSVM package [80]. The principal component analysis (PCA) program was based on the prcomp function in R [91]. The graph sparsification method (GeneProbe) was implemented in C++. In addition, R was used to write some utility tools for performing the experiments.

The performance of each classification algorithm was evaluated as follows. Each classification algorithm was trained as described in Section 3.2.3. For each link in a testing dataset, the trained model is used to predict its label. In evaluating the link prediction algorithm, a true positive is defined to be a true present link that is predicted as a present link. A false positive is a true missing link that is predicted as a present link. A true negative is a true missing link that is predicted as a missing link. A false negative is a true present link that is predicted as a missing link. In evaluating Inferelator, a true positive is defined to be a true present link that is an inferred present link. A false positive is a true missing link that is an inferred present link. A true negative is a true missing link that is an inferred missing link. A false negative is a true present link that is an inferred missing link. Let *TP* (*FP*, *TN*, *FN*, respectively) denote the number of true positives (false positives, true negatives, false negatives, respectively) for a testing dataset. The balanced error rate (*BER*) was adopted [16], defined as

$$BER = \frac{1}{2} \times \left(\frac{FN}{TP + FN} + \frac{FP}{FP + TN} \right) \tag{3.13}$$

Each classification algorithm was applied to each testing dataset and recorded the *BER* for that testing dataset. The lower *BER* a classification algorithm has, the better performance that algorithm achieves. The *improvement rate*, denoted *IR*, is defined on a testing dataset to be $(P^* - P) \times 100\%$ where P^* is the *BER* of Inferelator and *P* is the *BER* of a classification algorithm (NN, RF or SVM) for that dataset. Statistically significant performance differences were calculated using Wilcoxon signed rank tests [83] . As in [87], p-values below 0.05 are considered to be statistically significant.

3.3.3 Experimental Results

Table 3.2 shows the improvement rate (*IR*) each classification algorithm achieves on each of the 20 testing datasets. A positive (negative, respectively) *IR* for a classification algorithm indicates that the algorithm performs better (worse, respectively) than Inferelator. The larger the positive *IR* a classification algorithm has, the more improvement over Inferelator that algorithm achieves. For each testing dataset, the classification algorithm with the best performance, i.e., the largest positive *IR*, is shown in boldface. It can be seen from Table 3.2 that SVM (support vector machines) outperforms Inferelator, NN (neural networks) and RF (random forests). SVM improves

Inferelator on 16 testing datasets, and the improvement is statistically significant according to Wilcoxon signed rank tests ($p < 0.05$). NN improves Inferelator on 12 testing datasets; however, the improvement is not statistically significant according to Wilcoxon signed rank tests ($p > 0.05$).

Table 3.2 Improvement Rates of Three Classification Algorithms on Twenty Datasets

Dataset	NN	RF	SVM
Net1.test1	+11.10%	+0.90%	**+17.90%**
Net1.test2	**+17.00%**	+1.80%	+12.30%
Net1.test3	+7.10%	+4.90%	**+13.60%**
Net1.test4	+5.20%	+2.10%	**+9.20%**
Net2.test1	**+5.80%**	+1.80%	-1.60%
Net2.test2	**+14.80%**	-6.20%	+0.90%
Net2.test3	**+4.40%**	-12.10%	+1.20%
Net2.test4	-0.60%	+1.20%	**+5.20%**
Net3.test1	**+0.20%**	-9.40%	-1.10%
Net3.test2	+0.00%	**+8.00%**	+4.00%
Net3.test3	-8.00%	-6.00%	**+2.00%**
Net3.test4	-2.00%	-10.00%	**+8.00%**
Net4.test1	**+10.40%**	-4.80%	+3.00%
Net4.test2	-5.70%	-6.00%	**+4.50%**
Net4.test3	**+10.80%**	+2.50%	+5.20%
Net4.test4	-3.00%	**+8.20%**	+2.60%
Net5.test1	+1.20%	-5.00%	**+7.10%**
Net5.test2	-1.70%	-13.10%	**+2.90%**
Net5.test3	-7.00%	-13.30%	-3.50%
Net5.test4	-3.00%	-7.20%	-1.10%

Experiments were also carried out to evaluate the performance of different SVM kernel functions, including the linear kernel (SVM_L), polynomial kernel of degree 2 (SVM_P2), polynomial kernel of degree 15 (SVM_P15), Gaussian kernel (SVM_G), and sigmoid kernel (SVM_S). Figure 3.2 shows the *BER* values, averaged over the 20 testing datasets, for the different kernel functions. It can be seen that SVM with the polynomial kernel of degree 2 (SVM_P2) used in this study performs the best.

Finally, experiments are conducted to evaluate the effectiveness of the components of the proposed learning framework. There are two core components: graph sparsification (GeneProbe) and feature selection (PCA). Figure 3.3 compares the approach with graph sparsification (GS) only, the approach with feature selection (FS) only, and the proposed approach, which combines both graph sparsification (GS) and feature selection (FS). Each bar represents the average *BER* over the 20 testing datasets. The classifier used to generate the results was the SVM program with the polynomial kernel of degree 2. It can be seen from Figure 3.3 that the proposed approach combining GS and FS performs the best.

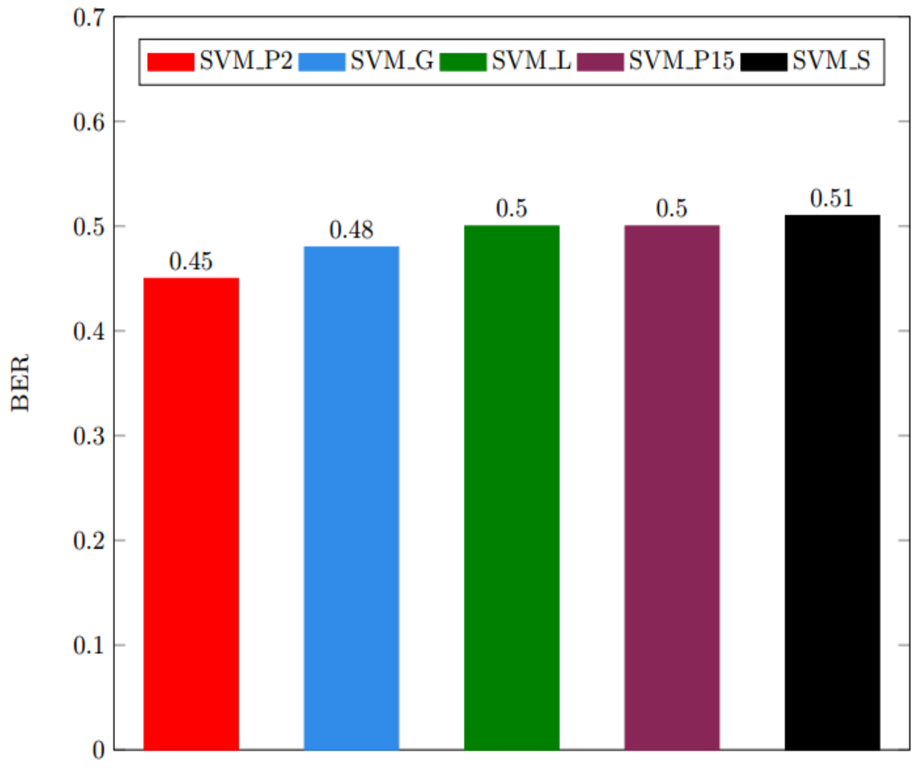


Figure 3.2 Performance evaluation of different SVM kernel functions.

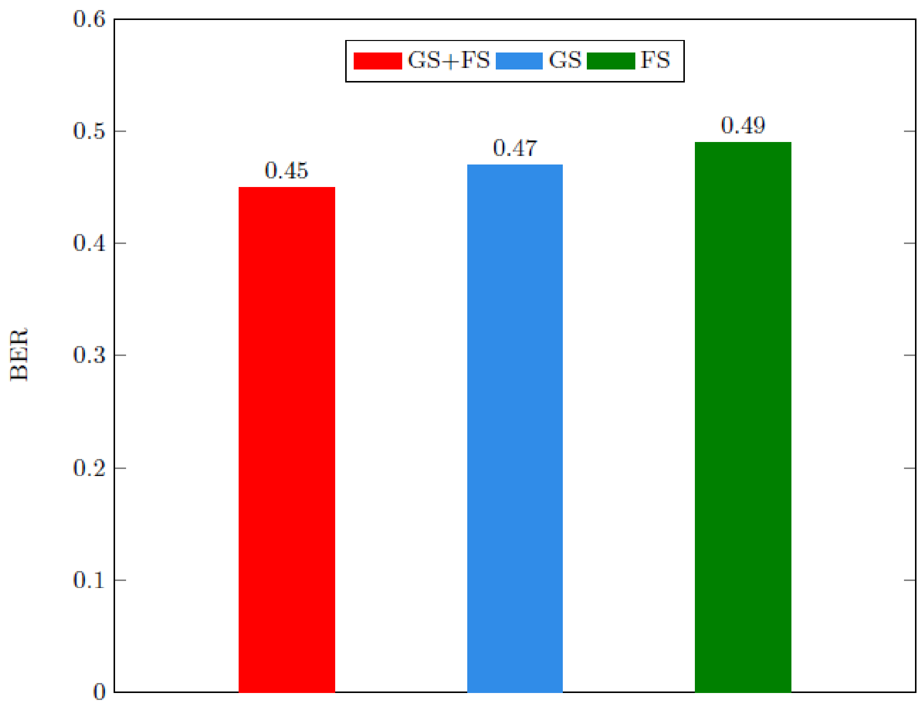


Figure 3.3 Effectiveness of the components of the proposed learning framework.

3.4 Summary

Given gene regulatory networks constructed by unsupervised network inference methods, the goal is to predict and clean the links in the networks. To achieve this goal, a learning framework is proposed, which employs (i) a graph sparsification technique (GeneProbe) for generating inference subgraphs from a given network, and (ii) principal component analysis (PCA) for selecting significant features from high-dimensional feature vectors. The selected feature values are then used to train three classifiers including neural networks (NN), random forests (RF) and support vector machines (SVM) for performing link prediction and link cleaning in the given network.

In this case study, the proposed framework is able to learn better quality training data from noisy networks constructed by a widely used network inference tool (Inferelator). Among the three classification algorithms studied in the paper, SVM with the polynomial kernel of degree 2 outperforms NN and RF in terms of improving the accuracy of Inferelator. This kernel is the best among all SVM kernel functions tested here. Experimental results also show that combining both graph sparsification and PCA is better than using PCA or graph sparsification alone.

To the best of found knowledge, this is the first study to predict and clean the links in gene regulatory networks constructed by unsupervised network inference methods.

CHAPTER 4

INFERRING GENE REGULATORY NETWORKS BY COMBINING SUPERVISED AND UNSUPERVISED METHODS

4.1 Introduction

Supervised methods for inferring gene regulatory networks (GRNs) perform well with good training data. However, when training data is absent, these methods are not applicable. Unsupervised methods do not need training data but their accuracy is low. In this chapter, supervised and unsupervised methods are combined to infer GRNs using time-series gene expression data. Specifically, results obtained from unsupervised methods are used to train supervised methods. Since the results contain noise, a data cleaning algorithm is developed to remove noise, hence improving the quality of the training data. These refined training data are then used to guide classifiers including support vector machines and deep learning tools to infer GRNs through link prediction. Experimental results on several data sets demonstrate the good performance of the classifiers and the effectiveness of the proposed data cleaning algorithm.

4.2 Related Work

Widely used unsupervised methods for time-series gene expression data include BANJO (Bayesian Network Inference with Java Objects) [18], TimeDelay-ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks) [92], tlCLR (Time-Lagged Context Likelihood of Relatedness) [93, 94], DFG (Dynamic Factor Graphs) [22], Jump3 [95], ScanBMA [26], and Inferelator [27].

BANJO models GRNs as a first-order Markov process; it searches through all possible GRNs, seeking the network with the best score. TimeDelay-ARACNE infers GRNs from time-series data using mutual information from information theory. The tlCLR method also uses mutual information and depends on ordinary differential equations to model time-series data. DFG models experimental noise as a fitted Gaussian and then infers GRNs based on an assumed underlying, idealized gene expression pattern. Jump3 uses a non-parametric procedure based on decision trees to reconstruct GRNs. ScanBMA is a Bayesian inference method that incorporates external information to improve the accuracy of GRN inference. Inferelator uses ordinary differential equations that learn a dynamical model for each gene using time-series data. Recent extensions of Inferelator incorporate prior knowledge into the tool, and are resilient to noisy inputs.

On the other hand, supervised methods use training data along with a classification algorithm such as support vector machines (SVMs) [5, 34, 43, 96]. The training data includes known regulatory relationships between genes, also called links, which are used to guide the classification algorithm to reconstruct GRNs through link prediction. The performance of the supervised methods depends on the quality and the amount of available training data.

4.3 Background and Overview

Central to the proposed approach of combining supervised and unsupervised methods for GRN inference is a linear algebra-based data cleaning algorithm. The input of the data cleaning algorithm contains a portion of a weighted directed graph $G = (V, E)$ that represents the topological structure of a GRN. This GRN is inferred by an unsupervised method based on a time-series gene expression dataset. E is the set of directed edges or links, and V is the set of vertices or nodes in G, where each link represents a regulatory relationship and each node represents a gene. Each edge $e = (u,v) \in E$ is associated with a weight, denoted by W(e), where $0 < W(\mathrm{e}) \leq 1.$ As a case study, Inferelator [27] is used as the unsupervised method in this paper. Inferelator is one of the most widely used unsupervised methods in the field.

The main drawback of employing an unsupervised method such as Inferelator for GRN inference is that the method often creates missing and spurious links [33]. Let G be a network constructed by Inferelator. A missing link or edge e_m refers to a regulatory relationship that exists in the ground truth but is not created by Inferelator, and hence $e_m \notin G.$ A spurious link e_s refers to a regulatory relationship that does not exist in the ground truth, but is created by Inferelator, and hence $e_s \in G.$ These missing and spurious links will be used to train supervised methods. The goal is to develop a data cleaning algorithm for removing the errors or noises in the links to get better quality training data. Since an inferred network is sizable, m links are selected with the largest weights and m links with the smallest weights to form an original training set. Then, feature vectors are constructed for the selected 2m links in the training set.

The proposed data cleaning algorithm consists of three steps. First, a distance matrix is calculated for the feature vectors using Laplacian kernel function. Second, a linear algebra technique is adopted to project the training set onto the eigenvectors of the distance matrix to obtain noise-removed features. Third, important features are selected from the noise-removed features. The feature vectors containing the selected important features form a cleaned training set.

Classifiers are built using the cleaned training set and apply these classifiers to predicting links in a regulatory network. The classification algorithms considered here include support vector machines and variants of deep neural networks. Support vector machines are commonly used in bioinformatics [97] while deep neural networks have recently received increasing attention for deep learning. This case study shows how the proposed data cleaning algorithm improves the quality of training examples used to guide the classifiers for inferring GRNs through link prediction.

4.4 Methodology

4.4.1 Feature Vector Construction

A sample of links is selected from the weighted network constructed by Inferelator. The sample contains *m* links (positive training examples) with the largest weights and *m* links (negative training examples) with the smallest weights in the constructed network. These 2*m* links form the original training set. (In the study presented here, *m* = 100.) For each ordered pair of genes u, v in a selected link, a feature vector x is created by concatenating the gene expression profiles of u and v as done in [4, 5]. That is,

$$\mathrm{x} = [u^1, u^2, ..., u^p, v^1, v^2, ..., v^p] \tag{4.1}$$

where $u^1, u^2, ..., u^p$ are the gene expression values of u, and $v^1, v^2, ..., v^p$ are the gene expression values of v. Each gene expression value is a feature.

4.4.2 Data Cleaning

To facilitate the discussion of the proposed data cleaning algorithm, the mathematical symbols and notation used here are first summarized. Matrices (vectors, respectively) are denoted by uppercase (lowercase, respectively) letters. The notation x_i denotes the *i*th vector in matrix X. Elements of a vector are denoted by italic lowercase with a superscript, e.g., x^i is the *i*th element of vector x; scalars are denoted by italic lowercase.

A feature matrix $\mathrm{X} \in \mathbb{R}^{2m \times 2p}$ is constructed in which each row corresponds to a link (feature vector) in the original training set. The proposed data cleaning algorithm consists of three steps.

Step 1: Compute the distance matrix D in Equation (4.2):

$$\mathrm{D} = [d(\mathrm{x}_i, \mathrm{x}_j)]_{ij} \in \mathbb{R}^{2m \times 2m} \tag{4.2}$$

where

$$d(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\sigma \| \mathrm{x}_i - \mathrm{x}_j \|\right), \forall i, j = 1, ..., 2m. \tag{4.3}$$

Here x_i (x_j, respectively) is the *i*th (*j*th, respectively) feature vector of the feature matrix X, $\| \mathrm{x}_i - \mathrm{x}_j \|$ is the Euclidean distance between x_i and x_j, and σ is a user-determined parameter. (In the study presented here, $\sigma = 1$.) The element of the distance matrix in the left hand side of Equation (4.3) corresponds to the element of the kernel K calculated using the Laplacian kernel function [58, 98] in Equation (4.4) below:

$$\mathrm{K} = [k(\mathrm{x}_i, \mathrm{x}_j)]_{ij} \in \mathbb{R}^{2m \times 2m} \tag{4.4}$$

where

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\sigma \| \mathrm{x}_i - \mathrm{x}_j \|\right). \tag{4.5}$$

Step 2: Denote by $\lambda_1 < \lambda_2 < \ldots < \lambda_{2m}$ the eigenvalues of the distance matrix D, and $v_1, v_2, \ldots, v_{2m}$ the corresponding eigenvectors. According to Courant-Fischer Theorem [99], one has

$$v_1 = \arg \min_{z:\|z\|_2=1} z^{\mathrm{T}} D z \tag{4.6}$$

and

$$v_l = \arg \min_{z:\|z\|_2=1, z \perp \mathrm{span}\{v_1, v_2, \ldots, v_{l-1}\}} z^{\mathrm{T}} D z. \tag{4.7}$$

Here $\| z \|_2$ is the 2-norm of the eigenvector z, i.e.

$$\| z \|_2 = \left(\sum_{i=1}^{2m} | z^i |^2 \right)^{\frac{1}{2}}. \tag{4.8}$$

The notation $\mathrm{span}\{v_1, v_2, \ldots, v_{l-1}\}$ is the span of the set of orthonormal eigenvectors, $z \perp \mathrm{span}\{v_1, v_2, \ldots, v_{l-1}\}$ represents that the orthonormal eigenvector z is perpendicular to the span of the set of orthonormal eigenvectors, $\mathrm{span}\{v_1, v_2, \ldots, v_{l-1}\}$ [100, 101].

Step 3: Let $V_t \in \mathbb{R}^{2m \times t}, 1 \le t \le 2m,$ be the matrix whose columns are the first t eigenvectors of the distance matrix D with the smallest eigenvalues. (In the study presented here, $t = 1$.) X is projected onto V_t to obtain $C \in \mathbb{R}^{2m \times 2p}$ with noise-removed features. That is,

$$C = V_t V_t^{\mathrm{T}} X. \tag{4.9}$$

Finally, $M \in \mathbb{R}^{2p}$ are calculated using Equation (4.10):

$$M = 1^{\mathrm{T}} C \tag{4.10}$$

where $\mathbb{1}$ is a 2m-dimensional column vector of all ones. The bottom k elements in M corresponding to the k minimum values in M are selected and their positions are stored in $\mathrm{P} \in \mathbb{R}^k$. (In the study presented here, $k = 10$.) Construct feature vectors by selecting only k features, based on the positions in P, from the feature vectors in C, and store the feature vectors of k features in the transformed (cleaned) training set, $\overline{\mathrm{C}}$.

4.4.3 Link Prediction

The cleaned training set $\overline{\mathrm{C}}$ (with feature vectors of k features obtained from step 3 above) is used to train classification algorithms including support vector machines [58, 98], deep neural networks with weights initialized by deep belief networks, and deep belief networks with weights initialized by stacked AutoEncoder [102]. Given a testing set with n ordered gene pairs whose labels are unknown, the goal here is to predict the label of each gene pair (u, v) in the testing set using a trained classification model. That is, the classification model will predict whether there is a link from gene u to gene v. The predicted label is +1 if it is predicted that there is a link from u to v, and −1 otherwise.

To perform the link prediction, a feature vector is constructed for each gene pair (u, v) in the testing set by concatenating the gene expression values of u and v as shown in Equation (4.1). A feature matrix S is created for the testing set, selecting k features based on the positions in P, and store the feature vectors of k features of the testing set in $\overline{\mathrm{S}}$. The labels of the testing examples are then predicted by a trained classification model.

4.5 Experiments and Results

4.5.1 Datasets

A series of experiments are carried out to evaluate the performance of the proposed approach, using three time-series gene expression datasets available in the DREAM4 100-gene *in silico* network inference challenge [20, 21]. Each dataset contains 10 times series, where each time series has 21 time points, for 100 genes. Each gene has $(10 \times 21) = 210$ gene expression values. Each link consists of two genes, and hence is represented by a 420-dimensional feature vector.

Each time-series dataset is associated with a gold standard file, where the gold standard represents the ground truth of the network structure for the time-series data. Each link in the gold standard represents a true regulatory relationship between two genes. For a given time-series dataset, Inferelator [27] outputs a list of ordered gene pairs where each gene pair is associated with a positive, non-zero weight. Gene pairs not shown in the output list are assumed to have a weight of -1. m gene pairs (positive training examples) with the largest weights and m gene pairs (negative training examples) with the smallest weights are selected in the output list of Inferelator. These $2m$ links formed the original training set. The proposed data cleaning algorithm is then used to clean the $2m$ links to obtain a cleaned training set.

Table 4.1 presents details of the data used in the experiments. The table shows the numbers of true present and true missing links in each gold standard network, as well as the numbers of gene pairs in the output list of Inferelator and the numbers of gene pairs not shown in the output list of Inferelator for each time-series dataset. Each network contains 100 nodes or genes, which form 9,900 ordered gene pairs totally.

Table 4.1 Data Used in the Experiments

	Net1	Net2	Net3
Directed	Yes	Yes	Yes
Nodes	100	100	100
True present links	176	249	195
True missing links	9,724	9,651	9,705
Gene pairs in output	6,232	6,066	6,186
Gene pairs not in output	3,668	3,834	3,714

For each network, three sets of testing data are generated. Each testing set contains 50 randomly selected links from the gold standard. Among the 50 links, 25 are true present links and 25 are true missing links in the gold standard. The label (+1 or −1) of each selected link is known, where +1 represents a true present link and −1 represents a true missing link. These testing data were excluded from the training sets, so the testing sets and training sets were disjoint. There were 9 testing sets totally.

4.5.2 Experimental Setup

Three classification algorithms are considered, namely support vector machines (SVM), deep neural networks with weights initialized by deep belief networks (DNN_DBN), and deep neural networks with weights initialized by stacked AutoEncoder (DNN_Auto). Software used in this work included: the deepnet package in R [103], the SVM program with the linear kernel in the LIBSVM package [104] and other kernel functions in the kernlab package [105]. In addition, R is used to write some utility tools for performing the experiments.

The performance of each classification algorithm was evaluated as follows. As in [104], each classification program is trained where the option of probability estimation in each program was turned on. Given a testing link x, the program calculates the probability of x being in the positive class C_+, i.e., $P(C_+|x)$. Each gene pair in a testing test is predicted to have the label +1 (i.e., predicted as a present link) if its probability is greater than or equal to the median of the probability estimates produced by the program. The gene pair is predicted to have the label −1 (i.e., predicted as a missing link) if its probability is less than the median probability. In evaluating the classification algorithms, a true positive is defined to be a true present link that is predicted as a present link. A false positive is a true missing link that is predicted as a present link. A true negative is a true missing link that is predicted as a missing link. A false negative is a true present link that is predicted as a missing link.

In evaluating Inferelator, a gene pair is considered as an inferred present link if its weight is greater than or equal to the median of the weighs produced by Inferelator. The gene pair is an inferred missing link if its weight is less than the median weight. A true positive is defined to be a true present link that is an inferred present link. A false positive is a true missing link that is an inferred present link. A true negative is a true missing link that is an inferred missing link. A false negative is a true present link that is an inferred missing link.

Let TP (FP, TN, FN, respectively) denote the number of true positives (false positives, true negatives, false negatives, respectively) for a testing set. The performance measure used in the study is the Area Under the ROC Curve (AUC) [97], defined as

$$\mathrm{AUC} = \frac{1}{2}\left(\frac{\mathrm{TP}}{\mathrm{TP+FN}} + \frac{\mathrm{TN}}{\mathrm{TN+FP}}\right). \tag{4.11}$$

Each classification algorithm is applied to each testing set and the AUC the algorithm obtains is recorded for the testing set. The larger AUC a classification algorithm has, the better performance that algorithm achieves. MAUC is used to denote the mean of the AUC values averaged over the three testing sets generated from a network, and use AMAUC to denote the average of the MAUC values over the three networks used in the experiments.

4.5.3 Experimental Results

Experiments are conducted first to evaluate the performance of SVM with different kernel functions, including the linear kernel (SVM_L), Gaussian kernel (SVM_G), sigmoid kernel (SVM_S), and polynomial kernel of degree 2 (SVM_P). Figure 4.1 shows the AMAUC values of SVM with the different kernels. It can be seen from Figure 4.1 that SVM with the linear kernel (SVM_L) performs the best. The non-linear kernels including SVM_G, SVM_S and SVM_P yield smaller AMAUC values, and hence perform worse, than SVM_L.

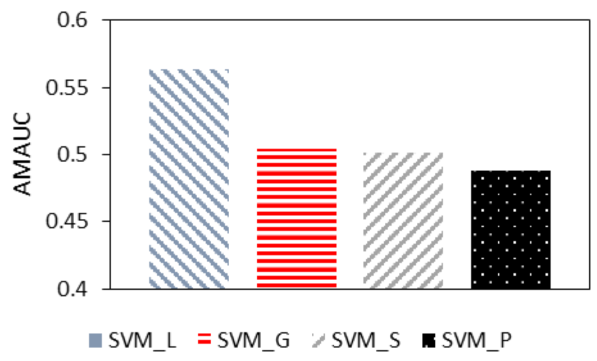


Figure 4.1 Comparison of the AMAUC values of SVM with four different kernels including the linear kernel (SVM_L), Gaussian kernel (SVM_G), sigmoid kernel (SVM_S), and polynomial kernel of degree 2 (SVM_P).

In subsequent experiments, the SVM kernel is fixed at the linear kernel. Table 4.2 lists the MAUC values of SVM_L, deep neural networks with weights initialized by deep belief networks (DNN_DBN), deep neural networks with weights initialized by stacked

AutoEncoder (DNN_Auto), and Inferelator (Inf). For each network, the classification algorithm with the best performance, i.e., the largest MAUC, is highlighted in boldface.

Table 4.2 MAUC Values of Three Classifiers and Inferelator

Dataset	SVM_L	DNN_DBN	DNN_Auto	Inf
Net1	**0.726**	0.626	0.546	0.380
Net2	0.460	0.440	**0.480**	0.353
Net3	0.506	**0.546**	0.506	0.380
Average	**0.564**	0.537	0.511	0.371

It can be seen from Table 4.2 that SVM_L is the best classifier for Net1 and yields the largest average MAUC (i.e., AMAUC) of 0.564. DNN_DBN is the best classifier for Net3 and yields the second largest average MAUC of 0.537. DNN_Auto is the best classifier for Net2 with the average MAUC of 0.511. All the three classifiers (i.e., supervised methods) perform better than the unsupervised method, Inferelator, whose average MAUC is 0.371.

It is worth noting that the kernel-based program, SVM_L, performs better than the deep learning programs DNN_DBN and DNN_Auto. Deep learning is a powerful tool for image classification on big data with hundreds of classes. The deep learning programs model high-level abstractions in data through multiple non-linear transformations. In contrast, this work focuses on binary classification with relatively small datasets in which the learned linear relationship between feature vectors and labels was shown to be effective in testing data classification. As a consequence, the deep learning programs perform worse than the kernel-based program.

Figure 4.2 shows the AMAUC values of the three classifiers, SVM_L, DNN_DBN and DNN_Auto, into which the proposed data cleaning algorithm was not incorporated. Thus, the classifiers were trained by uncleaned data. Comparing Figure 4.2 with Table 4.2 where data was cleaned, one can see that the AMAUC values in Figure 4.2 are smaller than those in Table 4.2. The performance of the classifiers degrades when running on uncleaned data, showing the effectiveness of the proposed data cleaning algorithm. Notably, SVM_L suffers the most when data is not cleaned.

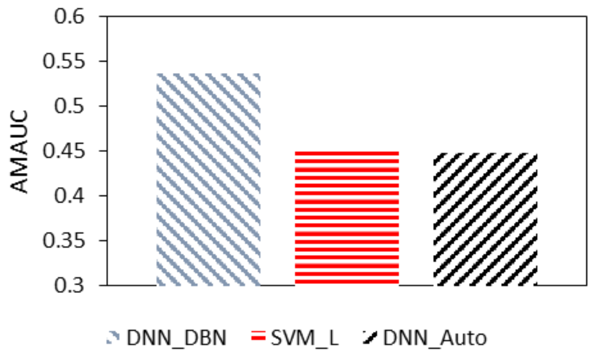


Figure 4.2 Comparison of the AMAUC values of three classifiers SVM_L, DNN_DBN and DNN_Auto where the AMAUC values were obtained by running the classifiers on uncleaned data.

One component of the data cleaning algorithm is feature selection where Equation (4.10) is used to select *k* most important features to form feature vectors of *k* features. Figure 4.3 shows the AMAUC values of the three classifiers, SVM_L, DNN_DBN and DNN_Auto, for varying *k* values. It can be seen from Figure 4.3 that SVM_L continues to be the best classifier when *k* values change. Its behavior is stable with respect to *k*. Selecting *k* features makes the proposed approach computationally efficient for large datasets with good predictive performance.

We also tested on different values for the parameters *m* in Equation (4.2), σ in Equation (4.3) and *t* in Equation (4.9) used in the proposed data cleaning algorithm. The results obtained were similar to those of using the default values for these parameters (*m* = 100, $\sigma = 1$, $t = 1$), and the qualitative conclusion remains the same.

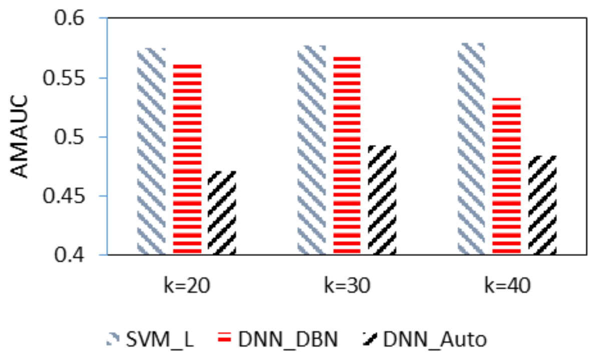


Figure 4.3 Impact of the number of selected features, k, on the performance of three classifiers SVM_L, DNN_DBN and DNN_Auto.

4.6 Summary

Machine learning in biomedicine has received increasing attention recently [16, 106, 107]. In this paper a hybrid approach is presented for learning gene regulatory networks (GRNs) by combining supervised and unsupervised methods. Central to the proposed approach is a linear algebra-based algorithm for cleaning the results of unsupervised methods. The cleaned results are then used to train supervised methods to perform GRN inference through link prediction. In this case study, a widely used unsupervised method is adopted, Inferelator, as well as three popular classifiers including support vector machines (SVM), deep neural networks with weights initialized by deep belief networks (DNN_DBN) and deep neural networks with weights initialized by stacked AutoEncoder (DNN_Auto). The experimental results show the superiority of the proposed hybrid approach over the unsupervised method. Among the three classifiers, SVM with the linear kernel outperforms the two variants of deep neural networks, DNN_DBN and

DNN_Auto. This linear kernel is the best among all SVM kernel functions tested here. The experimental results also show the effectiveness of the proposed data cleaning algorithm.

This data cleaning algorithm is related to the noise-filtering algorithm developed by Ouyang *et al*. [108]. While both algorithms aim to improve the quality of network data, they differ in two major ways. First, Ouyang *et al*.'*s* method is designed for undirected networks and employs the Laplacian matrix, which is symmetric for undirected networks. The eigenvalues of the symmetric matrix are real numbers, and the corresponding eigenvectors are orthonormal. In contrast, the networks considered here are directed networks. When applying Ouyang *et al*.'*s* method to directed networks, one would get a non-symmetric Laplacian matrix, whose eigenvalues may contain complex numbers. In such a situation, an orthonormal set of eigenvectors cannot be found, nor even any pair of eigenvectors that are orthogonal (except perhaps by rare chance) [109]. Thus, instead of using the Laplacian matrix, a distance matrix D is introduced as shown in Equation (4.2), which can be calculated by using the Laplacian kernel function as shown in Equation (4.5). The eigenvalues of this symmetric positive semidefinite distance matrix D are real, non-negative numbers, and the corresponding eigenvectors are orthonormal. Second, Ouyang *et al*.'*s* method does not include feature selection. In contrast, Equation (4.10) is used to select *k* most important features and use the selected features for link prediction. It should also be pointed out that the work of Ouyang *et al*. did not consider machine learning algorithms. In contrast, data cleaning algorithm is used here to get better quality training data, which are then used to guide machine learning tools to perform link prediction.

To the best of found knowledge, the proposed hybrid approach is the first work to combine supervised methods with an unsupervised method (Inferelator) for GRN inference.

CHAPTER 5

REVERSE ENGINEERING GENE REGULATORY NETWORKS USING SAMPLING AND BOOSTING TECHNIQUES

5.1 Introduction

Reverse engineering gene regulatory networks (GRNs), also known as network inference, refers to the process of reconstructing GRNs from gene expression data. Biologists model a GRN as a directed graph in which nodes represent genes and links show regulatory relationships between the genes. By predicting the links to infer a GRN, biologists can gain a better understanding of regulatory circuits and functional elements in cells. Existing supervised GRN inference methods work by building a feature-based classifier from gene expression data and using the classifier to predict the links in GRNs. Observing that GRNs are sparse graphs with few links between nodes, this chapter presents a new approach to supervised GRN inference. The imbalanced classification problem is tackled by using sampling techniques, including under-sampling and over-sampling, to obtain a balanced training set. This balanced training set, containing the same number of positive and negative training examples, is used to train a machine learning algorithm to make predictions. Furthermore, several boosting techniques are developed to enhance the prediction performance. As the experimental results show later, this new approach outperforms the existing supervised GRN inference methods [8, 43].

5.2 Methods

5.2.1 Problem Statement

n genes are given where each gene has p expression values. The gene expression profile of these n genes is denoted by $G \subseteq R^{n \times p}$, which contains n rows, each row corresponding to a gene, and p columns, each column corresponding to an expression value [8, 16, 17, 43, 110]. In addition, known regulatory relationships or links among some genes are given. Suppose these known regulatory relationships are stored in a matrix $X \subseteq R^{m \times 3}$, which forms the training dataset. X contains m rows, where each row shows a known regulatory relationship between two genes, and three columns. The first column shows a transcription factor (TF). The second column shows a target gene. The third column shows the label, which is $+1$ if the TF is known to regulate the expression of the target gene or -1 if the TF is known not to regulate the expression of the target gene. The matrix X represents a partially observed or known gene regulatory network for the n genes. If the label of a row in X is $+1$, then the TF in that row regulates the expression of the target gene in that row, and hence that row represents a directed link or edge of the network. That row is a positive training example. If the label of a row in X is -1, then there is no link between the corresponding TF and target gene in that row. That row is a negative training example. The positive and negative training examples in X are used to train a machine learning or classification algorithm. There are much more negative training examples than positive training examples in X.

The test dataset contains ordered pairs of genes (g_1, g_2) where the regulatory relationship between g_1 and g_2 is unknown. Given a test example, i.e., an ordered pair of genes (g_1, g_2) in the test dataset, the goal of *link prediction* is to use the trained classifier to predict the label of the test example. The predicted label is either $+1$ (i.e., a directed link is predicted to be present from g_1 to g_2) or -1 (i.e., a directed link is predicted to be absent from g_1 to g_2). Here, the present link means g_1 (a transcription factor) regulates the expression of g_2 (a target gene) whereas the absent link means g_1 does not regulate the expression of g_2.

5.2.2 Feature Vector Construction

To perform training and prediction, a feature matrix $D \subseteq R^{q \times 2p}$ with q feature vectors based on the gene expression profile G is constructed. Let g_1 and g_2 be two genes. Let $g_1^1, g_1^2, ..., g_1^p$ be the gene expression values of g_1 and $g_2^1, g_2^2, ..., g_2^p$ be the gene expression values of g_2. The feature vector of the ordered pair of genes (g_1, g_2), denoted D_d, is stored in the feature matrix D and constructed by concatenating their gene expression values as follows:

$$D_d = (g_1^1, g_1^2, ..., g_1^p, g_2^1, g_2^2, ..., g_2^p) \tag{5.1}$$

Thus, the ordered pair of genes (g_1, g_2) corresponds to a point in 2*p*-dimensional space. Each training and test example is represented by a 2*p*-dimensional feature vector. For a positive training example, the label of its feature vector is $+1$. For a negative training example, the label of its feature vector is -1. For a test example, the label of its feature vector is unknown and to be predicted. This feature vector construction method has been widely used by existing supervised GRN inference methods [8, 16, 17, 43, 110].

5.2.3 Under-sampling

Given is a training dataset X that is the union of two disjoint subsets X_+ and X_-. X_+ is the minority class, containing positive training examples (i.e., known present links). X_- is the majority class, containing negative training examples (i.e., known absent links). X_+ is much smaller than X_-. The under-sampling method works as follows [111-113]. It samples a random subset $X_-^s \subseteq X_-$ such that the size of X_-^s is equal to the size of X_+ (i.e., $|X_-^s| = |X_+|$). Thus, $X^s = X_+ \cup X_-^s$ forms a balanced dataset. Then X^s is used to train a machine learning algorithm. The trained model will be used to predict the labels of test examples.

5.2.4 Over-sampling

The over-sampling method is based on SMOTE [111, 114, 115]. Given is the training dataset $X = X_+ \cup X_-$ as described above. The proposed over-sampling method creates a new dataset X_{++} that contains all examples in X_+ and many synthetic examples generated as follows. For each example $x_i \in X_+$, h-nearest neighbors of x_i are selected, where

$$h = \left\lfloor \frac{1.1 \times |X_-|}{|X_+|} \right\rfloor \tag{5.2}$$

Here, Euclidean distances are calculated to find the h-nearest neighbors. Denote these h-nearest neighbors as x_r, $1 \le r \le h$.

A new synthetic example x_{new} along the line between x_i and x_r, $1 \le r \le h$, is created as follows:

$$x_{new} = x_i + (x_r - x_i) \times \delta \tag{5.3}$$

where $\delta \in (0,1)$ is a random number. x_{new} is added to X_{++}. Generating and adding such synthetic examples to X_{++} is continued until X_{++} is larger than X_{-}. Then a random subset $X_{++}^{s} \subseteq X_{++}$ is selected such that the size of X_{++}^{s} is equal to the size of X_{-} (i.e., $|X_{++}^{s}|$ = $|X_{-}|$). Thus, X^{s} = $X_{++}^{s} \cup X_{-}$ forms a balanced dataset. X^{s} is then used to train a machine learning algorithm. The trained model will be used to predict the labels of test examples.

5.2.5 Boosting

The performance of the proposed link prediction algorithms is further improved through boosting. Boosting algorithms such as AdaBoost [116-119], described below, have been used in various domains with great success. A weighted decision tree [120] is used as the base learning algorithm and a strong classifier is created through an iterative procedure as follows. Let X be the set of training examples $\{x_1, x_2, ..., x_m\}$. The label associated with example x_i is y_i such that

$$y_i = \begin{cases} \text{+1 if } x_i \text{ is a positive example (i.e., present link)} \\ \text{-1 if } x_i \text{ is a negative example (i.e., missing link)} \end{cases} \tag{5.4}$$

Initially, in iteration 1, each example is assigned an equal weight, i.e., $W_1(x_i) = \dfrac{1}{m}$, $1 \le i \le m$. In iteration k, $1 \le k \le K$, AdaBoost generates a base learner (i.e., model) H_k

by calling the base learning algorithm on the training set X with weights W_k. Then H_k is used to classify each training example x_i as either $+1$ (i.e., x_i is a predicted present link) or -1 (i.e., x_i is a predicted absent link). That is,

$$H_k(x_i) = \begin{cases} +1 \text{ if } H_k \text{ classifies } x_i \text{ as a positive example (i.e., present link)} \\ -1 \text{ if } H_k \text{ classifies } x_i \text{ as a negative example (i.e., absent link)} \end{cases} \tag{5.5}$$

Let $E_k = \{x_i \mid H_k(x_i) \neq y_i\}$. The error ε_k of H_k is:

$$\varepsilon_k = \sum_{x_i \in E_k} W_k(x_i) \tag{5.6}$$

The weight α_k of H_k is:

$$\alpha_k = \frac{1}{2} \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right) \tag{5.7}$$

AdaBoost then updates the weight of each training example x_i, $1 \leq i \leq m$, as follows:

$$\begin{aligned} W_{k+1}(x_i) &= \begin{cases} \dfrac{W_k(x_i)}{Z_k} \times e^{-\alpha_k} \text{ if } H_k(x_i) = y_i \\ \dfrac{W_k(x_i)}{Z_k} \times e^{\alpha_k} \text{ if } H_k(x_i) \neq y_i \end{cases} \\ &= \frac{W_k(x_i)\exp(-\alpha_k y_i H_k(x_i))}{Z_k} \end{aligned} \tag{5.8}$$

where Z_k is a normalization factor chosen so that W_{k+1} is normally distributed. The weights of incorrectly classified examples will increase in iteration $k+1$. Then, in iteration $k+1$, AdaBoost generates a base learner H_{k+1} by calling the base learning algorithm again on the training set X with weights W_{k+1}. Such a process is repeated K times. Using this technique, each weak classifier H_{k+1} should have greater accuracy than its predecessor H_k. The final, strong classifier H is derived by combining the votes of the weighted weak classifiers H_k, $1 \le k \le K$, where the weight α_k of a weak classifier H_k is calculated as shown in Equation (5.7).

Specifically, given an unlabeled test example $\hat{x}$, $H(\hat{x})$ is calculated as follows:

$$H(\hat{x}) = sign(\sum_{k=1}^{K} \alpha_k H_k(\hat{x})) \tag{5.9}$$

The *sign* function indicates that if the sum of the results of the weighted K weak classifiers is greater than or equal to zero, then H classifies $\hat{x}$ as $+1$ (i.e., $\hat{x}$ is a predicted present link); otherwise H classifies $\hat{x}$ as -1 (i.e., $\hat{x}$ is a predicted absent link).

Extensions of AdaBoost are proposed by modifying Equation (5.8) to obtain the following variants:

Boost I:

$$\begin{aligned} W_{k+1}(x_i) &= \begin{cases} \dfrac{e^{-\alpha_k}}{Z_k} \text{ if } H_k(x_i) = y_i \\ \dfrac{e^{\alpha_k}}{Z_k} \text{ if } H_k(x_i) \ne y_i \end{cases} \\ &= \frac{\exp(-\alpha_k y_i H_k(x_i))}{Z_k} \end{aligned} \tag{5.10}$$

Boost II:

$$W_{k+1}(x_i) = \frac{\exp(-\sum_{j=1}^{k} \alpha_j H_j(x_i) y_i)}{Z_k} \tag{5.11}$$

Boost III:

$$W_{k+1}(x_i) = \begin{cases} \dfrac{C \times e^{\alpha_k}}{Z_k} & \text{if } H_k(x_i) = -1 \text{ and } y_i = +1 \\ \dfrac{e^{-\alpha_k}}{Z_k} & \text{if } H_k(x_i) = +1 \text{ and } y_i = +1 \\ \dfrac{e^{\alpha_k}}{Z_k} & \text{if } H_k(x_i) = +1 \text{ and } y_i = -1 \\ \dfrac{e^{-\alpha_k}}{Z_k} & \text{if } H_k(x_i) = -1 \text{ and } y_i = -1 \end{cases} \tag{5.12}$$

where C is the number of examples in the majority class (i.e., negative class) divided by the number of examples in the minority class (i.e., positive class) in the training set.

Each one of the above variants is taken as a new boosting technique. Boost I is a simplified version of AdaBoost. For each training example x_i, $1 \le i \le m$, Boost I does not consider $W_k(x_i)$ when calculating $W_{k+1}(x_i)$. Boost II is an accumulative version of AdaBoost. It considers all weak classifiers H_j, $1 \le j \le k$, obtained in the previous k iterations when calculating the weight $W_{k+1}(x_i)$. Specifically, Boost II will increase the weight of a training example x_i in iteration $k+1$ if the majority of the weak classifiers obtained in the previous k iterations incorrectly classify x_i. Boost III can be regarded as a cost-sensitive boosting technique. For an imbalanced dataset, positive examples (i.e., those with labels of $+1$) are much fewer than negative examples (i.e., those with labels of -1). The objective here is to improve the classification performance on the minority (i.e., positive) class. Hence the cost C is introduced, giving more weights to misclassified examples in the minority class where the examples are classified as negative though they have labels of $+1$. For a training example x_i that is correctly classified in iteration k, its weight is decreased in iteration $k+1$ so that the next classifier H_{k+1} pays less attention to x_i while focusing more on the other examples that are incorrectly classified in iteration k.

5.2.6 The Proposed Approach

Figure 5.1 presents an overview of the proposed approach. In (A), a training set is given containing imbalanced labeled links. These labeled links include few positive examples (i.e., known present links with labels of $+1$) and a lot of negative examples (i.e., known absent links with labels of -1). In addition, a test set is given in which each test example is an unlabeled ordered gene pair. Feature vectors are constructed for both training examples and test examples as described in Section 8.2.2. In (B), a sampling technique is applied, either under-sampling as described in Section 8.2.3 or over-sampling as described in Section 8.2.4, to the training set to obtain a balanced training set. In (C), a boosting technique is applied as described in Section 8.2.5 to the balanced training set to learn K models (weak classifiers). These models predict the labels of the test examples. In (D), the weighted majority vote is taken from the weak classifiers as shown in Equation (5.9) to make final predictions of the labels of the test examples. A test example is a predicted present link if its predicted label is $+1$; a test example is a predicted absent link if its predicted label is -1.

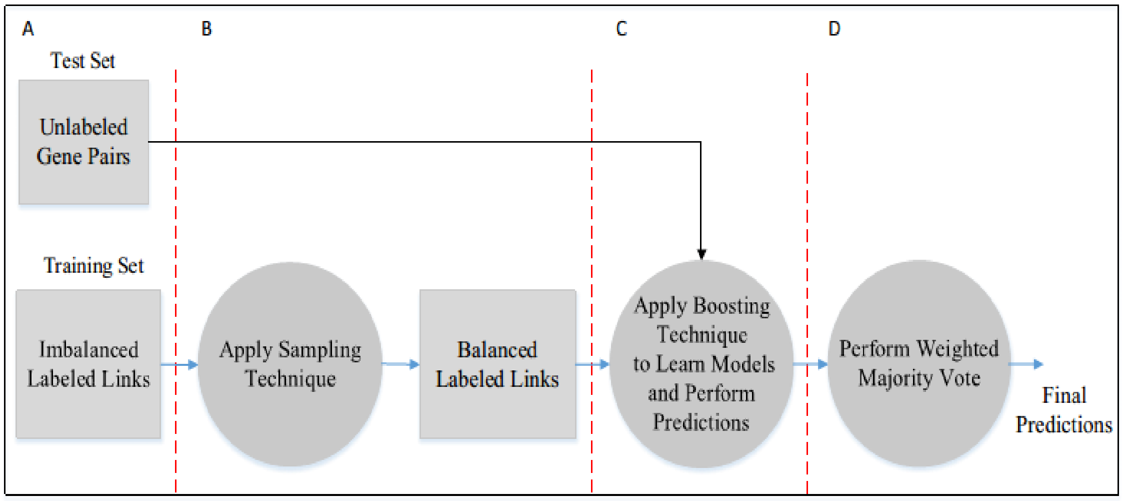


Figure 5.1 The proposed approach for link prediction in gene regulatory networks.

5.3 Experiments and Results

5.3.1 Datasets

GeneNetWeaver [77] is used to generate the datasets related to yeast and E. coli. Specifically four different networks are built for each organism where the networks contained 50, 100, 150, 200 genes (or nodes) respectively. Table 5.1 presents details of the yeast networks, showing the number of nodes (edges, respectively) in each network. The present edges or links in a network form positive examples. The absent edges or links in a network form negative examples. Table 5.2 presents details of the E. coli networks.

Table 5.1 Yeast Networks Used in the Experiments

Network	Directed	#Nodes	#Edges	#Positive examples	#Negative examples
Yeast 50	Yes	50	63	63	2387
Yeast 100	Yes	100	281	281	9619
Yeast 150	Yes	150	333	333	22017
Yeast 200	Yes	200	517	517	39283

For each network, three files of gene expression data are generated. These files were labeled as knockouts, knockdowns and multifactorial, respectively.

Table 5.2 E. Coli Networks Used in the Experiments

Network	Directed	#Nodes	#Edges	#Positive examples	#Negative examples
E. coli 50	Yes	50	68	68	2382
E. coli 100	Yes	100	177	177	9723
E. coli 150	Yes	150	270	270	22080
E. coli 200	Yes	200	415	415	39385

A knockout is a technique to deactivate the expression of a gene, which is simulated by setting the transcription rate of this gene to zero [7]. A knockdown is a technique to reduce the expression of a gene, which is simulated by reducing the transcription rate of this gene by half [7]. Multifactorial perturbations are simulated by randomly increasing or decreasing the activation of the genes in a network simultaneously [7]. Totally there were twelve gene expression datasets for yeast and E. coli respectively.

5.3.2 Experimental Methodology

The proposed approach is compared with existing supervised GRN inference methods [7, 8]. The existing methods employ support vector machines (SVM) and use the same feature vector construction method as the proposed approach (cf. Section 8.2.2); however, they lack sampling and boosting techniques. Table 5.3 lists the abbreviations of the fifteen algorithms that have been evaluated and compared in this study where twelve algorithms are boosting-related and three algorithms are SVM-related.

Table 5.3 Abbreviations of the Fifteen Algorithms Studied in This Paper

Abbreviation	Algorithm
AdaBoost	AdaBoost technique
AdaBoost+U	AdaBoost with under-sampling technique
AdaBoost+O	AdaBoost with over-sampling technique
Boost I	Boost I technique
Boost I+U	Boost I with under-sampling technique
Boost I+O	Boost I with over-sampling technique
Boost II	Boost II technique
Boost II+U	Boost II with under-sampling technique
Boost II+O	Boost II with over-sampling technique
Boost III	Boost III technique
Boost III+U	Boost III with under-sampling technique
Boost III+O	Boost III with over-sampling technique
SVM	SVM technique
SVM+U	SVM with under-sampling technique
SVM+O	SVM with over-sampling technique

The performance of each algorithm was evaluated through 10-fold cross validation. The positive examples (negative examples, respectively) were evenly distributed to the ten folds. When testing a fold, the Area Under the ROC Curve (AUC) of an algorithm was calculated where the AUC is defined as

$$\mathrm{AUC} = \frac{1}{2} \times \left(\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} + \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \right) \tag{5.13}$$

Here TP (FP, TN, FN, respectively) denotes the number of true positives (false positives, true negatives, false negatives, respectively) for the test set. A true positive (true negative, respectively) is a predicted present link (a predicted absent link, respectively) that is indeed a known present link (a known absent link, respectively). A false positive (false negative, respectively) is a predicted present link (a predicted absent link, respectively) that is in fact a known absent link (a known present link, respectively). For each algorithm, the mean AUC, denoted MAUC, over the ten folds was computed and recorded. The higher MAUC an algorithm has, the better performance that algorithm achieves.

5.3.3 Experimental Results

Experiments are first conducted to evaluate the performance of SVM with different kernel functions, including the linear kernel, polynomial kernel of degree 2, Gaussian kernel, and sigmoid kernel. It was observed that the Gaussian kernel performed the best. In subsequent experiments, the SVM kernel is fixed at the Gaussian kernel.

Figure 5.2 shows the AMAUC values of the three SVM-related algorithms, namely SVM, SVM+U, SVM+O, on the twelve yeast datasets used in the experiments. For each algorithm, the AMAUC was calculated by taking the average of the MAUC values the algorithm received over the twelve yeast datasets. It can be seen from Figure 5.2 that SVM+U performed better than SVM+O and SVM. Figure 5.3 shows the AMAUC values of the twelve boosting-related algorithms on the twelve yeast datasets used in the experiments. It can be seen from Figure 5.3 that Boost III+U performed the best on the twelve yeast datasets.

Table 5.4 shows the MAUC values of Boost III+U and SVM+U, and compares them with the existing approaches using SVM only [7, 8] on the twelve yeast datasets. For each dataset, the algorithm with the best performance (i.e., the highest MAUC) is shown in bold. It can be seen from Table 5.4 that Boost III+U has the best overall performance on the yeast datasets, and beats the existing approaches using SVM only [7, 8].

Figure 5.4 shows the AMAUC values of the three SVM-related algorithms, namely SVM, SVM+U, SVM+O, on the twelve E. coli datasets used in the experiments. It can be seen that SVM+O outperformed SVM+U and SVM. Figure 5.5 shows the AMAUC values of the twelve boosting-related algorithms on the twelve E. coli datasets used in the experiments. It can be seen from Figure 5.5 that Boost II+U performed the best among the twelve boosting-related algorithms. Table 5.5 shows the MAUC values of Boost II+U and SVM+O, and compares them with the existing approaches using SVM only [7, 8] on the twelve E. coli datasets. It can be seen from Table 5.5 that Boost II+U has the best overall performance on the E. coli datasets, and beats the existing approaches using SVM only [7, 8].

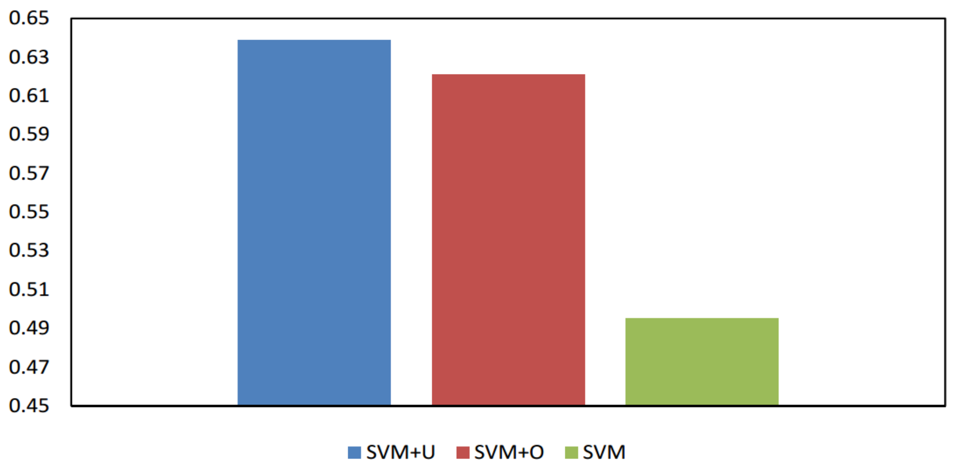


Figure 5.2 AMAUC values of three SVM-related algorithms on twelve yeast datasets.

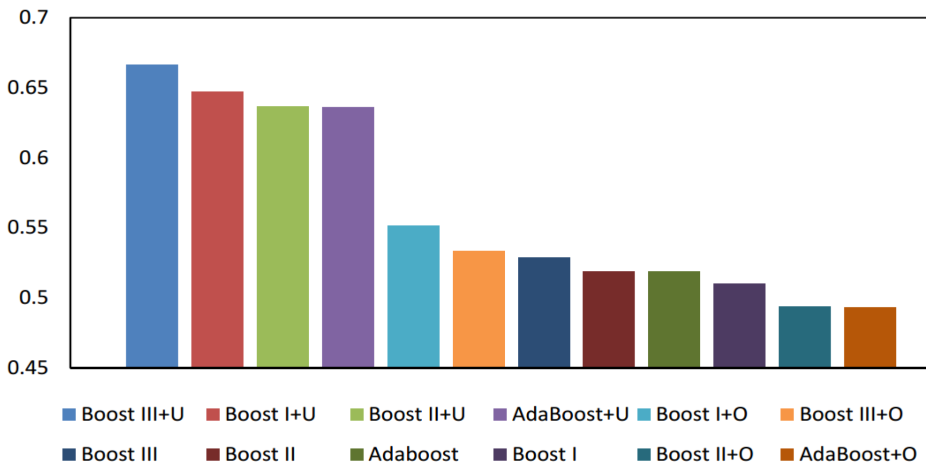


Figure 5.3 AMAUC values of twelve boosting-related algorithms on twelve yeast datasets.

Table 5.4 MAUC Values of Three Algorithms on Twelve Yeast Datasets

Dataset	Boost III+U	SVM+U	SVM
Yeast 50 knockouts	0.709	**0.802**	0.534
Yeast 50 knockdowns	0.664	**0.79**	0.529
Yeast 50 multifactorial	0.679	**0.736**	0.526
Yeast 100 knockouts	**0.729**	0.638	0.487
Yeast 100 knockdowns	**0.706**	0.69	0.489
Yeast 100 multifactorial	**0.701**	0.639	0.472
Yeast 150 knockouts	**0.633**	0.529	0.486
Yeast 150 knockdowns	**0.673**	0.519	0.494
Yeast 150 multifactorial	**0.68**	0.538	0.474
Yeast 200 knockouts	0.599	**0.622**	0.49
Yeast 200 knockdowns	0.612	**0.623**	0.49
Yeast 200 multifactorial	**0.608**	0.54	0.471
AMAUC	**0.666**	0.638	0.495

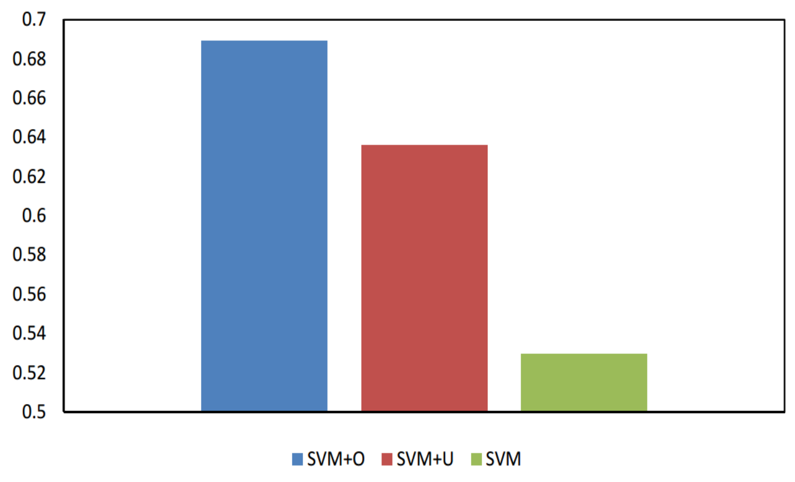


Figure 5.4 AMAUC values of three SVM-related algorithms on twelve E. coli datasets.

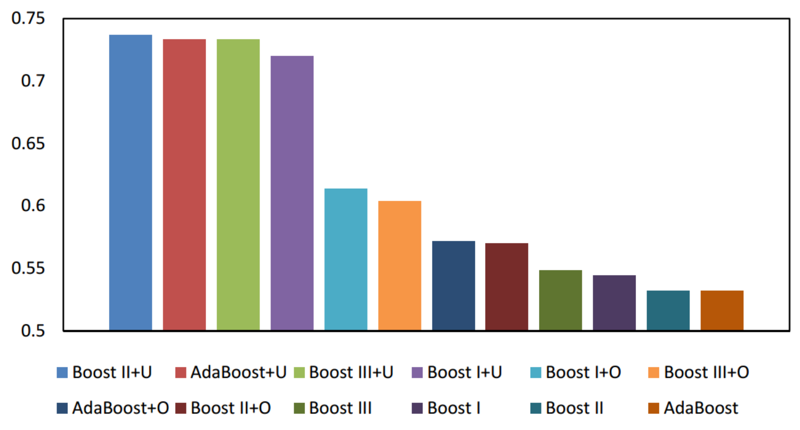


Figure 5.5 AMAUC values of twelve boosting-related algorithms on twelve E. coli datasets.

Table 5.5 MAUC Values of Three Algorithms on Twelve E. Coli Datasets

Dataset	Boost II+U	SVM+O	SVM
E. coli 50 knockouts	**0.872**	0.767	0.669
E. coli 50 knockdowns	**0.866**	0.776	0.505
E. coli 50 multifactorial	**0.879**	0.819	0.706
E. coli 100 knockouts	**0.77**	0.708	0.493
E. coli 100 knockdowns	**0.758**	0.712	0.492
E. coli 100 multifactorial	**0.75**	0.711	0.49
E. coli 150 knockouts	**0.625**	0.567	0.498
E. coli 150 knockdowns	**0.597**	0.596	0.5
E. coli 150 multifactorial	**0.636**	0.584	0.508
E. coli 200 knockouts	**0.702**	0.683	0.504
E. coli 200 knockdowns	**0.705**	0.682	0.495
E. coli 200 multifactorial	**0.678**	0.666	0.495
AMAUC	**0.736**	0.689	0.529

To summarize, one of the proposed boosting methods coupled with the under-sampling technique achieves the best performance among all the fifteen algorithms studied in this paper on the yeast and E. coli datasets respectively. For the yeast datasets, this proposed boosting method is Boost III. For the E. coli datasets, this proposed boosting method is Boost II. Both boosting methods coupled with the under-sampling technique are superior to the existing approaches using SVM only [7, 8].

The proposed boosting techniques are based on a weighted decision tree [120]. The boosting techniques are combined with other machine learning algorithms including random forests [121], SVM with the linear kernel, SVM with the sigmoid kernel, SVM with the Gaussian kernel, and SVM with the polynomial kernel of degree 2. However, the performance of these other machine learning algorithms is inferior to the performance of the weighted decision tree used in this paper. As a consequence, the results from the other machine learning algorithms are not reported here.

To evaluate the effectiveness of the proposed sampling and boosting techniques, the following four algorithms have also been tested and compared: (i) the weighted decision tree without boosting and sampling techniques; (ii) the weighted decision tree with boosting techniques only; (iii) the weighted decision tree with sampling techniques only; and (iv) the weighted decision tree with both boosting and sampling techniques. The results from the yeast and E. coli datasets are similar. For example, for the yeast datasets, the AMAUC value for the weighted decision tree without boosting and sampling techniques is 0.45. This is lower than the existing approaches using SVM only (with AMAUC being 0.495 as shown in Table 5.4). When the weighted decision tree is coupled with only the Boost III technique, its AMAUC value is 0.53. When the weighted decision tree is coupled with only the under-sampling technique, its AMAUC value is 0.61. When the weighted decision tree is coupled with both Boost III and under-sampling techniques, its AMAUC value is 0.666 as shown in Table 5.4, which is much higher than the AMAUC value of 0.45 achieved by the weighted decision tree without boosting and sampling techniques.

5.4 Summary

In this chapter a new approach is presented to gene network inference through regulatory link prediction. The proposed approach uses a weighted decision tree as the base learning algorithm coupled with sampling and boosting techniques to improve prediction performance. Experimental results demonstrated the superiority of the proposed approach over existing methods [7, 8], and the effectiveness of the proposed sampling and boosting techniques.

CHAPTER 6

LEARNING APPROACHES TO IMPROVE PREDICTION OF DRUG SENSITIVITY IN BREAST CANCER PATIENTS

6.1 Introduction

Predicting drug response to cancer disease is an important problem in modern clinical oncology that attracted increasing recent attention from various domains such as computational biology, machine learning, and data mining. Cancer patients respond differently to each cancer therapy owing to disease diversity, genetic factors, and environmental causes. Thus, oncologists aim to identify the effective therapies for cancer patients and avoid adverse drug reactions in patients. By predicting the drug response to cancer, oncologists gain full understanding of the effective treatments on each patient, which leads to better personalized treatment. This chapter presents three learning approaches to address the problems. The instance selection approach selects representative training cell lines for guiding the learning model through utilizing a learning scenario. The oversampling approach generates synthetic cell lines to improve the accuracy of prediction algorithms. The hybrid approach selects the top-*k* genes and then selects cell lines as in the first approach. Experimental results show later, the three approaches statistically significantly outperform the baseline drug sensitivity prediction approach proposed by Geeleher *et al*. [54].

6.2 Proposed Approaches

6.2.1 The Instance Selection Approach

Figure 6.1 outlines the instance selection approach, which works as follows. (A) Given expression profiles denoted by $\mathbf{X} \in \mathrm{R}^{mxn}$, which contains *m* cell lines, each cell line corresponding to a sample, and *n* columns, each column corresponding to a gene. To learn a model, the continuous real-values drug responses $\mathbf{Y}$ (i.e., drug IC_{50} values) are needed to be known, where $\mathbf{Y} \in \mathrm{R}^{m}$. A training set is defined as $\mathbf{S} = \{(x_1, y_1), (x_2, y_2), ..., (x_m, \mathrm{y}_m)\}$ where $x_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}$. Similarly, for given expression profiles denoted by $\mathbf{X}^{'} \in \mathrm{R}^{pxn}$, containing *p* samples and *n* genes. The test set is defined as $\mathbf{X}^{'} = \{x_1^{'}, x_2^{'}, ..., x_p^{'}\}$. (B) Slight modification of IPRed is adapted [122], to select representative training cell lines, consisting of the following two steps

1. Store *m* distances, which correspond to the *m* minimum distances between each training cell line $x_i \in \mathbf{X}$ and all tumors of patients (i.e., instances) in $\mathbf{X}^{'}$ defined as

$$w(x_i) = \mathrm{dist}(x_i, x_{j^*}^{'}) \text{ with } j^* = \underset{j \in \{1,2,...,|\mathbf{X}^{'}|\}}{\arg\min}\ \mathrm{dist}(x_i, x_j^{'}) \tag{6.1}$$

where $w(x_i)$ is the distance between x_i and $x_{j^*}^{'}$, s.t. $x_{j^*}^{'}$ is the nearest patient to cell line x_i, $\mathrm{dist}(x_i, x_{j^*}^{'}) = \sqrt{\sum_{d=1}^{n} (x_{id}, x_{j^*d}^{'})^2}$ is the Euclidean distance.

2. Let $\mathbf{W} = (w(x_1), w(x_2), ..., w(x_m))$. Each cell line $x_i \in \mathbf{X}$ is selected, s.t. $w(x_i) < \mathrm{c}$, where $\mathrm{c} = \mathrm{percentile}(\mathbf{W}, \%75), i=1,...,m.$ That is, each cell line whose weight is below the 75th percentile of $\mathbf{W}$ is selected. All cell lines satisfying the condition are then stored in $\mathbf{S}^{'}$. Eventually, $\mathbf{S}^{'} \in \mathrm{R}^{m^{'}xn}$ where $m^{'} < m.$ The learning algorithm

is applied on $\mathbf{S}^{'}$, to induce model h_1. (C) Model h_1 is applied to perform prediction on the test set. In the rest of the paper the approach is referred to as the instance selection approach employing machine learning algorithms (SVR and RR) as: IS+SVR+L, IS+SVR+S, and IS+RR (abbreviations are listed in Table 6.1).

Table 6.1 Abbreviation of Drug Sensitivity Prediction Algorithms

Abbreviation	Prediction Algorithm
IS+SVR+L	The instance selection approach using support vector regression with linear kernel
IS+SVR+S	The instance selection approach using support vector regression with sigmoid kernel
IS+RR	The instance selection approach using ridge regression
O+SVR+L	The oversampling approach using support vector regression with linear kernel
O+SVR+S	The oversampling approach using support vector regression with sigmoid kernel
O+RR	The oversampling approach using ridge regression
C+SVR+L	The hybrid approach using support vector regression with linear kernel
C+SVR+S	The hybrid approach using support vector regression with sigmoid kernel
C+RR	The hybrid approach using ridge regression
SVR+L	The baseline approach using support vector regression using linear kernel
SVR+S	The baseline approach using support vector regression using sigmoid ker-nel
RR	The baseline approach using ridge regression

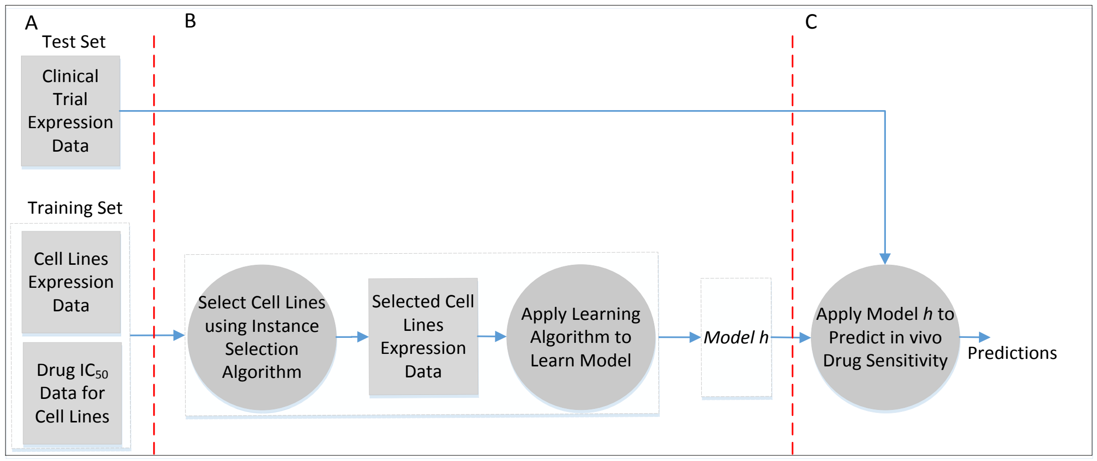

A
Test Set
Clinical Trial Expression Data
Training Set
Cell Lines Expression Data
Drug IC50 Data for Cell Lines
B
Select Cell Lines using Instance Selection Algorithm
Selected Cell Lines Expression Data
Apply Learning Algorithm to Learn Model
Model h
C
Apply Model h to Predict in vivo Drug Sensitivity
Predictions

Figure 6.1 Data flow diagram showing the instance selection approach to predicting in vivo drug sensitivity.

6.2.2 The Oversampling Approach

As shown in Figure 6.2, (A) The oversampling approach receives training set $\mathbf{S}$ and test set $\mathbf{X}'$, which are defined as in Section 4.2.1 (A).

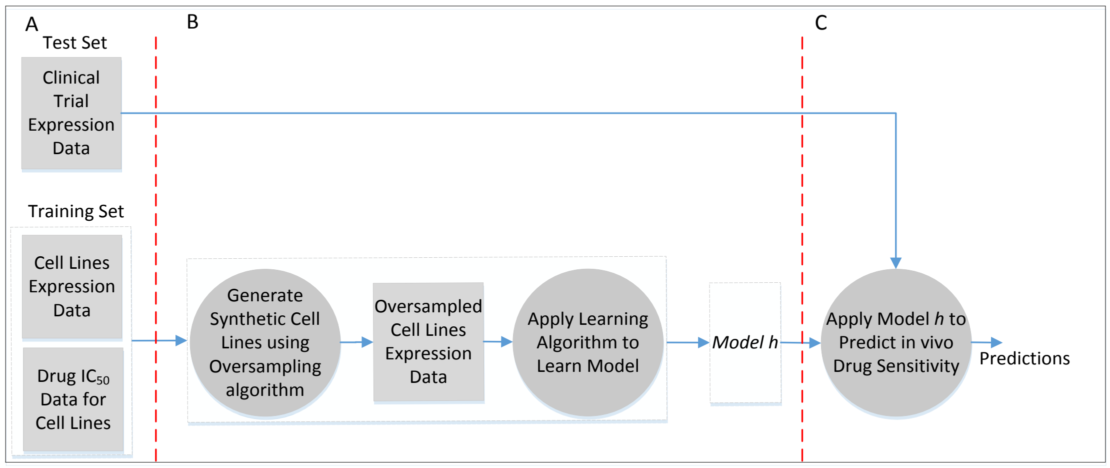

A
Test Set
Clinical Trial Expression Data
Training Set
Cell Lines Expression Data
Drug IC50 Data for Cell Lines
B
Generate Synthetic Cell Lines using Oversampling algorithm
Oversampled Cell Lines Expression Data
Apply Learning Algorithm to Learn Model
Model h
C
Apply Model h to Predict in vivo Drug Sensitivity
Predictions

Figure 6.2 Data flow diagram showing the oversampling approach to predicting in vivo drug sensitivity.

(B) Synthetic Minority Oversampling Approach (SMOTE) is employed [123] for generating synthetic cell lines according to the following steps:

1. Find k nearest neighbors $x_1^*, x_2^*, ..., x_k^*$ of each $x_i \in \mathbf{S}$. (In the study presented here, k=1.)
2. Generate synthetic cell lines along the lines between the randomly selected k nearest neighbors and each x_i by the following lines of code:

```
2.2. for $i$=1 to $m$
    2.1. for $j$=1 to $k$
        2.1.1.  $x_{new} = x_i + (x_j^* - x_i)\lambda$
        2.1.2.  $\text{Label}(x_{new}) = \text{Label}(x_i)$
        2.1.3.  Store $x_{new}$ in $\mathbf{X}^{++}$
    2.2. end for
2.3. end for
```

Where λ .is a random number. A learning algorithm is called on $\mathbf{S'} = \{\mathbf{X}^{++}, \mathbf{S}\}$, to induce model *h*. (C) Model *h* is applied for the prediction on the test set. For brevity, the following abbreviations are assigned to the oversampling approach employing machine learning algorithms: O+SVR+L, O+SVR+S, and O+RR (see Table 6.1).

6.2.3 The Hybrid Approach

Figure 6.3 shows the data flow diagram of the hybrid approach. (A) Given training set $\mathbf{S}$ and test set $\mathbf{X}'$, which are described as in Section 4.2.1. (B) LASSO (least absolute shrinkage and selection operator) has been used successfully in genomics, which performs feature selection and regularization to improve prediction accuracy [124-126].

Given cell lines $x_i \in \mathrm{R}^n$ and the drug response values $y_i \in \mathrm{R},\ i = 1,...,m.$ The objective function for LASSO is

$$\min_{\boldsymbol{\beta} \in \mathrm{R}^{n+1}} \mathrm{R}_{\lambda}(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta} \in \mathrm{R}^{n+1}} \frac{1}{2m} \sum_{i=1}^{m} (y_i - \sum_{j=0}^{n} x_{ij}\beta_j)^2 + \lambda \sum_{j=0}^{n} \| \beta_j \|_1 \qquad (6.2)$$

where $\lambda \geq 0$ is the tuning parameter, $\| \beta_j \|_1 = \sum_{j=0}^{n} | \beta_j |$ is the ℓ_1 penalty. In LASSO penalty, it is expected to have many coefficients in $\boldsymbol{\beta}$ to be close to zero, and small subset to be larger and nonzero [126]. Coordinate descent is applied to solve the problem in Equation 6.2 minimizing one-at-a-time one parameter and fixing all others [127], which works as follows:

1. Initialize all $\beta_j = 0$
2. Cycle over j= 0,1,2,…,n,0,1,2,…, till convergence:

   2.1 $\beta^* = \frac{1}{m} \sum_{i=1}^{m} x_{ij} (y_i - \sum_{l \neq j} x_{il}\beta_l)$

   2.2 $\beta_j = \begin{cases} \beta^* - \lambda \text{ if } \beta^* > 0 \text{ and } \lambda < |\beta^*| \\ \beta^* + \lambda \text{ if } \beta^* < 0 \text{ and } \lambda < |\beta^*| \\ 0 \qquad \text{if } \lambda \leq |\beta^*| \end{cases}$

Step 2.1 measures the correlation of the *j*th gene (x_{ij}) and the partial residual of the predicted drug response value without *j*th gene ($\sum_{l \neq j} x_{il}\beta_l$) and the true value (*y_i*). Higher β^* value means that *j*th gene (feature) is regarded as more correlated and an important gene. Similarly, lower β^* value means that *j*th gene (feature) is regarded as less correlated gene. Step 2.2 updates β_j by soft-thresholding [126, 127].

Coefficients $\boldsymbol{\beta}_{j\neq 0} \in \mathbf{R}^n$ are stored, i.e., consisting of all coefficients except coefficient β_0, produced by coordinate descent in $\mathbf{U}$. Then rerun coordinate descent *q*-1 times (in this study for model h_1, q=10), where each $\boldsymbol{\beta}$ is stored in $\mathbf{U}$. That is, $\mathbf{U} \in \mathrm{R}^{q \times n}$, where $\mathbf{U}$ contains *q* different $\boldsymbol{\beta}$ coefficients produced through rerunning coordinate descent. Then the columns sum is performed as:

$$\mathbf{M} = \mathbf{1}^T \mathbf{U} \tag{6.3}$$

where $\mathbf{1}$ is the *q*-dimensional column vector of all ones, and $\mathbf{M} \in \mathrm{R}^n$. The top-*k* in $\mathbf{M}$ corresponding to the *k* maximum values are then selected and their positions are in $\mathbf{I} \in \mathrm{R}^k$. (In the study presented here, *k*=50.) Top-*k* genes in the training set are selected using positions in $\mathbf{I}$. Cell lines are then selected as in Section 4.2.1 and stored in $\mathbf{S}'$. A learning algorithm is called on $\mathbf{S}'$, to induce model $h_{t=1}$.

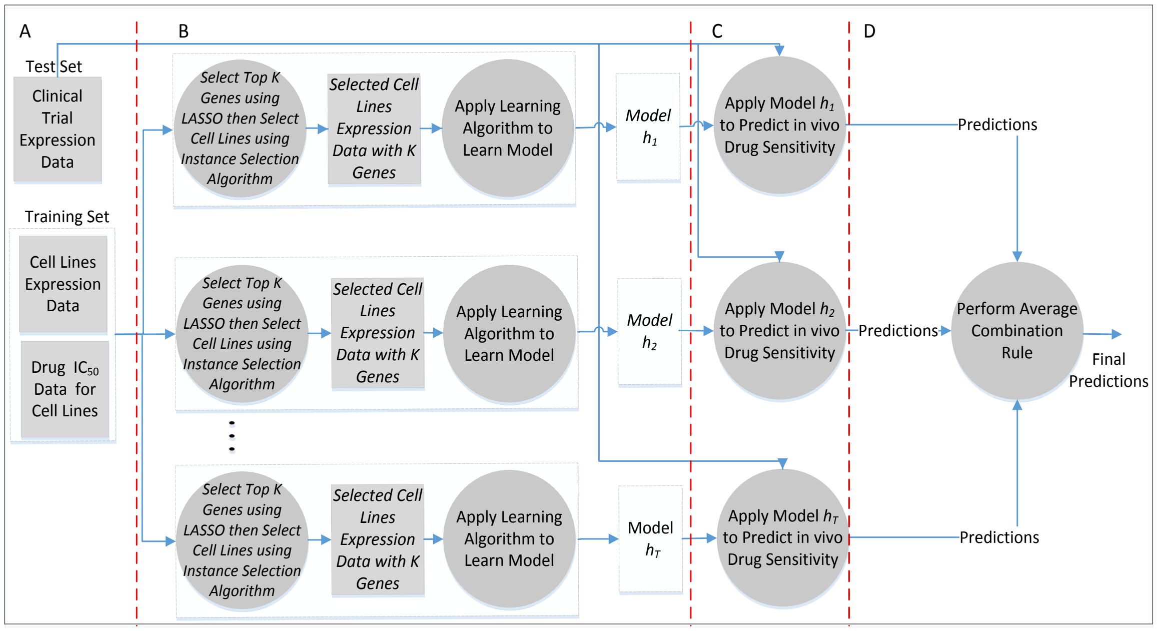

A
B
C
D
Test Set
Clinical Trial Expression Data
Training Set
Cell Lines Expression Data
Drug IC50 Data for Cell Lines
Select Top K Genes using LASSO then Select Cell Lines using Instance Selection Algorithm
Selected Cell Lines Expression Data with K Genes
Apply Learning Algorithm to Learn Model
Model h1
Model h2
Model hT
Apply Model h1 to Predict in vivo Drug Sensitivity
Apply Model h2 to Predict in vivo Drug Sensitivity
Apply Model hT to Predict in vivo Drug Sensitivity
Predictions
Perform Average Combination Rule
Final Predictions

Figure 6.3 Data flow diagram showing the hybrid approach to predicting in vivo drug sensitivity.

(C) Top-*k* genes in the test set are selected using positions of $\mathbf{I}$, and model h_1 is applied on the test set. Such processes (A, B, C) as in Figure 6.3 are repeated *T* times except that *q*=15 (*q*=20, respectively) for *t*=2 (*t*=3, respectively). (In the study presented here, *T*=3.) (D) For each test instance $x_i^{'} \in \mathbf{X}^{'}$ where $i = 1,..., p$, the final prediction is obtained by taking average prediction of T models [128]

$$H(x_i^{'}) = \frac{1}{T}\sum_{t=1}^{T} h_t(x_i^{'}) \tag{6.4}$$

6.3 Experiments and Results

In this section, the performance of the learning approaches is evaluated and compared against the baseline approach proposed by Geeleher *et al.* [54]. Below, the experimental datasets in the study are described, experimental methodology, and then present experimental results.

6.3.1 Experimental Datasets

The training set $\mathbf{S} \in \mathrm{R}^{482 \times 6539}$ consists of $\mathbf{Y} \in \mathrm{R}^{482}$ drug IC_{50} values and cell lines expression data $\mathbf{X} \in \mathrm{R}^{482 \times 6538}$ (i.e., 482 cell lines and 6538 genes). The test set $\mathbf{X}' \in \mathrm{R}^{24 \times 6538}$, which contains 24 breast cancer tumors of patients and 6538 genes. Both the training set and test set were downloaded and processed according to the approach proposed by Geeleher *et al.* [54] as follows. The drug IC_{50} values for docetaxel (chemotherapy drug) [129, 130] were downloaded from (http://genemed.uchicago.edu/~pgeeleher/cgpPrediction/). The cell lines expression data were downloaded from ArrayExpress repository [131] (accession number is E-MTAB-783 or available at https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-783/?query=EMTAB783). Clinical trial data were downloaded from Gene Expression Omnibus (GEO) repository (http://www.ncbi.nlm.nih.gov/geo/) with accession numbers GSE350 and GSE349 [132-134]. Data with accession number GSE350 (GSE349, respectively) contain 10 (14, respectively) samples (i.e., instances). If the remaining of tumor was $< 25\%$ ($\geq 25\%$, respectively), breast cancer patient would be considered as sensitive (resistant, respectively) to docetaxel treatment. Following data processing steps of Geeleher *et al.* [54].

6.3.2 Experimental Methodology

12 drug sensitivity prediction algorithms are considered, which are listed and abbreviated in Table 6.1. SVR+L (SVR+S, respectively) employs the same proposed approach by Geeleher *et al.* [54]. RR is the baseline approach [54]. The remaining prediction algorithms correspond to the prediction algorithms employing the proposed approaches.

Software used in this work included: support vector regression with linear and sigmoid kernels [80], ridge regression [54], LASSO [135], R code for processing datasets and performance evaluation [54].

R is used to write SMOTE [123], IPRed [122], and perform the experiments. The *Area Under the ROC Curve* (AUC) [54] is employed to measure the accuracy of prediction algorithms. The higher AUC an algorithm has, the better performance that algorithm achieves. We assess the stability of prediction algorithm, where stable prediction algorithm is the one for which prediction accuracy on the test set does not change dramatically by small changes to the training set [136, 137]. This kind of assessment is important, where the best prediction algorithm is the one that outperforms other algorithms many times. Statistical significance is measured between all pairs of the prediction algorithms [76, 83].

6.3.3 Experimental Results

Table 6.2 shows the AUC of six prediction algorithms on several runs of the clinical trial data used in the experiment. IS+SVR+L, O+SVR+S and IS+RR represent the top-3 prediction algorithms employing the learning approaches. The other three prediction algorithms employ the existing approach: SVR+S, RR, and SVR+L. For each run, the prediction algorithm having the best performance (i.e., the highest AUC) is shown in bold. Table 6.3 shows the p-values of Wilcox rank test (two tailed) between all pairs of algorithms, where the algorithm with statistical significance is shown in bold. It can be seen from Table 6.2 that IS+SVR+L is better than the prediction algorithms including RR, the proposed approach by Geeleher *et al*. [54]. In particular, IS+SVR+L gives the highest mean AUC (MAUC) of 0.907 and performed the best in 11 out of 11 runs. The second best is O+SVR+S that gives MAUC of 0.883. The third best is IS+RR that achieves MAUC of 0.874. SVR+S performs better than RR giving MAUC of 0.855. The remaining prediction algorithms, RR and SVR+L, yield MAUC of 0.828 and 0.825, respectively. IS+SVR+L and IS+RR keep approximately 75% of the cell lines provided during learning the model (see column ("m+IS") in Table 6.2). The removing of 25% cell lines did not degrade the performance of IS+SVR+L and IS+RR, compared to the baseline RR. Column ("m+O") shows the number of cell lines which consists of all $|m|$ cell lines in the training set plus $|m|$generated synthetic cell lines. Combining the generated synthetic cell lines with the cell lines in the learning stage improved the performance of O+SVR+S when compared to RR (results shown in Table 6.2). These results show the stable performance of the learning approaches under different settings of the training set size.

In Table 6.3, the p-values of Wilcox rank test are reported to measure the statistical significance between algorithms [76]. The p-values indicate that IS+SVR+L statistically significantly outperforms the other algorithms.

Table 6.2 AUC of Prediction Algorithms on the Test Set. The Column ("m") Shows the Number of Cell Lines (Instances) in the Training Set That Were Provided to Each Prediction Algorithm. Column ("m+o") Shows the Number of Cell Lines Used in O+SVR+S. Column ("m+IS") Shows the Number of Cell Lines Used in IS+SVR+L and IS+RR. The Algorithm with the Highest AUC Is Shown in Bold. MAUC, Mean AUC

m	IS + SVR+L	O + SVR+S	m+O	IS + RR	m+IS	SVR+S	RR	SVR+L
482	**0.871**	0.864	964	0.850	361	0.842	0.814	0.835
478	**0.878**	0.864	956	0.871	358	0.871	0.814	0.814
473	**0.900**	0.871	946	0.871	354	0.864	0.821	0.800
468	**0.907**	0.892	936	0.871	351	0.857	0.821	0.821
463	**0.892**	0.871	926	0.871	347	0.857	0.821	0.835
458	**0.900**	0.892	916	0.871	343	0.857	0.821	0.850
453	**0.914**	0.878	906	0.885	339	0.842	0.835	0.828
448	**0.921**	0.892	896	0.885	336	0.864	0.835	0.835
443	**0.928**	0.914	886	0.878	332	0.857	0.842	0.814
438	**0.942**	0.907	876	0.878	328	0.857	0.842	0.821
433	**0.921**	0.871	866	0.878	324	0.842	0.842	0.828
MAUC	0.907	0.883	-	0.874	-	0.855	0.828	0.825

Table 6.3 P-Values of Wilcox Rank Test (Two-Tailed) between All Pairs of Prediction Algorithms. Shown in Bold Is the Prediction Algorithm with Statistical Significance Where p<0.05

	O+SVR+S	IS+RR	SVR+S	RR	SVR+L
IS+SVR+L	**0.0033**	**0.0033**	**0.0033**	**0.0033**	**0.0033**
O+SVR+S	-	0.0727	**0.0051**	**0.0033**	**0.0033**
IS+RR	-	-	**0.0051**	**0.0033**	**0.0033**
SVR+S	-	-	-	**0.0051**	**0.0033**
RR	-	-	-	-	0.7789

In Figure 6.4 all prediction algorithms are ranked from the highest to the lowest MAUC. Each MAUC is calculated over the 11 runs on the clinical data. It can be seen from Figure 6.4 that the three proposed approaches outperform RR [54] ranked the 9th w.r.t MAUC.

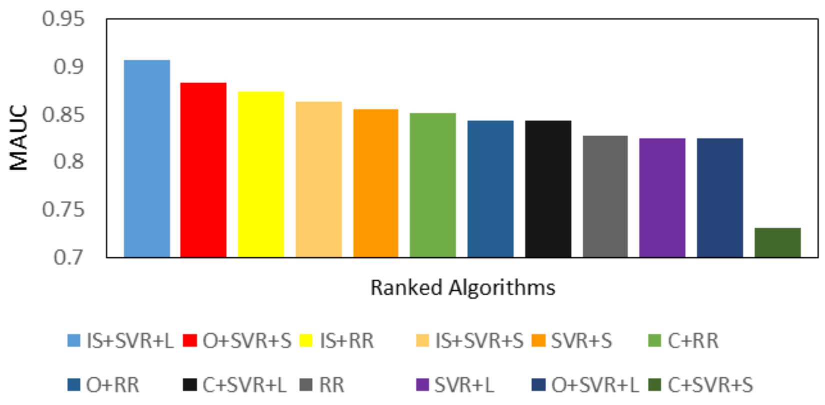


Figure 6.4 Mean AUC (MAUC) of prediction algorithms. Prediction algorithm with highest MAUC (to be ranked first from left) and lowest MAUC (to be ranked the last from the left.

Figure 6.5 shows the predictions of three prediction algorithms on the test set (clinical data of 24 breast cancer patients.) using training set with *m*=482 (i.e., the complete training set without any change). Figure 6.5(a) (Figure 6.5(b), Figure 6.5(c), respectively) shows the predictions of IS+SVR+L (SVR+S, RR, respectively). Figures 6.5(a) and 6.5(b) show the difference between the predicted drug sensitivity in breast cancer patients was statistically significant (*P*= 0.001 from a *t*-test) between trial-defined sensitivity and resistant groups for IS+SVR+L and SVR+S, respectively. RR (see Figure 6.5(c)) achieved statistical significance (*P*= 0.004 from a *t*-test). Training set size is 482 when learning a model. In Figure 6.5(d) the ROC reveals AUC of 0.871, 0.842, and 0.814 for IS+SVR+L, SVR+S, and RR, respectively, as in Table 6.2.

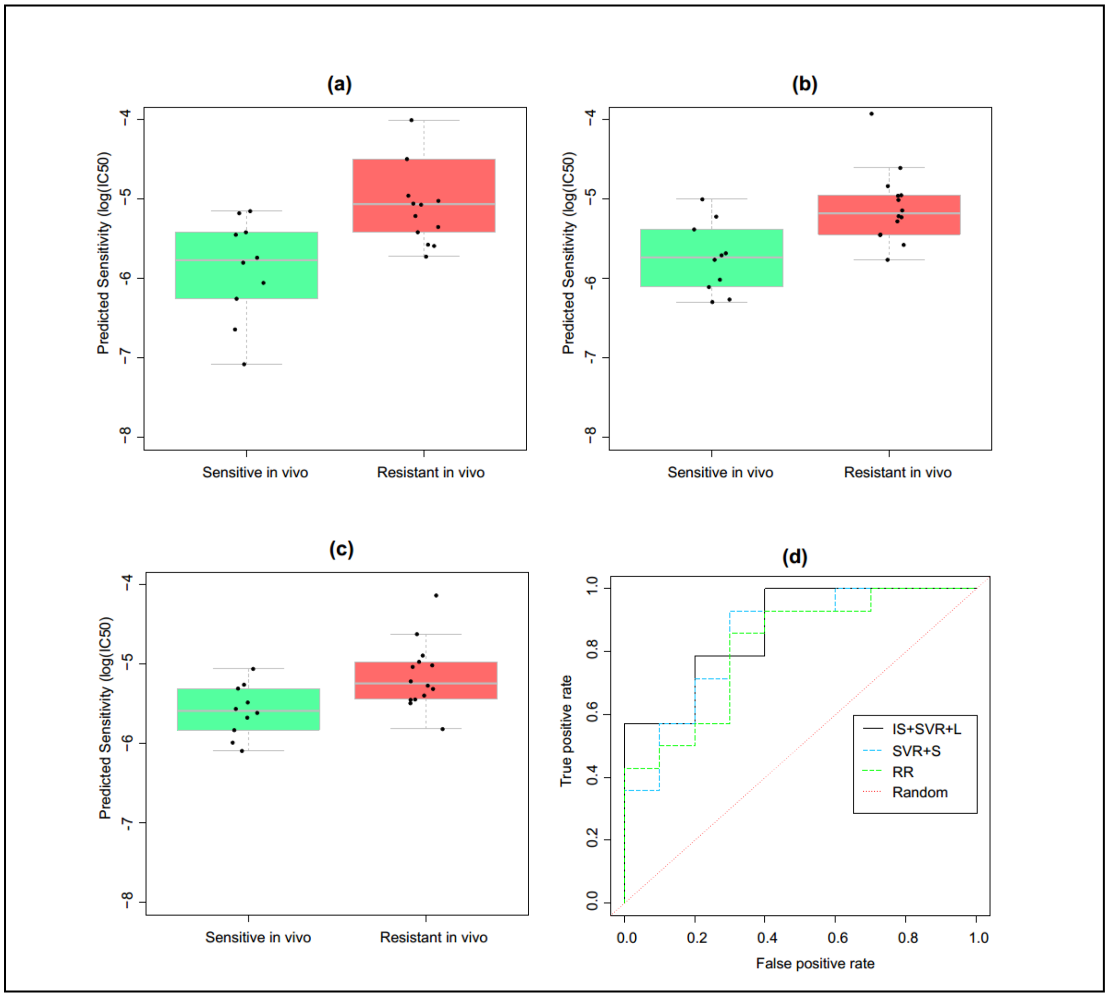


Figure 6.5 Prediction of docetaxel sensitivity in breast cancer patients. Strip chart a (b,c, respectively) showing the difference in predicted drug sensitivity for individuals sensitive or resistant to docetaxel treatment using IS+SVR+L (SVR+S, RR, respectively) prediction algorithm. (d) ROC curves of prediction algorithms showing the proportion of true positives against the proportion of false positives. ROC, receiver operating characteristic.

It is worth mentioning that the performance of other machine learning algorithms are assessed, including random forests [78], support vector regression with polynomial kernel of degree 2, support vector regression with Gaussian kernel. However, these algorithms did not show good predictive performance (results not shown here).

6.4 Summary

This chapter introduces three learning approaches to improve the prediction of drug sensitivity. The first learning approach employs (i) IPRed algorithm to select cell lines. The second learning approach employs (i) SMOTE algorithm to generate synthetic cell lines. The third learning approach employs (i) LASSO for gene selection, (ii) IPRed algorithm to select cell lines, and (iii) ensemble averaging of predictions obtained from different models.

The learning approaches use two machine learning algorithms: support vector regression and ridge regression. The experimental results on clinical trial data of breast cancer patients demonstrate the stable performance of the learning approaches achieving statistically significant improvements over the existing approach.

CHAPTER 7

A LINK PREDICTION APPROACH TO CANCER DRUG SENSITIVITY PREDICTION

7.1 Introduction

Predicting the response to a drug for cancer disease patients based on genomic information is an important problem in modern clinical oncology. This problem occurs in part because many available drug sensitivity prediction algorithms do not consider better quality cancer cell lines and the adoption of new feature representations; both lead to the accurate prediction of drug responses. By predicting accurate drug responses to cancer, oncologists gain a more complete understanding of the effective treatments for each patient, which is a core goal in precision medicine.

In this chapter, cancer drug sensitivity is modeled as a link prediction, which is shown to be an effective technique. The proposed link prediction algorithms are evaluated and compared with an existing drug sensitivity prediction approach based on clinical trial data. The experimental results based on the clinical trial data show the stability of the proposed link prediction algorithms, which yield the highest *area under the ROC curve* (AUC) and are statistically significant.

7.2 Related Work

7.2.1 Link Prediction in Gene Regulatory Networks

Given *m* genes, in which each gene has *n* expression values, their gene expression profiles can be denoted by $\mathbf{G} \in \mathbb{R}^{m \times n}$, which contains *m* rows—each row corresponds to a gene—and *n* columns—each column corresponds to an expression value [16].

To learn a model, the regulatory relationships (i.e., labels) among the genes are needed to be known, which are stored in the matrix $\mathbf{H} \in \mathbb{R}^{p \times 3}$. $\mathbf{H}$ contains *p* rows—each row shows a known regulatory relationship between two genes—and three columns. The first column shows the source gene (i.e., the transcription factor). The second column shows the target gene, and the third column shows the label, which is denoted as +1 (i.e., present link) when the source gene regulates the target gene or -1 (i.e., missing link) when the source gene does not regulate the target gene. Thus, $\mathbf{H}$ represents the observed (i.e., known) gene regulatory network. To learn a model, the training set $\mathbf{D} \in \mathbb{R}^{p \times 2n+1}$ is needed to be constructed. The *p* examples in **D** are constructed as follows: For each pair of genes with the associated label in matrix $\mathbf{H}$, the *n* expression values of each pair of genes in matrix $\mathbf{G}$ are extracted, and the concatenation of the *n* expression values of each pair of genes and the corresponding label is performed. For example, consider the *i*th example in the training set **D**, which is denoted by $\mathbf{D}_i$ and defined as

$$\mathbf{D}_i = [g_i^1, g_i^2, \ldots, g_i^n, g_l^1, g_l^2, \ldots, g_l^n, y_i], \qquad (7.1)$$

where $g_i^1, g_i^2, \ldots, g_i^n$ are the *n* expression values of $\mathbf{g}_i$ (also called the expression profile of $\mathbf{g}_i$), $g_l^1, g_l^2, \ldots, g_l^n$ are the *n* expression values of $\mathbf{g}_l$, and $y_i \in \{1, -1\}$. The *i*th example of the test set, $\mathbf{T}$, is denoted by $\mathbf{T}_i$ and constructed as follows:

$$\mathbf{T}_i = [g_i^1, g_i^2, \ldots, g_i^n, g_j^1, g_j^2, \ldots, g_j^n], \qquad (7.2)$$

where $g_i^1, g_i^2, \ldots, g_i^n$ are the *n* expression values of $\mathbf{g}_i$, and $g_j^1, g_j^2, \ldots, g_j^n$ are the *n* expression values of $\mathbf{g}_j$. These feature vector definitions have been used by the existing supervised inference of gene regulatory networks [4, 5, 7, 16, 17, 110]. After constructing the feature vectors, the learning algorithm is applied to $\mathbf{D}$ to induce (i.e., learn) the model *h*. The resulting model is used to perform prediction on $\mathbf{T}$. The known regulations among genes enable using the induction principle to predict new regulations (i.e., labels): If gene $\mathbf{g}_j$ has an expression profile that is similar to gene $\mathbf{g}_l$, which is known to be regulated by $\mathbf{g}_i$, then $\mathbf{g}_j$ is likely to be regulated by $\mathbf{g}_i$ [8]. Genes with similar expression profiles that are likely to be co-regulated have been used in the unsupervised clustering of expression profiles [138-140].

7.2.2 Cancer Drug Sensitivity Prediction

The gene expression profiles denoted by $\mathrm{X} \in \mathbb{R}^{p \times n}$, which contains *p* rows—each row corresponds to a cell line or a sample—and *n* columns—each column corresponds to a gene. $\mathrm{Y} = (y_1, ..., y_p)^T$ consists of the corresponding real-value drug responses (i.e., drug IC$_{50}$ values) to X, where $\mathrm{Y} \in \mathbb{R}^p$ (i.e., the *p*-dimensional column vector). IC$_{50}$ is defined as the concentration of a compound that is required to produce 50% cancer cell growth inhibition after 48 hours of treatment [66]. A training set is defined as $\mathrm{D}=\{(\mathbf{g}_i, y_i)\}_{i=1}^{p}$, where $\mathbf{g}_i \in \mathrm{X}$ and $y_i \in \mathrm{Y}$. Let the *i*th example of the training set $\mathbf{D}$, denoted by $\mathbf{D}_i$, be defined as

$$\mathbf{D}_i = [g_i^1, g_i^2, ..., g_i^n, y_i], \tag{7.3}$$

where $g_i^1, g_i^2, ..., g_i^n$ represent the *n* genes of the cancer cell line $\mathbf{g}_i$ (also called the expression profile of $\mathbf{g}_i$), and $y_i \in \mathbb{R}$ is the drug response value.

The *i*th example of the test set $\mathbf{T}$, denoted by $\mathbf{T}_i$, is constructed as follows:

$$\mathbf{T}_i = [g_j^1, g_j^2, \ldots, g_j^n]. \tag{7.4}$$

These feature vector definitions have been used by existing supervised cancer drug sensitivity prediction algorithms [52-54, 56, 66, 141, 142]. A learning algorithm is applied to $\mathbf{D}$ to induce model *h*, which is subsequently used to perform predictions on $\mathbf{T}$. Known cancer cell lines with associated drug responses enabled the use of the induction principle: If tumor $\mathbf{g}_j$ has an expression profile similar to $\mathbf{g}_i$, then $\mathbf{g}_j$ is likely to have a drug response value closer to the drug response value associated with $\mathbf{g}_i$.

7.3 Methods

The fundamental task of cancer drug sensitivity prediction is to correctly predict the response of a tumor to the drug. This prediction is typically achieved based on *how closely* this tumor (also referred to as the test example) is related to a known cancer cell line with the associated drug response. Proximity, which is a measure of closeness, lies at the heart of both link prediction in gene regulatory networks and cancer drug sensitivity prediction [8, 143].

7.3.1 Feature Vector Construction

To bridge link prediction and cancer drug sensitivity, the feature representations of Equations (7.3) and (7.4) are transformed to the corresponding Equations (7.1) and (7.2) as follows: Let $\{(\mathbf{g}_i, y_i)\}_{i=1}^{p} \subseteq \mathbf{D}$ be the cancer cell lines, where $\mathbf{D} \in \mathbb{R}^{p \times n+1}, b = p$.

3. Find the k' nearest neighbors $\mathbf{g}_1^*, \mathbf{g}_2^*, \ldots, \mathbf{g}_{k'}^*$ of each $\mathbf{g}_i$ in $\mathbf{D}$. (In this study k'=1.)
4. Generate synthetic cell lines along the lines between the randomly selected k' nearest neighbors and each $\mathbf{g}_i$ using the following lines of code:

```
2.1 for $i$=1 to $p$
    2.1.1 for $j$=1 to $k'$
        2.1.1.1 $b = b + 1$
        2.1.1.2 $\mathbf{g}_b = \mathbf{g}_i + (\mathbf{g}_j^* - \mathbf{g}_i)\lambda$
        2.1.1.3 Store $[\mathbf{g}_i, \mathbf{g}_b, y_i]$ in $\mathbf{G}$
    2.1.2 end for
2.2 end for
```

where the index *b* refers to only those synthetic cell lines (e.g., $\mathbf{g}_{p+1}$ when the index *b*=*p*+1) that differ from the cell lines in $\mathbf{D}$, whose indexes run from 1 to *p*, $\lambda = 0.3$, and $\mathbf{G} \in \mathbb{R}^{p \times 2n+1}$ is the new feature representation of the cell lines of the training set. Step 2.1.1.2 creates the synthetic cell line $\mathbf{g}_b$. Let $\mathbf{G}_i$ be the *i*th row of $\mathbf{G}$, defined as

$$\mathbf{G}_i = [g_i^1, g_i^2, \ldots, g_i^n, g_{p+1}^1, g_{p+1}^2, \ldots, g_{p+1}^n, y_i], \tag{7.5}$$

where $g_i^1, g_i^2, \ldots, g_i^n$ represent *n* genes of the cancer cell line $\mathbf{g}_i$, $g_{p+1}^1, g_{p+1}^2, \ldots, g_{p+1}^n$ represent the synthetic *n* genes of the synthetic cancer cell line $\mathbf{g}_{p+1}$, and $y_i \in \mathbb{R}$ denotes that both $\mathbf{g}_i$ and $\mathbf{g}_{p+1}$ are linked by sharing the same drug response value. Let $\{(\mathbf{g}_i, y_i)\}_{i=1}^{q} \subseteq \mathbf{T}$ be the test set of tumors, where $\mathbf{T} \in \mathbb{R}^{q \times n}$. Note that Steps 1–2 are similar to the Synthetic Minority Oversampling Approach (SMOTE) [123, 144], However, Step 2.1.1.3 is a different core step in which the dimensionality (i.e., the number of features) is increased instead of the size, as SMOTE does. We then apply the previous steps (i.e., Steps 1 and 2—changing Step 2.1 to i=1 to q and Step 2.1.1.3 to Store $[\mathbf{g}_i, \mathbf{g}_b]$ in $\mathbf{G}'$) to $\mathbf{T}$ to obtain $\mathbf{G}' \in \mathbb{R}^{q \times 2n}$. $\mathbf{G}'$ is the new feature representation of the clinical trial expression data of the test set. Let $\mathbf{G}_i'$ be the ith row of $\mathbf{G}'$, which is defined as

$$\mathbf{G}_i' = [g_j^1, g_j^2, \ldots, g_j^n, g_{p+2k'+1}^1, g_{p+2k'+1}^2, \ldots, g_{p+2k'+1}^n]. \tag{7.6}$$

where $g_j^1, g_j^2, \ldots, g_j^n$ represent n genes of tumor $\mathbf{g}_j$, and $g_{p+2k'+1}^1, g_{p+2k'+1}^2, \ldots, g_{p+2k'+1}^n$ represent n synthetic genes of the synthetic tumor $\mathbf{g}_{p+2k'+1}$. A learning algorithm is called on the training set, $\mathbf{G}$ to induce the model h, which is subsequently used to perform predictions on the test set $\mathbf{G}'$. The logic behind the mechanism of the induction principle is as follows: If the expression profiles of the pair of tumors $(\mathbf{g}_j, \mathbf{g}_{p+2k'+1})$ are similar to those of the cell lines $(\mathbf{g}_i, \mathbf{g}_{p+1})$, then $(\mathbf{g}_j, \mathbf{g}_{p+2k'+1})$ is likely to have a drug response value closer to the drug response value associated with $(\mathbf{g}_i, \mathbf{g}_{p+1})$. In machine learning terms, let $(\mathbf{g}_i, \mathbf{g}_{p+1}, y_i) \in \mathbb{R}^{2n+1}$ be a row feature vector that encodes information about the pair of cancer cell lines $(\mathbf{g}_i, \mathbf{g}_{p+1})$. Given a new pair of tumors encoded by $(\mathbf{g}_j, \mathbf{g}_{p+2k'+1})$, if $(\mathbf{g}_j, \mathbf{g}_{p+2k'+1})$ has feature values similar to $(\mathbf{g}_i, \mathbf{g}_{p+1})$, whose label is y_i, then $(\mathbf{g}_j, \mathbf{g}_{p+2k'+1})$ is more likely to have a closer response (i.e., label) value to y_i.

7.3.2 Notations and Algorithms

7.3.2.1 Notations. To provide a better understanding of the proposed prediction algorithms, the notations used throughout the remainder of this paper are summarized as follows: Matrices are denoted by boldface uppercase letters, e.g., matrix $\mathbf{X}$. The row vectors of a matrix is denoted by boldface uppercase letters with a subscript, e.g., $\mathbf{X}_j$ is the *j*th row of matrix $\mathbf{X}$. Vectors are denoted by boldface lowercase letters, e.g., vector $\mathbf{v}$. Vector entries are denoted by italic lowercase letters with a subscript, e.g., v_i is the *i*th entry of vector $\mathbf{v}$. The number of entries of a vector is denoted by the cardinality symbol, e.g. $|\mathbf{v}|$ is the number of elements of vector $\mathbf{v}$. Scalars are denoted by italic lowercase letters, e.g., *m*. $f, f^*,\text{and } h$ are reserved letters, where f refers to a learning algorithm (e.g., SVR), f^* refers to an induced (i.e., learned) model, and h is an induced model used to perform predictions on the test set. specific learning algorithms and induced models are referred to using subscripts. For example, $f_i(f_i^*, \text{respectively})$ denotes the *i*th learning algorithm and induced model, respectively.

7.3.2.2 The Supervised Link Prediction Algorithm (A1). Figure 7.1 outlines the supervised link prediction algorithm, which is designated as A1, as follows. (A) Given a training set of cancer cell lines with associated drug responses $\mathbf{D}\in\mathbb{R}^{p\times n+1}$ and a test set of tumors $\mathbf{T}\in\mathbb{R}^{q\times n}$ that are described as in cancer drug sensitivity prediction subsection. (B) Transform $\mathbf{D}$ and $\mathbf{T}$ using the feature vector construction method described in feature vector construction subsection, to obtain a new feature representation $\mathbf{G}\in\mathbb{R}^{p\times 2n+1}$ for the training set and a new feature representation $\mathbf{G}^{'}\in\mathbb{R}^{q\times 2n}$ for the test set. (C) The proposed link filtering method aims to select a better quality training set that works as follows: Each row (i.e., feature vector) in the new representations $\mathbf{G}$ and $\mathbf{G}^{'}$ can be viewed as a cell line or tumor, represented by a 2*n*-dimensional row vector when the drug responses of the training set $\mathbf{G}$ are excluded. Each cell line [122] $\mathbf{g}_i$ in the training set $\mathbf{G}$ is weighted by the minimum distance from the cell line $\mathbf{g}_i$ to all tumors $\mathbf{g}^{'}_j$ in the testing set $\mathbf{G}^{'}$:

$$w_i = \text{dist}(\mathbf{g}_i, \mathbf{g}^{'}_{j^*}) \text{ with } j^* = \arg\min_{j\in\{1,\ldots,q\}} \text{dist}(\mathbf{g}_i, \mathbf{g}^{'}_j), \tag{7.7}$$

where $\mathbf{g}_i \in \mathbb{R}^{2n}, \mathbf{g}^{'}_j \in \mathbb{R}^{2n}$, w_i is the weight assigned to $\mathbf{g}_i$, and $\text{dist}(\mathbf{g}_i, \mathbf{g}^{'}_{j^*})$ is the Euclidean distance. Let $\boldsymbol{w} = (w_1, w_2, \ldots, w_p)$. Then, the following steps are performed to select better quality training cell lines using the modified version of *Query by Committee* (QBC) [145-147]:

1. Let *med* be the median of the $\boldsymbol{w}$ vector of weights of each $\mathbf{g}_i$ in $\mathbf{G}$

2. Let $\mathbf{X} = \{(\mathbf{g}_i, y_i) \mid (\mathbf{g}_i, y_i) \in \mathbf{G} \text{ and } w_i \le med\}$

3. Let $\mathbf{X}^{'} = \{\mathbf{g}_i \mid \mathbf{g}_i \text{ in } \mathbf{G} \text{ and } w_i \le med\}$

4. Let $\mathbf{Z} = \{(\mathbf{g}_i, y_i) \mid (\mathbf{g}_i, y_i) \in \mathbf{G} \text{ and } w_i \ge med\}$

5. Let $\mathbf{Z}^{'} = \{\mathbf{g}_i \mid \mathbf{g}_i \text{ in } \mathbf{G} \text{ and } w_i \ge med\}$

6. Apply the learning algorithm, f_1 or f_2, to $\mathbf{X}$ or $\mathbf{Z}$, respectively, to induce the model $f_1^* \, (f_2^*, \text{respectively})$. (In this study, ridge regression is chosen as the learning algorithm)

7. Apply the model $f_1^* \, (f_2^*,$ respectively) to perform predictions on $\mathbf{Z}^{'}$ or $\mathbf{X}'$, respectively) and store predictions in $\mathbf{v}$ or $\mathbf{b}$ respectively)

8. Let $q = |\mathbf{v}| = |\mathbf{b}|$

9. Let $\mathbf{P} = (\mathbf{v}, \mathbf{b})^T$

10. Let $\mathbf{r} = \{y_i \mid y_i \text{ in } \mathbf{Z}\}$ and $\mathbf{e} = \{y_i \mid y_i \text{ in } \mathbf{X}\}$

11. Let $\mathbf{R} = (\mathbf{r}, \mathbf{e})^T$

12. $j^* = \underset{j \in \{1,2\}}{\arg\max} \frac{1}{q} (\mathbf{P}_j - \mathbf{R}_j)^2$

13. $\mathbf{S} = \begin{cases} \mathbf{X} \text{ if } j^* == 1 \\ \mathbf{Z} \text{ otherwise} \end{cases}$

14. $\mathbf{U} = \begin{cases} \mathbf{Z} \text{ if } j^* == 1 \\ \mathbf{X} \text{ otherwise} \end{cases}$

QBC aims to partition the training set $\mathbf{G}$ into $\mathbf{S}$ and $\mathbf{U}$, where $\mathbf{S}$ or $\mathbf{U}$ is treated as the labeled or unlabeled set, respectively. QBC is accompanied by two major items: (1) the set of models (i.e., the committee) that are consistent with all labeled cell lines in $\mathbf{S}$; and (2) given the unlabeled set, $\mathbf{U}$, the QBC applies the models (i.e., the committee) to $\mathbf{U}$ to select the unlabeled tumor that maximizes the disagreement because it represents the most important tumor that will be added to $\mathbf{S}$, in addition to querying the drug response value associated with the tumor. The main obstacle of the first major step of QBC is to find models that agree on all the labels of set $\mathbf{S}$ with reasonable computational complexity [147]. Thus, the first major step is relaxed according to Steps 1–14, where relaxation is practiced to address the first major step [145]. Steps 1–5 partition the training set into $\mathbf{X}$ and $\mathbf{Z}$ using the median as a threshold, where $\mathbf{X}$ or $\mathbf{Z}$ contains cell lines from $\mathbf{G}$ that are near or far, respectively, from the test set $\mathbf{G}'$. Steps 6–14 aim to assign the set of cell lines where the model incurred fewer errors (or more errors, respectively) to $\mathbf{S}$ or $\mathbf{U}$, respectively. The logic behind these steps (i.e., Steps 13–14) is that **S** or **U**, respectively, is wanted to contain the set of cell lines that are more or less, respectively, correctly labeled by one model (i.e., one member of the committee). Steps 1–14 are motivated by other QBC approaches [145-147], in which the success of the second major step of QBC is dependent on the first major step.

15. Repeat *k''* times

15.1. Apply the learning algorithms $f_1, f_2, ..., f_t$ on $\mathbf{S}$ to induce the models (i.e., committee) $f_1^*, f_2^*, ..., f_t^*$. (In this study, *t*=3, and the learning algorithms include support vector regression with a linear kernel (SVR+L), SVR with a polynomial kernel of degree 5, and SVR with a sigmoid kernel (SVR+S))

15.2. Let $w_i^{'}$ be the weight of the *i*th model f_i^* where $\boldsymbol{w}^{'} = \sum_{i=1}^{t} w_i^{'} = 1$. (In this study, *t*=3 and $w_1^{'} = w_2^{'} = w_3^{'} = \frac{1}{3}$)

15.3. For each $\mathbf{g}_j$ in $\mathbf{U}$, let $f^{'}(\mathbf{g}_j) = \sum_{i=1}^{t} \boldsymbol{w}_i^{'} f_i^*(\mathbf{g}_j)$ where $f_i^*(\mathbf{g}_j)$ is the prediction of the *i*th learned model on the *j*th cell line $\mathbf{g}_j$, and $f^{'}(\mathbf{g}_j)$ is the weighted ensemble average of the *j*th cell line $\mathbf{g}_j$.

15.4. Find the cell line $\mathbf{g}_{j^*}$ that maximizes the disagreement:

15.4.1. $j^* = \arg\max_{j \in \{1,...,|\mathbf{v}|\}} \sum_{i=1}^{t} \boldsymbol{w}_i^{'} (f_i^*(\mathbf{g}_j) - f^{'}(\mathbf{g}_j))^2$

15.5. Find the label y_{j^*} of $\mathbf{g}_{j^*}$ in $\mathbf{U}$

15.6. Add the pair $(\mathbf{g}_{j^*}, y_{j^*}) \in \mathbf{U}$ to $\mathbf{S}$ and remove the pair $(\mathbf{g}_{j^*}, y_{j^*})$ from $\mathbf{U}$

15.7. Update $|\mathrm{v}| = |\mathrm{v}| - 1$

16. Return **S**

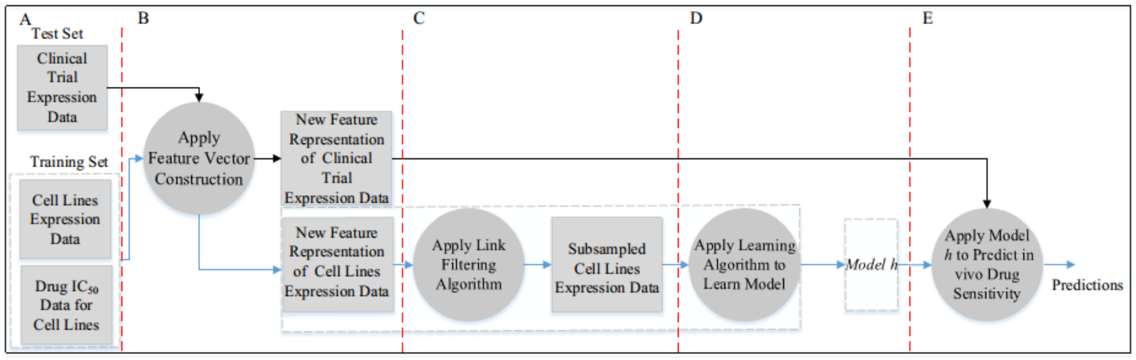

A
Test Set
Clinical Trial Expression Data
Training Set
Cell Lines Expression Data
Drug IC$_{50}$ Data for Cell Lines
B
Apply Feature Vector Construction
New Feature Representation of Clinical Trial Expression Data
New Feature Representation of Cell Lines Expression Data
C
Apply Link Filtering Algorithm
Subsampled Cell Lines Expression Data
D
Apply Learning Algorithm to Learn Model
Model h
E
Apply Model h to Predict in vivo Drug Sensitivity
Predictions

Figure 7.1 Data flow diagram that shows the proposed supervised link prediction algorithm to predict in vivo drug sensitivity.

Steps 15.1–15.4.1 return the index of the cell line in set $\mathbf{U}$ that maximizes the disagreement, where disagreement is defined in Step 15.4.1 [148]. Then, $(\mathbf{g}_{j^*}, y_{j^*})$ is added to or removed from $\mathbf{S}$ or $\mathbf{U}$ respectively, as shown in Steps 15.5–15.6. (In this study, *k''=5*.) Step 15.7 updates |v| as the size of U is reduced after each iteration. **S** (Step 16) is the returned set that will be used as the training set. (D) A learning algorithm is applied on $\mathbf{S}$ to induce the model *h*. Finally (i.e., (E in Figure 7.1)), model *h* is applied to perform predictions on the test set $\mathbf{G}^{'}$ (i.e., the set of new feature representations of the clinical trial expression data). In the remainder of this paper, the supervised link prediction algorithms that employ the following machine learning algorithms (SVR and RR) is referred to as: A1+SVR+L, A1+SVR+S, and A1+RR (abbreviations are listed in Table 7.1).

Table 7.1 Abbreviations of the Drug Sensitivity Prediction Algorithms

Abbreviation	Prediction Algorithm
A1+SVR+L	The supervised link prediction algorithm using support vector regression with a linear kernel
A1+SVR+S	The supervised link prediction algorithm using support vector regression with a sigmoid kernel
A1+RR	The supervised link prediction algorithm using ridge regression
A2+SVR+L	The extended supervised link prediction algorithm using support vector regression with a linear kernel
A2+SVR+S	The extended supervised link prediction algorithm using support vector regression with a sigmoid kernel
A2+RR	The extended supervised link prediction algorithm using ridge regression
B+SVR+L	The baseline approach using support vector regression with a linear kernel
B+SVR+S	The baseline approach using support vector regression with a sigmoid kernel
B+RR	The baseline approach using ridge regression

7.3.2.3 The Extended Supervised Link Prediction Algorithm (A2). Figure 7.2 shows the data flow diagram of the extended supervised link prediction (A2). Steps (A), (B), and (C) are the same as Steps (A), (B), and (C) of the supervised link prediction algorithm. (D) Mahoney *et al.* [149] proposed CUR matrix decomposition as a dimensionality reduction paradigm that aims to obtain a low rank approximation of matrix $\mathbf{S}$, which is expressed in terms of the actual rows and columns of the original matrix $\mathbf{S}$:

$$\mathbf{S} \approx \mathbf{CUR}, \tag{7.8}$$

where $\mathbf{C}$ consists of a small number of the actual columns of $\mathbf{S}$, $\mathbf{R}$ consists of a small number of the actual rows, and $\mathbf{U}$ is a constructed matrix that guarantees that $\mathbf{CUR}$ is close to $\mathbf{S}$. k genes are selected based on their importance score (refer to Equation 7.9), which depends on matrix $\mathbf{S}$ and the input rank parameter l (in this study, the default parameter value for l in CUR function is used [150].) If v_j^{ξ} is the j-th element of the

$\xi-\text{th}$ right singular vector of $\mathbf{S}$, then the normalized statistical leverage scores are equal to

$$\pi_j = \frac{1}{l}\sum_{\xi=1}^{l}(v_j^{\xi})^2 \tag{7.9}$$

for all j=1..2n, and $\sum_{j=1}^{2n}\pi_j = 1$. Statistical leverage scores have been successfully employed in data analysis to identify the most influential genes and outlier detection [149]. A high statistical leverage score for a given gene indicates that the gene is regarded as an important (i.e., influential) gene. A low statistical leverage score for a given gene indicates that the gene is regarded as a less important gene. The indexes of the highest *k* leverage scores are stored in $\mathbf{I}$; these correspond to the positions of the *k* most influential genes in matrix $\mathbf{S}$. *k* genes are selected from the training set $\mathbf{S}$ using their positions in $\mathbf{I}$ and store subsampled cell line expression data with *k* genes in $\mathbf{S}'$. (E) A learning algorithm is called on $\mathbf{S}'$ to induce model *h*. (F) The *k* genes in the test set $\mathbf{G}'$ are selected using their positions in $\mathbf{I}$ and stored in $\mathbf{G}''$.

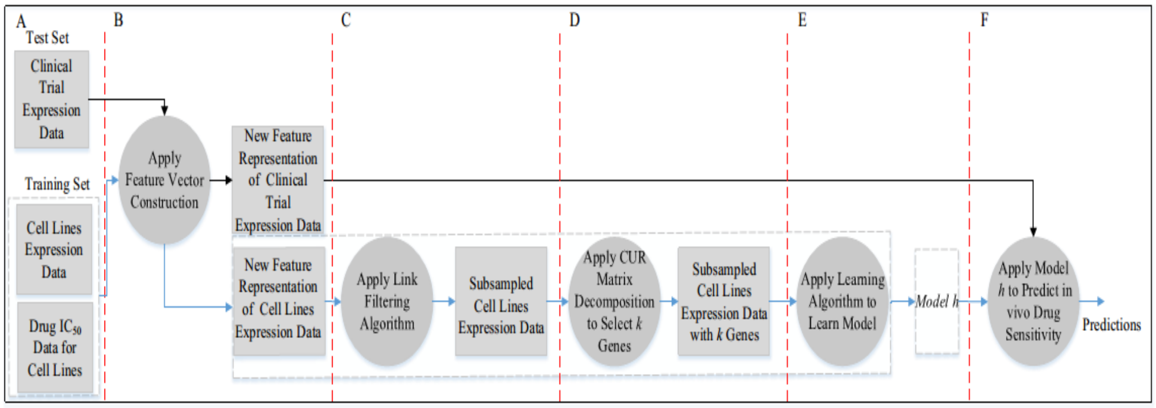

A
Test Set
Clinical Trial Expression Data
Training Set
Cell Lines Expression Data
Drug IC50 Data for Cell Lines
B
Apply Feature Vector Construction
C
New Feature Representation of Clinical Trial Expression Data
New Feature Representation of Cell Lines Expression Data
Apply Link Filtering Algorithm
D
Subsampled Cell Lines Expression Data
Apply CUR Matrix Decomposition to Select k Genes
E
Subsampled Cell Lines Expression Data with k Genes
Apply Learning Algorithm to Learn Model
Model h
F
Apply Model h to Predict in vivo Drug Sensitivity
Predictions

Figure 7.2 Data flow diagram showing the major steps in the proposed extended supervised link prediction algorithm to predict in vivo drug sensitivity.

Model *h* is applied on the test set $\mathbf{G}''$ to perform predictions. The extended supervised link prediction algorithms that employ machine learning algorithms is referred to as A2+SVR+L, A2+SVR+S, and A2+RR (see Table 7.1).

7.4 Experiments and Results

The proposed approach is empirically evaluated and compared against the baseline approach proposed by Geeleher *et al*. [54] on clinical trial datasets. This section first describes the datasets and experimental methodology and presents the experimental results.

7.4.1 Datasets

Data Pertaining to Breast Cancer. The training set $\mathbf{D} \in \mathbb{R}^{482 \times 6539}$ contains 482 cancer cell lines, 6,538 genes, and drug IC_{50} values that correspond to a 482-dimensional column

vector. The test set $\mathbf{T} \in \mathbb{R}^{24 \times 6538}$ consists of 24 breast cancer tumors and 6538 genes. The drug IC_{50} values for docetaxel (a chemotherapy drug) [129, 130] were downloaded from (http://genemed.uchicago.edu/~pgeeleher/cgpPrediction/). The cell line expression data were downloaded from the ArrayExpress repository [131] (accession number is E-MTAB-783, also available at https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-783/?query=EMTAB783).
The clinical trial data corresponding to the test set were downloaded from the Gene Expression Omnibus (GEO) repository (http://www.ncbi.nlm.nih.gov/geo/) with accession numbers GSE350 and GSE349 [132-134]. The data with accession numbers GSE350 and GSE349 contain 10 and 14 samples, respectively. If the remaining tumor was <25% or ≥25%, a breast cancer patient is considered to be sensitive or resistant, respectively, to docetaxel treatment. All the data were downloaded and processed according to the approach proposed by Geeleher *et al*. [54]**.**

***Data Pertaining to Multiple Myeloma*.** The training set $\mathbf{D} \in \mathbb{R}^{280 \times 9115}$ contains 280 cancer cell lines, 9,114 genes, and drug IC_{50} values that correspond to a 280-dimensional column vector. The test set $\mathbf{T} \in \mathbb{R}^{188 \times 9114}$ is composed of 188 multiple myeloma patients and 9,114 genes. The drug IC_{50} values for bortezomib [151, 152] were downloaded from (http://genemed.uchicago.edu/~pgeeleher/cgpPrediction/), and the data for the cancer cell lines were downloaded from the ArrayExpress repository (accession number is E-MTAB-783 or available at https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-783/?query=EMTAB783).
The clinical trial data corresponding to the test set were downloaded from the Gene Expression Omnibus (GEO) repository (http://www.ncbi.nlm.nih.gov/geo/) with accession

number GSE9782 [153]. The data were downloaded, processed and mapped according to Geeleher *et al*. [54].

Data pertaining to non-small cell lung cancer and triple-negative breast cancer. The training sets correspond to an 258×9508 matrix and an 497×9621 matrix for non-small cell lung cancer and triple-negative breast cancer, respectively. The test sets correspond to an 25×9507 matrix (excluding labels) and an 24×9620 matrix (excluding labels) for non-small cell lung cancer and triple-negative breast cancer, respectively. The data were downloaded from (http://genemed.uchicago.edu/~pgeeleher/cgpPrediction/) [54].

7.4.2 Experimental Methodology

Kernel-based methods, such as SVM and support vector regression (SVR), are popular machine learning algorithms and exhibit state-of-art performance in many applications [154, 155], including biological fields [156]. Therefore, in the experiments, SVR with linear kernel (SVR+L) and sigmoid kernel (SVR+S) are used as machine learning algorithms, coupled with the proposed link prediction algorithms (A1 or A2). The proposed link prediction algorithms are also employed with linear ridge regression (RR). In total, 9 drug sensitivity prediction algorithms are considered, as summarized in Table 7.1.

Each prediction algorithm was trained on the same training set, whose labels are continuous to yield models (see Methods section). Then, each model is applied to the same test set to yield predictions, as discussed in Methods section. The test set consists of the clinical trial expression data of patients, including baseline tumor expression data from primary tumor biopsies prior to treatment with an anticancer drug. The responses (i.e., labels) of the test set are categorical (e.g., either “sensitive” or “resistant”). These labels were clinically evaluated by the degree of reduction in tumor size to the given drug [54].

To evaluate whether the proposed approach exhibits stable superior performance as the sample size changes, the sample size for the training set was gradually reduced by 1% to 4% in each run. That is, 5 runs with sample sizes of 482, 478, 473, 468, and 463 and 280, 278, 275, 272, and 269 were performed for the two datasets, respectively.

The accuracy of the prediction algorithms is measured using the *Area Under the ROC Curve* (AUC), as shown in [54]. The higher is the AUC of an algorithm is, the better the performance that it achieved is. The mean of the AUC values averaged over the five runs of the test set is denoted as the MAUC. A run of the test set is defined as predictions of a learned model on the test set, such that the model is learned from the training set. The size of this training set is varied to assess the stability of prediction algorithms, in which a stable prediction algorithm is one for which the prediction accuracy on the test set does not change dramatically due to small changes in the size of the training set [136, 137]. This type of assessment is important in biological systems, in which the best prediction algorithm outperforms other algorithms many times in the conducted experiments. Statistical significance is measured between all pairs of the prediction algorithms.

The software employed in this study included support vector regressions with linear and sigmoid kernels in the LIBSVM package [80], ridge regression [54], gene selection using CUR and topLeverage functions in the rCUR package [150], and R code for processing the datasets and performance evaluation [54]. R is used to write the code for the link prediction algorithms and perform the experiments.

7.4.3 Experimental Results

Tables 7.2 and 7.3 show the AUC of 9 docetaxel and bortezomib, respectively, sensitivity prediction algorithms on clinical breast cancer or multiple myeloma trial data. For each variation in training set size the prediction algorithm with the best performance (i.e., the highest AUC) on the clinical trial data is shown in bold.

Table 7.2 shows that the proposed prediction algorithms perform better than the baseline prediction algorithms (i.e., B+SVR+L and B+SVR+S) including B+RR, which is a prediction algorithm proposed by Geeleher *et al*. Row “m” and “d”, shows the number of cell lines or genes, respectively, in the training set that were provided to each prediction algorithm. The same training set was provided to each prediction algorithm. Rows “m+A1” and “m+A2”, or “d+A1” and “d+A2” show the number of selected cell lines or genes, respectively, that were used in the prediction algorithms that employed the proposed approach for learning the models. The results of the proposed prediction algorithms are dominant compared with the baseline prediction algorithms that employ clinical trial data of breast cancer in terms of the AUC of four runs and the MAUC. In contrast to the baseline prediction algorithms, the performance of the proposed prediction algorithms on the test set outperforms in terms of the AUC when the training set size is reduced.

Table 7.3 shows that the proposed prediction algorithms perform better than the baseline prediction algorithms (i.e., B+SVR+L and B+SVR+S) and B+RR, which is a prediction algorithm proposed by Geeleher *et al*. Row “m” or “d”, respectively, shows the number of cell lines or genes, respectively, in the training set that were provided to each prediction algorithm. The same training set is provided to each prediction algorithm. Rows “m+A1” and “m+A2” or “d+A1” and “d+A2” show the number of selected cell lines or genes, respectively, used in the prediction algorithms that employ the proposed approach for learning the models. The results of the proposed prediction algorithms are dominant compared with the baseline prediction algorithms on the clinical breast cancer trial data in terms of the AUC of each run and the MAUC. In particular, A2+RR achieves the highest mean AUC (MAUC) of 0.693 and performed the best in all runs. In contrast to the baseline prediction algorithms, the performance of A2+RR on the test results in the best AUC as the training set size is reduced, which indicates that A2+RR has a stable performance.

Table 7.2 AUC Scores of Docetaxel Sensitivity Prediction Algorithms in Breast Cancer Patients on the Test Set. The Algorithm with the Highest AUC Is Shown in Bold. MAUC = Mean AUC

M	482	478	473	468	463	MAUC
D	6538	6538	6538	6538	6538	-
A1+SVR+L	0.878	0.864	**0.871**	0.857	0.871	0.868
A1+SVR+S	0.871	0.857	0.814	0.828	**0.878**	0.849
A1+RR	0.850	0.828	0.821	0.850	0.842	0.838
m+A1	246	244	242	239	237	-
d+A1	13076	13076	13076	13076	13076	-
A2+SVR+L	**0.892**	0.857	0.864	**0.864**	0.864	0.868
A2+SVR+S	0.871	0.850	0.814	0.814	**0.878**	0.845
A2+RR	0.857	0.842	0.835	0.835	0.835	0.841
m+A2	246	244	242	239	237	-
d+A2	13000	13000	13000	13000	13000	-
B+SVR+L	0.835	0.814	0.800	0.821	0.835	0.821
B+SVR+S	0.842	**0.871**	0.864	0.857	0.857	0.858
B+RR	0.814	0.814	0.821	0.821	0.821	0.818

Table 7.3 AUC Scores of Bortezomib Sensitivity Prediction Algorithms in Multiple Myeloma Patients on the Test Set. The Algorithm with the Highest AUC Is Shown in Bold. MAUC = Mean AUC

m	280	278	275	272	269	MAUC
d	9114	9114	9114	9114	9114	-
A1+SVR+L	0.668	0.669	0.665	0.663	0.656	0.664
A1+SVR+S	0.638	0.623	0.637	0.642	0.662	0.640
A1+RR	0.685	0.673	0.679	0.677	0.690	0.681
m+A1	145	144	143	141	140	-
d+A1	18228	18228	18228	18228	18228	-
A2+SVR+L	0.678	0.678	0.671	0.668	0.654	0.670
A2+SVR+S	0.661	0.657	0.659	0.659	0.668	0.661
A2+RR	**0.686**	**0.689**	**0.696**	**0.695**	**0.699**	0.693
m+A2	145	144	143	141	140	-
d+A2	9114	9114	9114	9114	9114	-
B+SVR+L	0.613	0.609	0.622	0.628	0.632	0.621
B+SVR+S	0.602	0.600	0.601	0.605	0.598	0.601
B+RR	0.614	0.611	0.603	0.607	0.606	0.608

Table 7.4 shows the p-values of the two-tailed Wilcoxon signed rank test [76, 83] to measure the statistical significance between the prediction algorithms using clinical trial data of breast cancer and multiple myeloma patients. The p-values indicate that A1+SVR+L and A2+SVR+L prediction algorithms significantly outperformed the baseline prediction algorithms B+SVR+L, B+SVR+S, and B+RR. The remaining prediction algorithms that employ the proposed approach are not statistically different from B+SVR+S.

Table 7.4 P-Values of Wilcoxon Signed Rank Test (Two-Tailed) between All Pairs of Prediction Algorithms. Values with Statistical Significance ($p<0.05$) Are Shown in Bold

	A1+SVR+S	A1+RR	A2+SVR+L	A2+SVR+S	A2+RR	B+SVR+L	B+SVR+S	B+RR
A1+SVR+L	**0.0160**	0.5092	0.3077	0.0836	0.8807	**0.0051**	**0.0149**	**0.0051**
A1+SVR+S	-	0.1675	**0.0208**	0.1282	0.0929	**0.0051**	0.1830	**0.0080**
A1+RR	-	-	0.2846	0.5418	0.0672	**0.0051**	0.1388	**0.0076**
A2+SVR+L	-	-	-	0.0587	0.5754	**0.0051**	**0.0207**	**0.0047**
A2+SVR+S	-	-	-	-	0.1388	**0.0069**	0.0836	**0.0124**
A2+RR	-	-	-	-	-	**0.0076**	0.1675	**0.0051**
B+SVR+L	-	-	-	-	-	-	0.5754	0.1609
B+SVR+S	-	-	-	-	-	-	-	0.2040

Figures 7.3 and 7.4 show the ranking of all prediction algorithms from the highest to the lowest MAUC using clinical trial data pertaining to breast cancer and multiple myeloma patients, respectively. Each MAUC is calculated over the 5 runs of the clinical trial data. As shown in Figures 7.3 and 7.4, the proposed prediction algorithms outperform the baseline prediction algorithms [54] w.r.t the MAUC.

Figure 7.5 shows the predictions of three prediction algorithms on the test set (clinical data samples of 24 breast cancer patients) when the prediction algorithms were trained on a dataset with the size m=482 (i.e., the complete training set without any reductions).

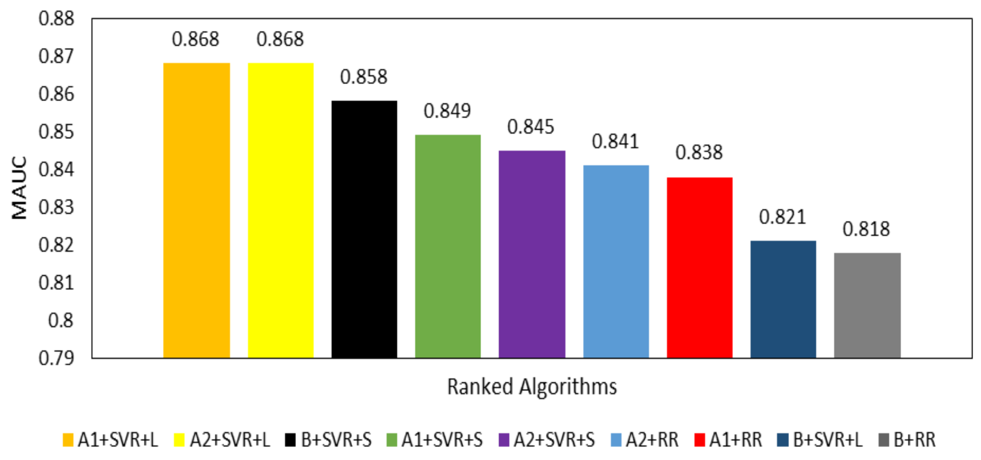


Figure 7.3 Mean AUC (MAUC) results of docetaxel sensitivity prediction algorithms in breast cancer patients ranked from the highest MAUC (left) to the lowest MAUC (right).

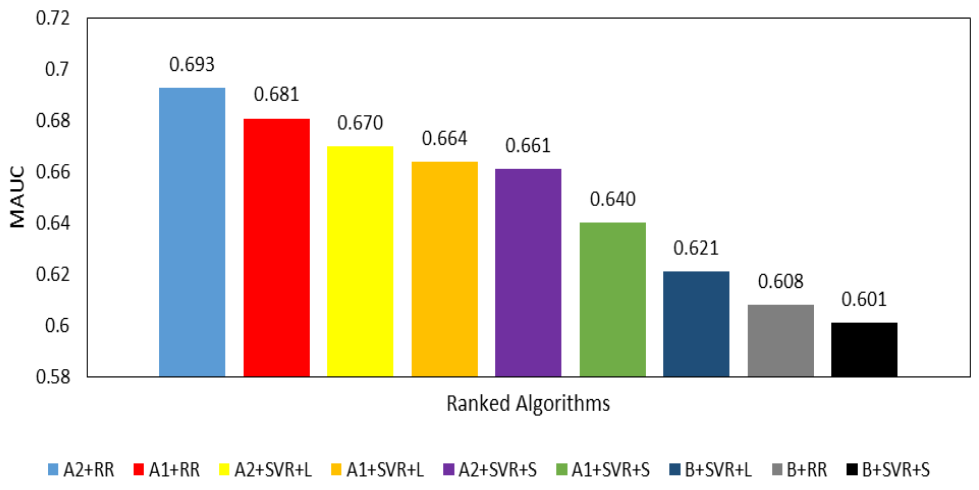


Figure 7.4 Mean AUC (MAUC) of bortezomib sensitivity prediction algorithms in multiple myeloma patients ranked from highest MAUC (left) to lowest MAUC (right).

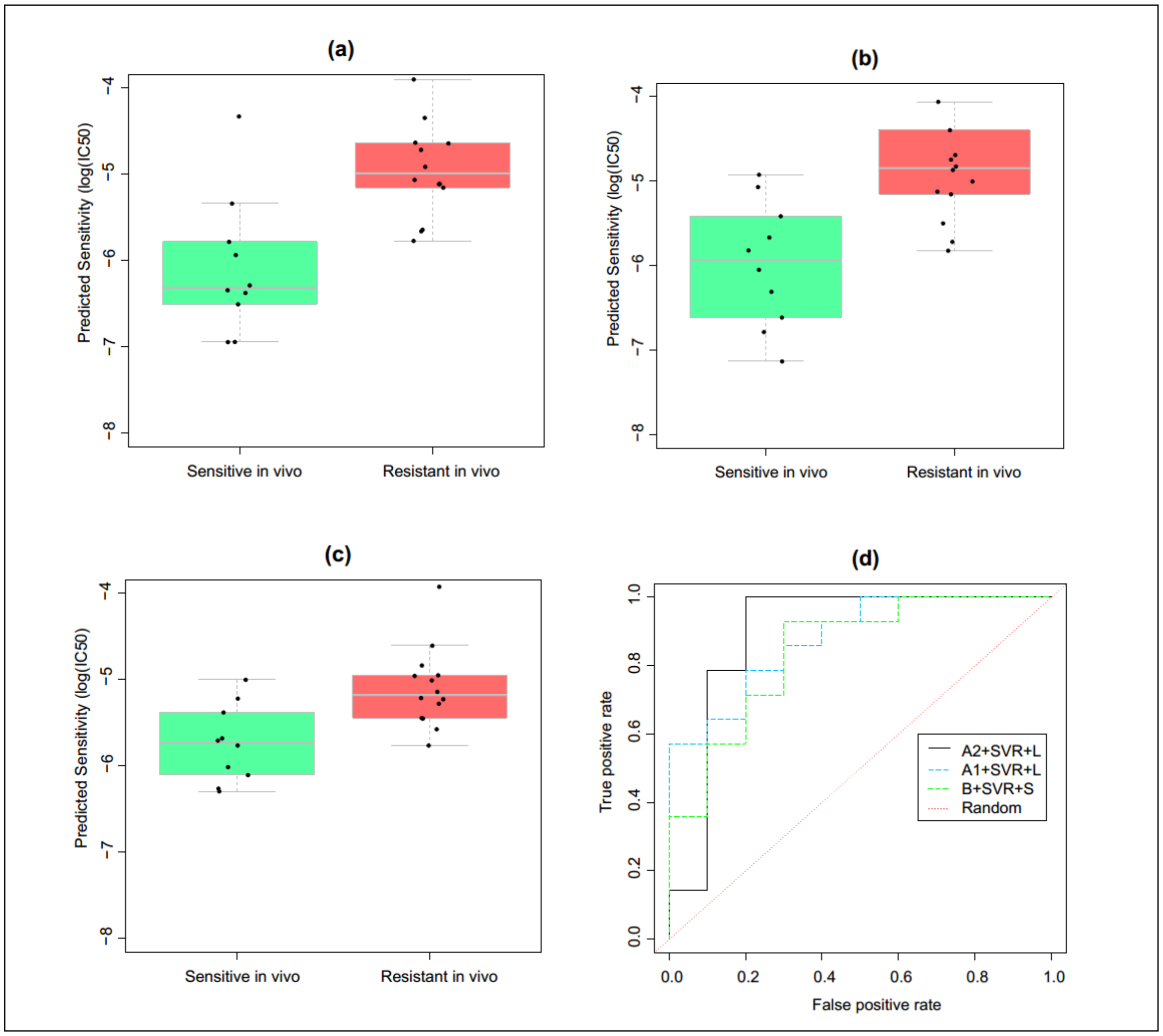
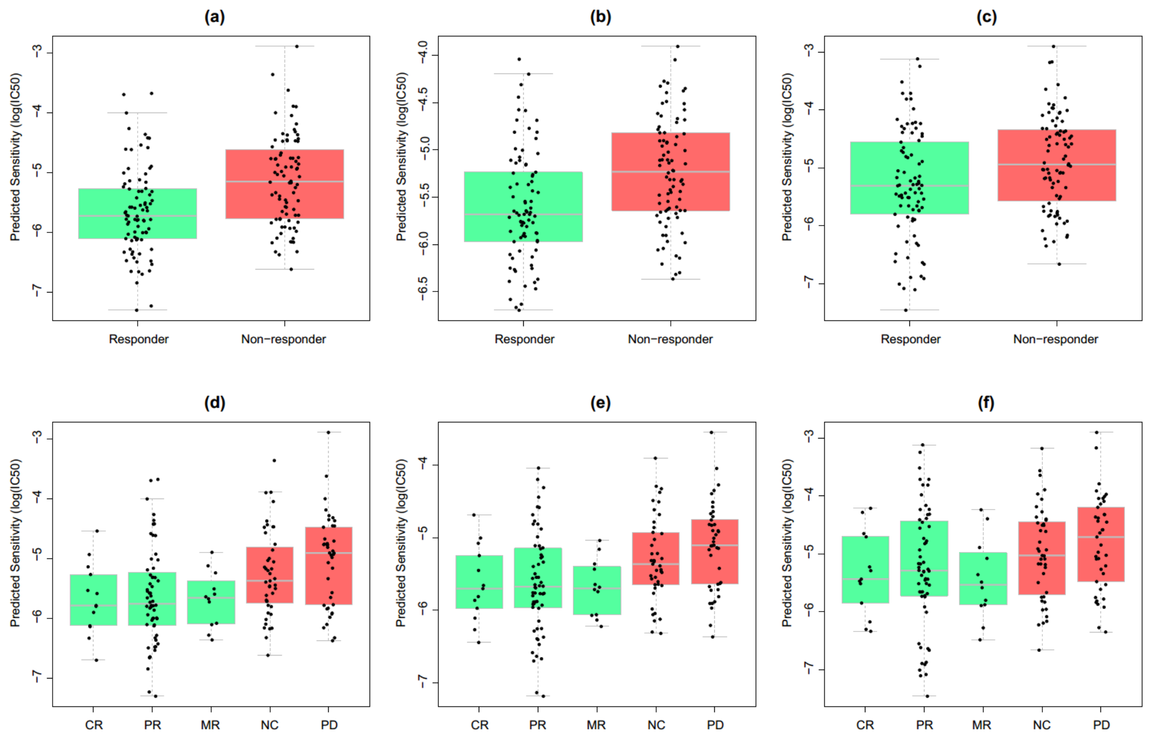


Figure 7.5 Prediction of docetaxel sensitivity in breast cancer patients. Strip charts and boxplots in (a), (b), and (c) show the differences in predicted drug sensitivity for individuals who are sensitive or resistant to docetaxel treatment using the prediction algorithms A2+SVR+L, A1+SVR+L and B+SVR+S, respectively, while (d) shows the ROC curves of prediction algorithms, revealing the proportion of true positives compared to the proportion of false positives. ROC = receiver operating characteristics.

Figure 7.5(a), Figure 7.5(b) and Figure 7.5(c) show the predictions of A2+SVR+L A1+SVR+L and B+SVR+S, respectively. For A2+SVR+L in Figure 7.5(a), the difference between the predicted drug sensitivity in breast cancer patients was highly statistically significant ($P = 472 \times 10^{-6}$ from the result of a *t*-test) between the trial-defined sensitive and resistant groups. The result of A1+SVR+L in Figure 7.5(b) was also highly statistically significant (*P*=614×10^{-6} from a *t*-test). B+SVR+S in Figure 7.5(c) achieved statistical significance (*P*=1176×10^{-6} from a *t*-test). Higher sensitivity or higher resistance, respectively, denote the greater or lesser effectiveness of the drug. In Figure 7.5(d), the ROC reveals AUC values of 0.892, 0.878 and 0.842 for A2+SVR+L, A1+SVR+L, and B+SVR+S, respectively, as shown in Table 7.2.

In Figure 7.6, the predictions of three prediction algorithms are reported on the test set (clinical trial data of 188 multiple myeloma samples of patients) when prediction algorithms learned models from a training set of size m=280 (i.e., the training set without any reductions). Figure 7.6(a) Figure 7.6(b) and Figure 7.6(c) show the predictions of the A2+RR, A1+RR, and B+RR, algorithms, respectively. For A2+RR (Figure 7.6(a)), the difference between the predicted drug sensitivity in multiple myeloma patients was highly significant (*P*=8×10^{-6} from a *t*-test) between trial-defined responder groups and non-responder groups. The result of A1+RR was also highly significant (*P*=11×10^{-6} from a *t*-test), while B+RR achieved statistically significant result (*P*=2612×10^{-6} from a *t*-test). Figure 7.6(d), Figure 7.6(e), and Figure 7.6(f)) break down the responders and non-responders of Figure 7.6(a), Figure 7.6(b), and Figure 7.6(c), respectively, to CR, PR, MR, NC or PD. In Figure 7.6 (g), The ROC reveals AUCs of 0.686, 0.685, and 0.614 for A2+RR, A1+RR, and B+RR, respectively, as shown in Table 7.3.



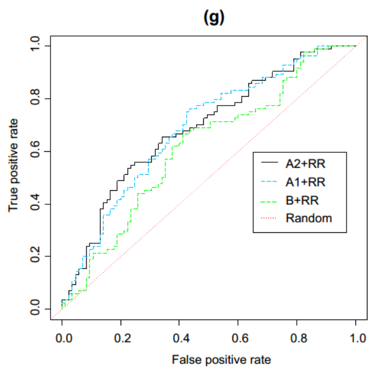


Figure 7.6 Prediction of bortezomib sensitivity in multiple myeloma patients. Strip charts and boxplots in (a), (b), and (c) show predicted drug sensitivity for in vivo responders and non-responders to bortezomib using A2+RR, A1+RR and B+RR prediction algorithms, respectively. Strip charts and boxplots d, e, and f further break down responders and non-responders of strip charts and boxplots a, b, and c as showing CR, PR, MR, NC or PD using A2+RR, A1+RR and B+RR, respectively, prediction algorithms. (g) ROC curves illustrating estimated prediction accuracy of prediction algorithms. CR, complete response; PR, partial response; MR, minimal response; NC, no change; PD, progressive

The performance of prediction algorithms is also evaluated on the clinical trial data pertaining to non-small cell lung cancer patients and the triple-negative breast cancer patients. Similar results are observed that the proposed prediction algorithms noticeably outperform the baseline prediction algorithms (See Appendix A: Table A.1 and Table A.2).

It is worth mentioning that the performance of other machine learning algorithms was also assessed, including random forests [121], support vector regression with a polynomial kernel of degree 2, and support vector regression with a Gaussian kernel. Moreover, other dimensionality reduction methods were applied such as principal component analysis (PCA) [157] based on the prcomp package in R [91], sparse PCA [158, 159], non-negative and sparse cumulative PCA, and negative and sparse PCA [160, 161]. However, they did not exhibit acceptable predictive performance; consequently, their results are not included in this paper.

7.5 Summary

In this chapter, a link prediction approach to cancer drug sensitivity prediction is introduced. The benefit of introducing a link prediction approach is to obtain satisfactory feature representation for better prediction performance. Two algorithms that employ the link prediction approach are proposed: (1) A supervised link prediction algorithm, which selects better quality training cancer cell lines using a modified version of QBC; and (2) An extended supervised link prediction, which selects both better training cancer cell lines and a subset of important genes using state of the art CUR matrix decomposition.

In this study, the link prediction algorithms use two machine learning algorithms: support vector regression and ridge regression. The experimental results demonstrate the stability of the proposed link prediction algorithms, which outperform drug sensitivity prediction algorithms of an existing approach as measured by their higher and statistically significant AUC scores.

Gene (feature) selection is important to the success of the proposed method. After many years of biomedical research, some signaling pathways have been known for being implicated in various cancers. It is tempted to exploit this pathway information for feature selection. For example, adding the signaling pathways might be considered as a constraint to get reliable feature sets. Consequently, the performance of the proposed prediction algorithms was assessed using only the genes in the signaling pathways that are known to the cancers. Inferior results were obtained (See Appendix B for details). It is noted that the current pathway information is limited. If those signaling genes are only considered, those important genes not identified yet by domain knowledge may be missed. This may hurt the overall performance as shown in this case. Therefore, a better strategy may be to include all genes but assign more weights to those signaling pathway genes.

CHAPTER 8

A NOISE-FILTERING APPROACH FOR CANCER DRUG SENSITIVITY PREDICTION

8.1 Introduction

Accurately predicting drug responses to cancer is an important problem hindering oncologists' efforts to find the most effective drugs to treat cancer, which is a core goal in precision medicine. The scientific community has focused on improving this prediction based on genomic, epigenomic, and proteomic datasets measured in human cancer cell lines. Real-world cancer cell lines contain noise, which degrades the performance of machine learning algorithms. This problem is rarely addressed in the existing approaches. In this chapter, a learning approach is proposed that removes the noisiest cell lines, allowing a model to be learned from better quality cell lines to improve the predictive performance. The proposed approach consists of three steps. First, a distance matrix is calculated that corresponds to all the inner products of the rows of a given matrix constructed using the Manhattan distance of the training input. Second, technique from linear algebra is adopted to project the training input on the eigenvectors of the distance matrix to yield transformed training input that corresponds to feature vectors with noise-filtered features. Third, information retrieval technique [162] is adopted to retrieve (i.e., select) a subset of better quality cell lines with the associated drug responses from the training set using the degrees between the training input cell lines and the corresponding transformed training input cell lines, where smaller degrees denote better quality cell lines. Then, a learning algorithm is applied to the better quality training set to induce (i.e., learn) a model used for prediction on the test set. The learning algorithms considered here are support vector regression and ridge regression [54, 98]. Other learning algorithms are excluded such as random forests [121] because of their poor performance, as in [54]. As the experimental results show, the proposed approach outperforms the baseline prediction algorithms proposed by Geeleher *et al*. [54].

8.2 The Proposed Approach

Figure 8.1 outlines the proposed approach, which works as follows. Consider the gene expression profiles denoted by $\mathrm{X} \in \mathbb{R}^{m \times n}$., which consists of m cell lines (i.e., samples) and n genes. $\mathrm{Y} = (y_1, ..., y_m)^T$ consists of the corresponding real-value drug responses to X. A training set is define as $\mathrm{S} = \{(\mathrm{x}_i, y_i)\}_{i=1}^{m}$, $\mathrm{x}_i \in \mathrm{X} \text{and} \, y_i \in \mathrm{Y}$. A test set is defined as $\mathrm{T} = \{\mathrm{x}_i^{'}\}_{i=1}^{p}$, where $\mathrm{x}_i^{'} \in \mathbb{R}^n$. (B) The distance matrix D is computed as

$$\mathrm{D} = \mathrm{L}\mathrm{L}^T \tag{8.1}$$

Where

$$\mathrm{L} = [l(\mathrm{x}_i, \mathrm{x}_j)]_{ij} \in \mathbb{R}^{m \times m} \tag{8.2}$$

and

$$l(\mathrm{x}_i, \mathrm{x}_j) = \| \mathrm{x}_i - \mathrm{x}_j \|_1, \forall i, j = 1, ..., m. \tag{8.3}$$

Note that $\mathrm{D} \in \mathbb{R}^{m \times m}$, $\mathrm{x}_i \, (\mathrm{x}_j,$respectively) is the ith (jth, respectively) feature vector of feature matrix S, and $\| \mathrm{x}_i - \mathrm{x}_j \|_1 = \sum_{a=1}^{n} | x_i^a - x_j^a |$ is the Manhattan distance between $\mathrm{x}_i \, \text{and} \, \mathrm{x}_j$. The eigenvalues of the distance matrix D are denoted by $\lambda_1 < \lambda_2 < ... < \lambda_m$, and $\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_m$ denote the corresponding eigenvectors. According to the Courant-Fischer Theorem [108], we then have

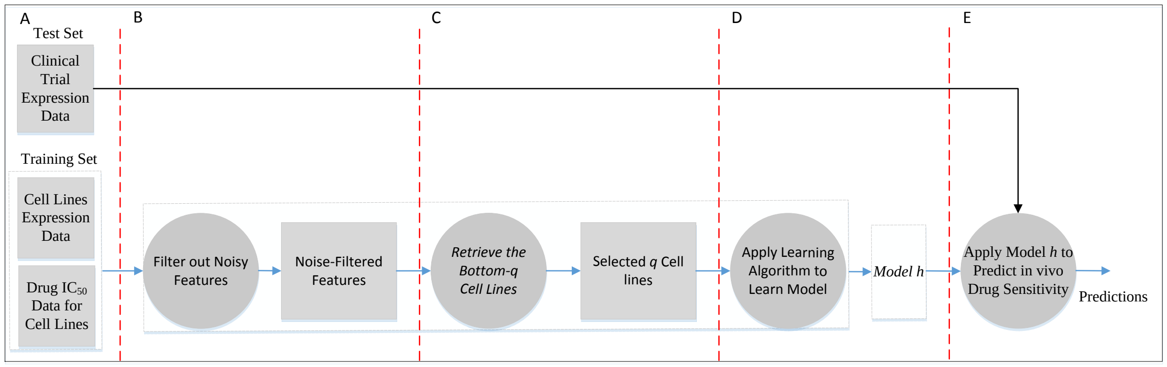

A
B
C
D
E
Test Set
Clinical Trial Expression Data
Training Set
Cell Lines Expression Data
Drug IC50 Data for Cell Lines
Filter out Noisy Features
Noise-Filtered Features
Retrieve the Bottom-q Cell Lines
Selected q Cell lines
Apply Learning Algorithm to Learn Model
Model h
Apply Model h to Predict in vivo Drug Sensitivity
Predictions

Figure 8.1 Data flow diagram showing the proposed approach to predicting in vivo drug sensitivity.

$$v_1 = \arg\min_{z:\|z\|_2=1} z^T Dz \tag{8.4}$$

and

$$v_l = \arg\min_{z:\|z\|_2=1,\, z\perp \mathrm{span}\{v_1,v_2,\ldots,v_{l-1}\}} z^T Dz. \tag{8.5}$$

Let $V_t = [v_1, v_2, \ldots, v_t]$ be the matrix whose columns are the first t eigenvectors of the distance matrix D with the smallest eigenvalues (in this study, $t = 1$.) The training input of cell lines X is projected onto V_t to obtain the transformed training input $\overline{X} \in \mathbb{R}^{m\times n}$ with noise-filtered features. That is,

$$\overline{X} = V_t V_t^T X. \tag{8.6}$$

(C) Compute the degrees between each training input $x_i \in X$ and the corresponding transformed training input $\overline{x}_i \in \overline{X}$ for $i = 1..m$

$$i^* = \arg\min_{i\in\{1,2,\ldots,m\}} \theta_i \tag{8.7}$$

and

$$\theta_i = \cos^{-1}\left(\frac{\overline{\mathrm{x}}_i.\mathrm{x}_i}{\|\overline{\mathrm{x}}_i\|_2 \|\mathrm{x}_i\|_2}\right) \times \frac{180}{\pi} \quad . \tag{8.8}$$

In this case, i^* is the index of $\mathrm{x}_{i^*} \in \mathbf{X}$, whose degree is θ_{i^*}. Denote by $\overline{\mathrm{S}} = \{(x_{i^*}, y_{i^*})\}_{i^*=1}^{q} \subseteq \mathrm{S}$ the reduced training set whose q feature vectors correspond to the training set with the smallest q degrees (i.e., $\theta_{i^*=1} < \theta_{i^*=2} < ... < \theta_{i^*=q}$ where $q < m$).

(D) A learning algorithm is called on the training set $\overline{\mathrm{S}}$ of size q to induce model h.

(E) Model h is applied to make predictions on the test set. In the rest of this paper, the prediction algorithms that employ the proposed approach using the abbreviations will be referred to as PA+SVR+L, PA+SVR+S, and PA+RR (see Table 8.1).

Table 8.1 Abbreviation of Drug Sensitivity Prediction Algorithms

Abbreviation	**Prediction Algorithm**
PA+SVR+L	The proposed approach using support vector regression with a linear kernel
PA+SVR+S	The proposed approach using support vector regression with a sigmoid kernel
PA+RR	The proposed approach using ridge regression
B+SVR+L	The baseline approach using support vector regression with a linear kernel
B+SVR+S	The baseline approach using support vector regression with a sigmoid kernel
B+RR	The baseline approach using ridge regression

8.3 Experiments and Results

8.3.1 Datasets

The training sets correspond to an 482×6539 matrix and an 280×9115 matrix for breast cancer and multiple myeloma, respectively. The test sets correspond to an 24×6538 matrix (excluding labels) and an 188×9114 matrix (excluding labels) for breast cancer and multiple myeloma, respectively. Cell lines expression data for breast cancer and multiple myeloma were download from the ArrayExpress repository (accession number E-MTAB-783). The drug IC_{50} values for docetaxel and bortezomib were downloaded from (http://genemed.uchicago.edu/~pgeeleher/cgpPrediction/). The clinical trial data for breast cancer (multiple myeloma, respectively) were downloaded from the Gene Expression Omnibus (GEO) repository with accession numbers GSE350 and GSE349 (GSE9782, respectively) [132, 133]. The responses (i.e., labels) of the clinical trial data are categorical (e.g., "sensitive" or "resistant"). These labels were clinically evaluated by the degree of reduction in tumor size to the given drug [54]. The downloaded data were processed according to Geeleher *et al*. [54].

8.3.2 Experimental Methodology

10-fold cross validation is not suitable in this study as labels of the testing sets are categorical while the labels of the corresponding training sets are real values. Hence, to evaluate whether the proposed approach exhibits stable superior performances as sample size changes, the sample size for the training set was reduced by 1% for each run, until the reduction reached 4%. In other words, 5 runs with sample sizes of (482, 478, 473, 468, 463) and (280, 278, 275, 272, 269) were performed for the two real datasets, respectively.

Each prediction algorithm was trained on the same given training set, whose labels were continuous, to yield models. Then, each model was applied to the same test set to yield predictions, where the accuracy of the prediction algorithms were measured using the *Are Under the ROC Curve* (AUC) as in [54].

The software used in this work included support vector regression with linear and sigmoid kernels in the LIBSVM package [104], ridge regression [54], and R code for processing the datasets and performance evaluation [54]. R was used to write code for the prediction algorithms and to perform the experiments.

8.3.3 Experimental Results

Table 8.2 and Table 8.3 show the AUCs of 6 docetaxel and bortezomib sensitivity prediction algorithms on the clinical trial data for breast cancer and multiple myeloma, respectively. Columns "m" and "d" show the number of cell lines and genes, respectively, in the training sets that were provided to each prediction algorithm; the same training sets were provided to each prediction algorithm. Column "q+PA" shows the number of selected (i.e., retrieved) cell lines that were used in PA+SVR+L, PA+SVR+S, and PA+RR to learn the models.

Table 8.2 and Table 8.3 show that the proposed prediction algorithms perform better than the baseline prediction algorithms (i.e., B+SVR+L and B+SVR+S) and B+RR, the proposed prediction algorithm by Geeleher *et al* [54]. The results are dominant compared to the other prediction algorithms on the clinical trial data in terms of the AUC of each run and of MAUC. These results indicate the stability of the proposed prediction algorithms.

Table 8.2 The AUC of Docetaxel Sensitivity Prediction Algorithms for Breast Cancer Patients on the Test Set. The Algorithm with the Highest AUC Is Shown in Bold. MAUC Is the Mean AUC

m	d	PA+SVR+L	PA+SVR+S	PA+RR	q+PA	B+SVR+L	B+SVR+S	B+RR
482	6,538	0.842	**0.878**	0.821	478	0.835	0.842	0.814
478	6,538	0.807	**0.871**	0.814	474	0.814	**0.871**	0.814
473	6,538	0.814	**0.878**	0.821	469	0.800	0.864	0.821
468	6,538	0.828	**0.864**	0.828	464	0.821	0.857	0.821
463	6,538	0.857	**0.864**	0.821	459	0.835	0.857	0.821
MAUC	-	0.829	0.871	0.821	-	0.821	0.858	0.818

Table 8.3 The AUC of Bortezomib Sensitivity Prediction Algorithms for Multiple Myeloma Patients on the Test Set. The Algorithm with the Highest AUC Is Shown in Bold. The MAUC Is the Mean AUC

m	d	PA+SVR+L	PA+SVR+S	PA+RR	q+PA	B+SVR+L	B+SVR+S	B+RR
280	9,114	**0.659**	0.635	0.654	210	0.613	0.602	0.614
278	9,114	**0.656**	0.623	0.65	209	0.609	0.600	0.611
275	9,114	**0.679**	0.626	0.647	207	0.622	0.601	0.603
272	9,114	**0.685**	0.641	0.653	204	0.628	0.605	0.607
269	9,114	**0.681**	0.658	0.657	202	0.632	0.598	0.606
MAUC	-	0.672	0.636	0.652	-	0.62	0.601	0.608

Figures 8.2 and 8.3 show the predictions of three prediction algorithms on the test sets when the prediction algorithms learned models from the training sets with sizes m=482 and m=280, respectively. For PA+SVR+S in Figure 8.2(a), the difference between the predicted drug sensitivity in breast cancer patients was highly statistically significant ($P = 67 \times 10^{-5}$ from a t-test) between trial-defined sensitivity and resistant groups. The B+SVR+S (Figure 8.2(b)) and B+RR (Figure 8.2(c)) achieved significant results with $P = 117 \times 10^{-5}$ and $P = 434 \times 10^{-5}$, respectively, from t-tests. PA+SVR+L

(Figure 8.3(a)) achieved highly significant results ($P = 10 \times 10^{-5}$ from a t-test). The results of B+RR (Figure 8.3(b)) and B+SVR+L (Figure 8.3(c)) were also significant with $P = 261 \times 10^{-5}$ and $P = 556 \times 10^{-5}$, respectively, from t-tests.

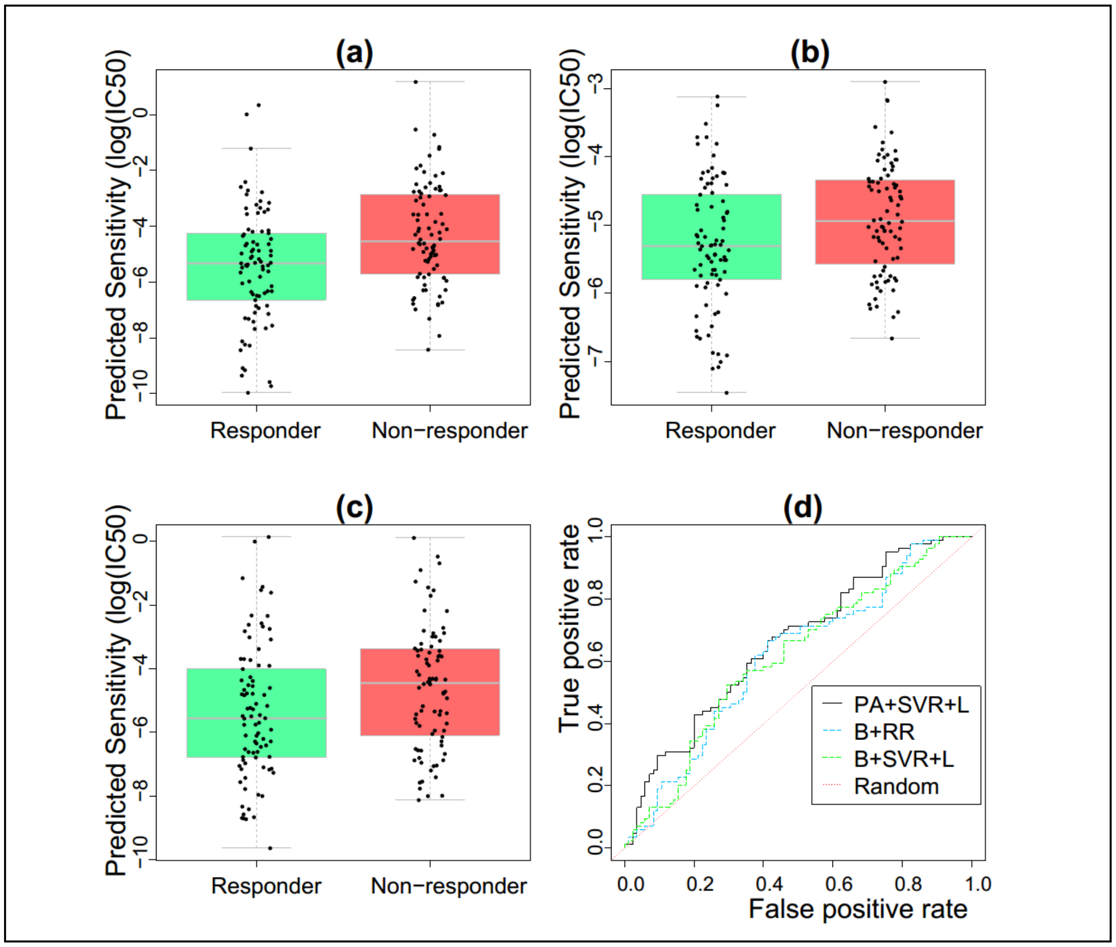


Figure 8.2 Predictions of docetaxel sensitivity in breast cancer patients. Strip charts (a), (b), and (c) show the differences in predicted drug sensitivity for individuals sensitive or resistant to docetaxel treatment using PA+SVR+S, B+SVR+S, and B+RR prediction algorithms, while (d) shows the ROC curves of the prediction algorithms, which reveal the proportion of true positives compared to the proportion of false positives. ROC = receiver operating characteristic.

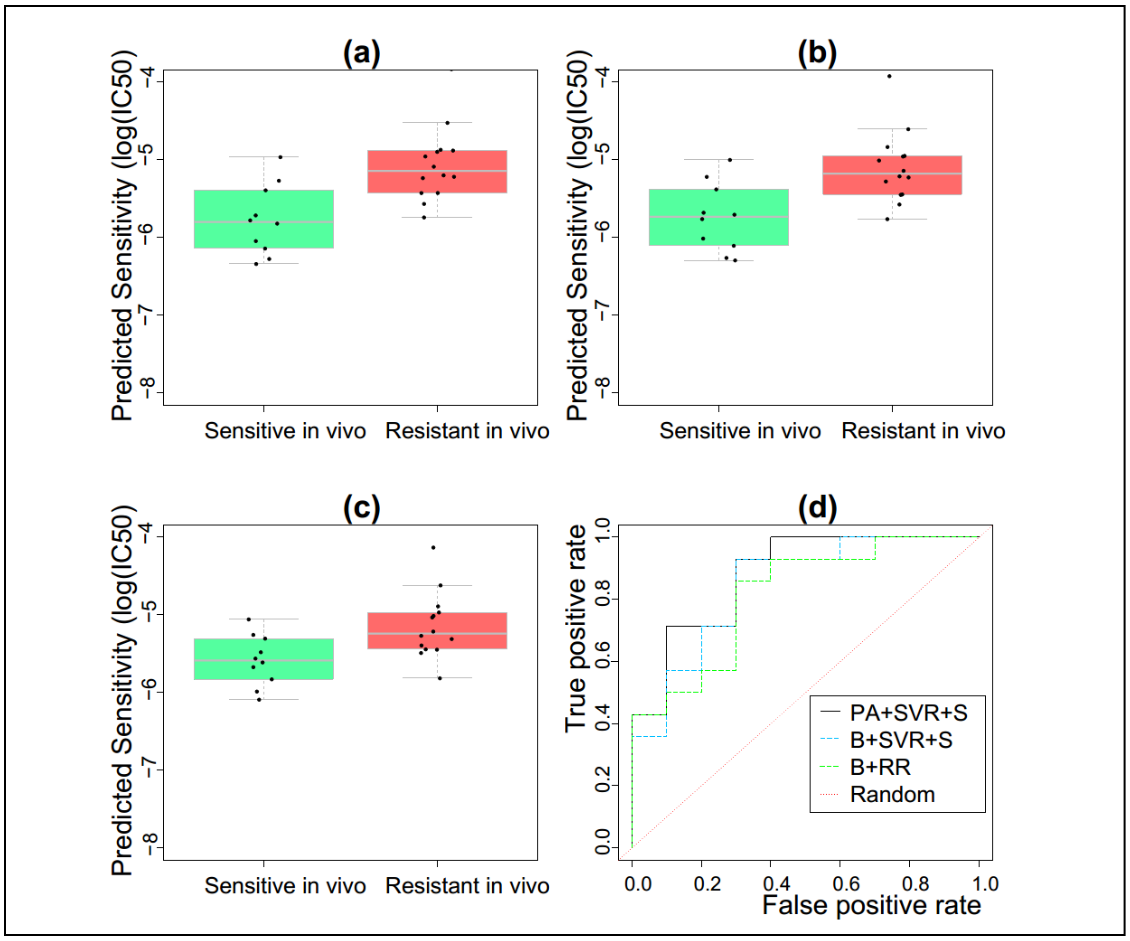


Figure 8.3 Prediction of bortezomib sensitivity in multiple myeloma patients. The strip charts and boxplots in (a), (b), and (c) show predicted drug sensitivity for in vivo responders and non-responders to bortezomib using the PA+SVR+L, B+RR, and B+SVR+L prediction algorithms, respectively. The ROC curve in (d) illustrates the estimated prediction accuracy of the prediction algorithms.

8.4 Summary

In this chapter, a noise-filtering approach is proposed that blends the following techniques: (1) Numerical linear algebra to yield transformed training input corresponding to noise-filtered features; (2) Information retrieval to retrieve better quality cancer cell lines from the training set according to the minimum degrees between the training input and the transformed training input; (3) Machine learning to learn a model from better quality, reduced-size training sets and perform predictions on test sets. The proposed approach uses two machine learning algorithms: support vector regression and ridge regression. Compared to an existing drug sensitivity approach, the proposed approach results in higher and statistically significant AUC values when using clinical trial data.

CHAPTER 9

TRANSFER LEARNING APPROACHES TO IMPROVE DRUG SENSITIVITY PREDICTION IN MULTIPLE MYELOMA PATIENTS

9.1 Introduction

Traditional machine learning approaches to drug sensitivity prediction assume that training data and test data must be in the same feature space and have the same underlying distribution. However, in real-world applications, this assumption does not hold. For example, we sometimes have limited training data for the task of drug sensitivity prediction in multiple myeloma patients (target task), but we have sufficient auxiliary data for the task of drug sensitivity prediction in patients with another cancer type (related task), where the auxiliary data for the related task are in a different feature space or come from a different distribution. In such cases, transfer learning, if applied correctly, would improve the performance of the prediction algorithms on the test data of the target task via leveraging the auxiliary data from the related task. In this chapter, transfer learning approaches are presented that combine the auxiliary data from the related task with the training data of the target task to improve the prediction performance on the test data of the target task. The performance of the proposed transfer learning approaches is evaluated exploiting three auxiliary datasets and compare them against baseline approaches using the area under the ROC curve (AUC) on test data of the target task. Experimental results demonstrate the good performance of the proposed approaches and their superiority over the baseline approaches when auxiliary data are incorporated.

In the sequel, the terms "sensitive" ("resistant", respectively) and "responder" ("non-responder", respectively) are used interchangeably. The terms "genes" and "features" are also used interchangeably throughout the paper.

9.2 Background

This section provides an introduction to the methods related to the proposed work: synthetic minority over-sampling technique (SMOTE) [123] and CUR matrix decomposition [149]. Each of them is introduced respectively after summarizing notations used in the paper.

9.2.1 Notations

To give a better understanding of the algorithms, the notations used in the paper are first summarized. Matrices are written as uppercase letters, e.g., matrix X. Vectors are denoted by lowercase letters, e.g., x. Vector elements are denoted by italic lowercase letters as scalars, e.g., y_i or x. A transpose of a matrix or a vector is indicated by T. So, for example, if x is a row vector, x^T is the corresponding column vector.

9.2.2 Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE [123] is a popular and a powerful over-sampling method that has shown a great deal of success in many applications [163-165]. Here, we are given a dataset $\mathrm{D}_+ \cup \mathrm{D}_-$. $\mathrm{D}_+ \in \mathbb{R}^{m \times d}$ contains examples from the minority class, $\mathrm{D}_- \in \mathbb{R}^{n \times d}$ contains examples from the majority class, and $m \ll n$. For each example $\mathrm{x}_i \in \mathrm{D}_+$, SMOTE finds the k-nearest neighbors $\mathrm{x}_i^1, \mathrm{x}_i^2, \ldots, \mathrm{x}_i^k$ of $\mathrm{x}_i \in \mathrm{D}_+$, where $\mathrm{x}_i^j \in \mathbb{R}^d$, $1 \le j \le k$, refers to the *j*th nearest neighbor of the *i*th example x_i in D_+. Then SMOTE generates synthetic examples $\mathrm{z}_i^1, \mathrm{z}_i^2, ..., \mathrm{z}_i^k$ along the line between each minority example $\mathrm{x}_i \in \mathrm{D}_+$ and its selected k nearest neighbors in the minority class as follows:

```
1. for $i$ = 1 to $m$
    1.1. for $j$ = 1 to $k$
        1.1.1. $\mathrm{z}_i^j = \mathrm{x}_i + (\mathrm{x}_i^j - \mathrm{x}_i)\lambda$
        1.1.2. Store [$\mathrm{z}_i^j$,+] in $\mathrm{D}_{++}$
    1.2. end for
2. end for
```

where $\mathrm{z}_i^j \in \mathbb{R}^d$ refers to the *j*th synthetic example generated from the *i*th example $\mathrm{x}_i \in \mathrm{D}_+, \lambda \in (0,1)$ is a random number, and the + sign indicates that synthetic examples are labeled with the minority class label. A random subset $\mathrm{D}_{++}^{'} \subseteq \mathrm{D}_{++}$ is then selected, where $\mathrm{D}_{++}^{'}$ consists of *n*-*m* synthetic examples. A learning algorithm could be called on the balanced dataset $\mathrm{D}_{++}^{'} \cup \mathrm{D}_+ \cup \mathrm{D}_-$, to induce a model and perform predictions on a given test set.

9.2.3 CUR Matrix Decomposition

Suppose that a dataset $\mathrm{F} \in \mathbb{R}^{m \times p}$ is given. Mahoney *et al*. [149] proposed CUR matrix decomposition as a dimensionality reduction paradigm that aims to obtain a low rank approximation of matrix F, which is expressed in terms of the actual rows and columns of the original matrix F:

$$\mathrm{F} \approx \mathrm{CUR}, \tag{9.1}$$

where C consists of a small number of the actual columns of F, R consists of a small number of the actual rows of F, and U is a constructed matrix that guarantees that CUR is close to F. Let v_j^ξ be the *j*th element of the ξth right singular vector of F. Let *l* be the rank of F. Then the normalized statistical leverage scores equal:

$$\pi_j = \frac{1}{l}\sum_{\xi=1}^{l}(v_j^\xi)^2 \tag{9.2}$$

for all $j = 1..p$, and $\sum_{j=1}^{p}\pi_j = 1$. C, U, and R matrices are constructed after calling COLUMNSELECT algorithm of Mahoney *et al.*, which takes an input matrix F, a rank parameter l, and an error parameter ε, and then performs the following steps:

1. Compute $\mathrm{v}^1, \mathrm{v}^2, ..., \mathrm{v}^l$ (the top l right singular vectors of F) and the normalized statistical leverage scores of Eq. 9.2.
2. Keep the jth column of F with probability $p_j = \min\{1, c\pi_j\}$ for all $j = 1..p$, where $c = O(l \log l / \varepsilon^2)$.
3. Return the matrix C consisting of the selected columns of F.

In step 1, the singular value decomposition SVD of F is computed, which decomposes F to $\mathrm{U\Sigma V}^T$, where $\mathrm{U} \in \mathbb{R}^{m\times l}$ is orthogonal matrix containing the top l left singular vectors of F, $\Sigma \in \mathbb{R}^{l\times l}$ is diagonal matrix containing singular values of F, $\mathrm{V}^T \in \mathbb{R}^{l\times p}$ is orthogonal matrix containing the top l right singular vectors of F, and l is the rank of F. The columns of U are pairwise orthogonal and normal (i.e., orthonormal), but its rows are not orthonormal as Euclidean norm is between 0 and 1. The rows of V^T are pairwise orthogonal and normal (i.e., orthonormal), but its columns are not orthonormal as

Euclidean norm is between 0 and 1 [166]. The other matrices (i.e., R, and U) are constructed as follows:

1. Run COLUMNSELECT on F^T with $c = O(l \log l / \varepsilon^2)$ to choose rows of F (columns of F^T) and construct matrix R.
2. Matrix U is defined as $\mathrm{U}=\mathrm{C}^{+}\mathrm{F}\mathrm{R}^{+}$, where C^{+} and R^{+} denote the Moore–Penrose generalized inverse of the matrix C and R, respectively.

Statistical leverage scores have been successfully employed in data analysis to identify the most influential genes and outlier detection [149]. A high statistical leverage score for a given gene indicates that the gene is regarded as an important (i.e., influential) gene. A low statistical leverage score for a given gene indicates that the gene is regarded as an unimportant gene. To select the q most important genes from matrix F, where $q < p$, we find the highest q statistical leverage scores used in computing matrix C of F, which correspond to the q most influential (i.e., important) genes.

9.3 Proposed Approaches

9.3.1 The Transfer Learning Approach

Figure 9.1 illustrates the proposed transfer learning approach, which works as follows. Suppose that we are given a target training set $\mathrm{F} = \{ (\mathrm{x}_1, y_1), ..., (\mathrm{x}_m, y_m)\}$ and a target test set $\mathrm{T} = \{ \mathrm{t}_1, ..., \mathrm{t}_r \}$. In the target training set, $\mathrm{x}_i \in \mathbb{R}^p$ is the ith target training example with p genes (i.e., features), $y_i \in \mathbb{R}$ is the corresponding label of x_i, and $\mathrm{t}_i \in \mathbb{R}^p$ is the ith target testing example of p genes. The target training set and target test set are disjoint, where m and r are the numbers of training and testing examples in the target task, respectively. In addition, we also have an auxiliary dataset $\mathrm{S} = \{(\mathrm{s}_1, u_1), \ldots, (\mathrm{s}_l, u_l)\}$, where $\mathrm{s}_i \in \mathbb{R}^n$ is the *i*th example (i.e., cell line of a cancer type) with n genes, $u_i \in \mathbb{R}$ is the corresponding label of s_i, and n, the number of genes in the auxiliary data, is different from p, the number of genes used in the target task. Our goal is to improve the prediction performance on the target test set T of the target task (prediction of bortezomib sensitivity in multiple myeloma patients) via learning an accurate model using the auxiliary dataset S and the target training set F. We summarize the problem definition in Table 9.1.

To incorporate the auxiliary data with the target training set, the following steps are performed.

(i) If the number of genes p in the target training set F is greater than the number of genes n in the auxiliary data S, then gene (i.e., feature) selection is performed on F as explained in step (ii). Otherwise, gene selection is performed on S. Assume without loss of generality that $p > n$.

(ii) q genes are selected from F based on their importance scores as defined in Equation (9.2), which depends on computing matrix C of F and the input rank parameter l (in this study, $q = n$ and the default parameter values are used for l, c, and ε in CUR function [150].) Then, the indexes of the highest q leverage scores are stored in I (where $q < p$); these correspond to the positions of the q most influential genes in matrix F. q genes are selected from the target training set F using their positions in I and stored in $\mathrm{F}' = \{(\mathrm{x}_1', y_1'), \ldots, (\mathrm{x}_m', y_m')\}$.

Table 9.1 Problem Formulation

Learning objective	Make predictions on the target test set of the target task
Target task	Prediction of bortezomib sensitivity in multiple myeloma patients
Related task	Prediction of specific drug sensitivity in patients with a cancer type
Target task data	Target Training set: $F = \{ (x_i, y_i) \}_{i=1}^{m}$ Target Test set: $T = \{ t_i \}_{i=1}^{r}$
Related task data	Auxiliary Data: $S = \{ (s_i, u_i) \}_{i=1}^{l}$ Cancer Types: Breast Cancer, Triple-Negative Breast Cancer, and Non-Small Cell Lung Cancer

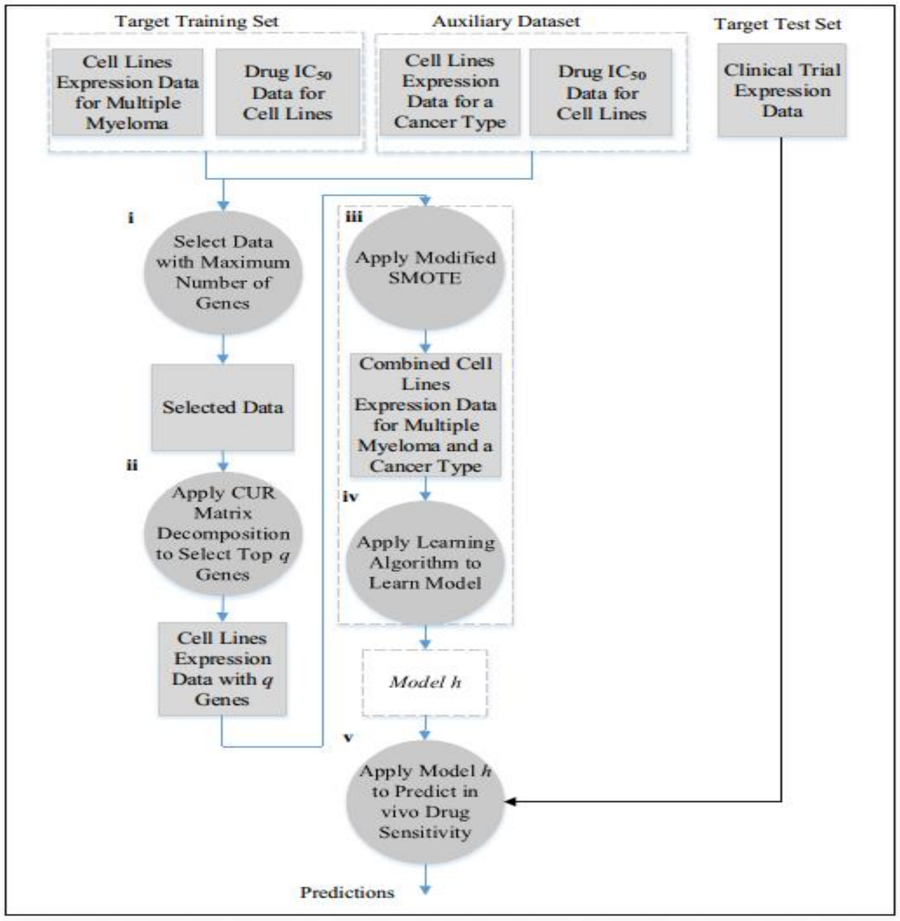

Target Training Set
Cell Lines Expression Data for Multiple Myeloma
Drug IC50 Data for Cell Lines
Auxiliary Dataset
Cell Lines Expression Data for a Cancer Type
Drug IC50 Data for Cell Lines
Target Test Set
Clinical Trial Expression Data
i
Select Data with Maximum Number of Genes
Selected Data
ii
Apply CUR Matrix Decomposition to Select Top q Genes
Cell Lines Expression Data with q Genes
iii
Apply Modified SMOTE
Combined Cell Lines Expression Data for Multiple Myeloma and a Cancer Type
iv
Apply Learning Algorithm to Learn Model
Model h
v
Apply Model h to Predict in vivo Drug Sensitivity
Predictions

Figure 9.1 Flowchart of the proposed transfer learning approach to predicting in vivo drug sensitivity.

(iii) The following steps are based on a modified version of SMOTE [123] where each example in the auxiliary dataset gets a representation closer to the target training set F':

1. Select b examples from the auxiliary dataset S. (In the study presented here, $b = 100$.) For each example s_i, $1 \le i \le b$, selected from S, pick one of s_i's nearest target training examples from F', denoted x_i^*, such that the picked example is different from all the target training examples previously picked for s_j, $1 \le j \le i$. More precisely, suppose s_i's k nearest target training examples are among the target training examples previously picked for s_j, $1 \le j \le i$. Then x_i^* is s_i's (k + 1)th nearest target training example from F'. Let y_i' be the corresponding label of x_i^*.
2. Change the representation of the examples selected from S using the following lines of code:

```
2.1 for $i$ = 1 to $b$
    2.1.1    $\mathrm{s}_i^* = \mathrm{s}_i + (\mathrm{x}_i^* - \mathrm{s}_i)\lambda$
    2.1.2    Store $[\mathrm{s}_i^*, (y_i' - \alpha)]$ in $\mathrm{S}^+$
2.2 end for
```

where $\lambda = 0.99$, $\alpha = 0.01$, and S^+ contains the new representations of the auxiliary data. Let $\mathrm{D} = \mathrm{S}^+ \cup \mathrm{F}'$ contain the combined cell lines expression data, where $\mathrm{D} \in \mathbb{R}^{m' \times n}$, and $m' = m + b.$

(iv) A learning algorithm is called on D to induce a model h.

(v) The n genes in the target test set T are selected using their positions in I and stored in T'. The model h is applied to the target test set T' to perform predictions.

9.3.2 The Boosted Transfer Learning Approach

Figure 9.2 illustrates the proposed boosted transfer learning approach. Steps (i), (ii), and (iii) are the same as steps (i), (ii), and (iii) of the transfer learning approach. Steps (iv) and (v) work as follows.

(iv) A modification of AdaBoost [116-118, 167] is employed, which works as follows. Initially, each training example $(\mathrm{x}_i, y_i) \in \mathrm{D}$ is assigned a weight $w_i = 1$ for $i = 1, \ldots, m'$. The probability for selecting the *i*th training example (x_i, y_i) in the training set D is

$$p_i = \frac{w_i}{\sum_{i=1}^{m'} w_i} \qquad (9.3)$$

where $\sum_{i=1}^{m'} p_i = 1$. Select m' training examples (without replacement) from D, to form the training set D'. A learning algorithm is called on D' to learn a model h and perform predictions on D, where predictions are then stored in $\mathrm{y}' = (y_1', \ldots, y_{m'}')$. Select the n genes in the target test set T using positions in I, and store the selected data with n genes in T'. Apply the model h on the target test set T', to yield predictions, which are stored as the first row vector in a matrix G. Repeat the following j times. (In this study, $j = 6$.)

1. Update the weights: $w_i = (y_i - y_i')^2$. for $i = 1, \ldots, m'$.
2. Calculate probabilities $\mathrm{p} = (p_1, p_2, \ldots, p_{m'})$ of the training examples in D, where

   p_i, $1 \le i \le m'$, is as defined in Equation (3).
3. Calculate the median of probabilities $p_1, p_2, \ldots, p_{m'}$ and store the median in v.

4. Select training examples from D where the weight of each selected example must be greater than or equal to v. Store the selected training examples in D'. Let p^* contain the probabilities corresponding to the selected training examples.

5. Select m' training examples (with replacement) from D' according to the probabilities in p^* and store the selected training examples in D''. The higher probability a training example is associated with, the more likely this training example will be included in D''.

6. A learning algorithm is called on D'' to learn model h and perform predictions on D.

7. Store the predictions performed on D in q.

8. Let $\mathrm{y}' = \mathrm{y}' + \mathrm{q}$, which corresponds to the cumulative predictions on the training set D.

9. Apply the learned model h on the target test set T' and store the predictions as the (j +1)th row vector in G.

(v) Output the final predictions as

$$\mathrm{Q} = \mathrm{e}^T \mathrm{G} \tag{9.4}$$

where G is an $(j+1)\times r$ matrix of predictions on the target test set T', the *i*th row vector of G corresponds to the predictions made in the (i -1)th iteration in step (iv), $\mathrm{e} = (\frac{1}{j+1},...,\frac{1}{j+1})^T$ is a $(j+1)\times 1$ column vector, and Q is a $1\times r$ row vector where the *i*th element in Q is the mean of the values in the *i*th column of G.

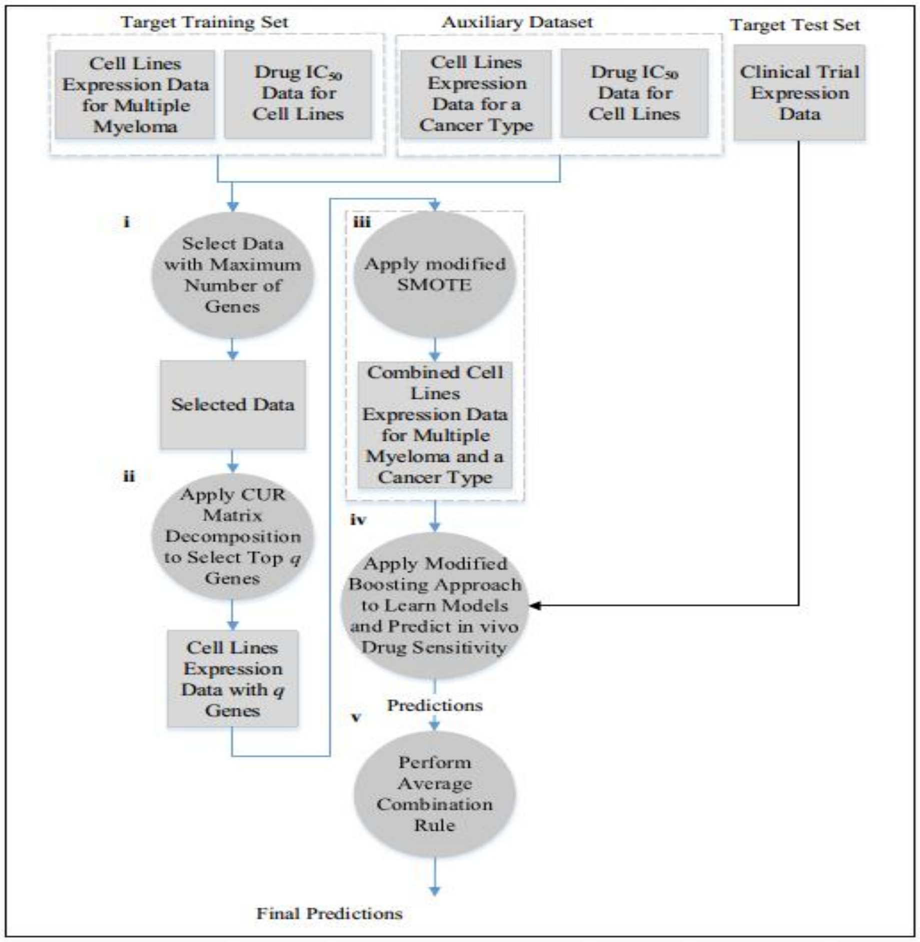

Target Training Set
Cell Lines Expression Data for Multiple Myeloma
Drug IC50 Data for Cell Lines
Auxiliary Dataset
Cell Lines Expression Data for a Cancer Type
Drug IC50 Data for Cell Lines
Target Test Set
Clinical Trial Expression Data
i
Select Data with Maximum Number of Genes
Selected Data
ii
Apply CUR Matrix Decomposition to Select Top q Genes
Cell Lines Expression Data with q Genes
iii
Apply modified SMOTE
Combined Cell Lines Expression Data for Multiple Myeloma and a Cancer Type
iv
Apply Modified Boosting Approach to Learn Models and Predict in vivo Drug Sensitivity
Predictions
v
Perform Average Combination Rule
Final Predictions

Figure 9.2 Flowchart of the proposed boosted transfer learning approach to predicting in vivo drug sensitivity.

9.4 Experiments

9.4.1 Datasets

Data Pertaining to Multiple Myeloma. The target training set $\mathrm{F} \in \mathbb{R}^{280 \times 9115}$ contains 280 target training examples (i.e., cancer cell lines), 9,114 genes, and drug IC_{50} values that correspond to a 280-dimensional column vector. The target test set $\mathrm{T} \in \mathbb{R}^{188 \times 9114}$ is composed of 188 samples of multiple myeloma patients and 9,114 genes. The drug IC_{50}

values for bortezomib [151, 152] were downloaded from (http://genemed.uchicago.edu/~pgeeleher/cgpPrediction/) [54], and the data for the cancer cell lines were downloaded from the ArrayExpress repository (accession number is E-MTAB-783 or available at https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-783/?query=EMTAB783) [131, 168, 169]. The clinical trial data corresponding to the test set were downloaded from the Gene Expression Omnibus (GEO) repository (http://www.ncbi.nlm.nih.gov/geo/) with accession number GSE9782. The data were downloaded, processed and mapped according to Geeleher *et al*. [54].

Data Pertaining to Breast Cancer. The auxiliary data correspond to an 482×6539 matrix (i.e., 482 examples and 6538 genes plus labels, i.e., drug IC_{50} values) for breast cancer patients. The drug IC_{50} values for docetaxel [130, 170]. (a chemotherapy drug) were downloaded from (http://genemed.uchicago.edu/~pgeeleher/cgpPrediction/) [54]. The cell line expression data were downloaded from the ArrayExpress repository (accession number is E-MTAB-783, also available at https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-783/?query=EMTAB783) [131, 168, 169]. All the data were downloaded and processed according to the approach proposed by Geeleher *et al*. [54].

Data Pertaining to Non-Small Cell Lung Cancer and Triple-Negative Breast Cancer. The auxiliary data correspond to an 258×9508 matrix (i.e., 258 examples and 9507 genes plus labels) and an 497×9621 matrix (i.e., 497 examples and 9620 genes plus labels) for non-small cell lung cancer and triple-negative breast cancer patients, respectively. The data were downloaded from (http://genemed.uchicago.edu/~pgeeleher/cgpPrediction/).

9.4.2 Evaluation and Baseline Approaches

The two proposed transfer learning approaches are compared with two different baseline approaches, which are described below.

First Baseline (B1). This baseline employs the proposed approach by Geeleher *et al*. [54].

Second Baseline (B2). In this baseline, CUR matrix decomposition is applied to F. Then, the indexes of the largest n statistical leverage scores of F are stored in I, as in the proposed approaches. The n genes from the target training examples in F are selected using positions in I. A learning algorithm is called on the auxiliary data with n genes combined with the target training set with n genes, to learn a model h. Then, the n genes in the target test set are selected using positions in I. The model h is applied to the target test examples with n genes, to yield drug sensitivity predictions. Thus, this baseline, which does not have a transfer learning mechanism (cf. steps (iii) in Section 9.3.1), differs from the proposed transfer learning approaches.

The proposed transfer learning approaches and the baseline approaches employ two machine learning algorithms, namely support vector regression (SVR) and ridge regression (RR). Table 9.2 summarizes the twelve prediction algorithms studied in this chapter.

Table 9.2 Summary of the Twelve Drug Sensitivity Prediction Algorithms in This Chapter

Abbreviation	Prediction Algorithm
T+SVR+L	The transfer learning approach using support vector regression with a linear kernel
T+SVR+S	The transfer learning approach using support vector regression with a sigmoid kernel
T+RR	The transfer learning approach using ridge regression
BT+SVR+L	The boosted transfer learning approach using support vector regression with a linear kernel
BT+SVR+S	The boosted transfer learning approach using support vector regression with a sigmoid kernel
BT+RR	The boosted transfer learning approach using ridge regression
B1+SVR+L	The first baseline approach using support vector regression with a linear kernel
B1+SVR+S	The first baseline approach using support vector regression with a sigmoid kernel
B1+RR	The first baseline approach using ridge regression
B2+SVR+L	The second baseline approach using support vector regression with a linear kernel
B2+SVR+S	The second baseline approach using support vector regression with a sigmoid kernel
B2+RR	The second baseline approach using ridge regression

Each prediction algorithm was trained on a training set, whose labels were continuous, to yield a model. Then, each model was applied to the target test set to yield predictions. The target test set consists of patients' clinical trial expression data, which are baseline tumor expression data from primary tumor biopsies before treatment with anticancer drugs. The labels of the target test set are not translated from continuous to categorical. Instead, the labels of the target test set are categorical (i.e., either “sensitive” or “resistant”), where these labels were clinically evaluated by the degree of reduction in tumor size to the given drug. A cancer patient is categorized as “sensitive” to cancer drug (e.g., docetaxel or bortezomib) treatment if the cancer patient exhibits less than 25% residual tumor. A cancer patient is categorized as “resistant” to cancer drug treatment if the cancer patient exhibits greater than or equal to 25% residual tumor [54].

Using in vitro drug sensitivity of the training data to predict in vivo drug sensitivity of the target test set is a challenging task and main goal in precision medicine, which corresponds to predicting the clinical outcome that is crucial for the life of the human being [171]. If the clinical drug response (i.e., clinical response to anticancer drug) is incorrectly predicted, the tumor size of a cancer patient would increase significantly over the time, which cause sequelae that lead to death. If the clinical drug response is correctly predicted, the tumor size would decrease significantly over the time and that would save the patient. By predicting clinical outcomes in the target test set correctly, clinicians would benefit from understanding the relationship between in vivo and in vitro drug sensitivity, which leads to better personalized treatment.

Ten-fold cross validation is not suitable in this study as labels of the target test set are categorical while the labels of the corresponding target training set are real numbers. Hence, to evaluate whether the proposed approach exhibits stable superior performances as sample size changes, the sample size for the target training set was randomly reduced by 1% for each run, until the reduction reached 4%. In other words, 5 runs were performed with sample sizes of 280, 278, 275, 272, 269, respectively.

The accuracy of the prediction algorithms was measured using the area under the ROC curve (AUC), as described in [54]. The higher the AUC score an algorithm achieves, the better its performance is. MAUC is used to denote the mean of the AUC values averaged over the five runs of the target test set. A run of the target test set is defined as the predictions of a learned model on the target test set in which the model was learned from a training set whose size is varied to assess the stability of the prediction algorithms. A stable prediction algorithm is the one whose prediction accuracy on the target test set does not change dramatically owing to small changes of the training set size [136, 137]. This type of assessment is important in biological systems, where the best prediction algorithm is the one that outperforms the other algorithms many times on conducted experiments. The statistical significance of the approaches was measured.

The software used in this work included support vector regression with linear and sigmoid kernels (with their default parameter values) in the LIBSVM package [104], ridge regression [54], gene selection using CUR and topLeverage functions in the rCUR package [150], and R code for processing the datasets and performance evaluation [54]. R is used to write code for the prediction algorithms and to perform the experiments.

9.4.3 Experimental Results

In this section, the performance of the proposed approaches is evaluated and compared to the baseline approaches. Each time the target training set of multiple myeloma is used with one of the auxiliary datasets pertaining to breast cancer, triple-negative breast cancer, and non-small cell lung cancer respectively to train the approaches (except the B1 approach that uses only target training set), to yield prediction algorithms and perform prediction on the target test set.

9.4.3.1 Exploiting Breast Cancer Auxiliary Data. Table 9.3 shows details of the target training set and auxiliary dataset pertaining to breast cancer used by each prediction algorithm. The target training set is obtained from the target task (i.e., prediction of bortezomib sensitivity in multiple myeloma patients) and the auxiliary data are acquired from the related task (i.e., prediction of docetaxel sensitivity in breast cancer patients). Rows "m / l" shows the number of examples or cell lines in the target training set/auxiliary dataset used in each run. Row "p / n" shows the number of genes or features in the target training set/auxiliary dataset used in each run. Row "$p \cap n$" shows the number of overlapped (i.e., intersected) genes between the target training set and the auxiliary data in each run. Rows "$P_{m/l}$" and "$P_{p/n}$" show the number of selected examples in the target training set/auxiliary dataset and the number of selected genes in the target training set/auxiliary dataset, respectively, that were used in the prediction algorithms employing the approaches during the training stage to learn models. Rows "$B1_m$" and "$B1_p$" show the number of selected examples and genes, respectively, in the target training set that were used during the training stage by the prediction algorithms employing the first baseline approach (B1). Row "$B2_{m/l}$" and "$B2_{p/n}$"Show the number of selected examples and genes, respectively, in the target training set/auxiliary dataset that were used by the prediction algorithms employing the second baseline approach (B2). In each run the target training set size is changed to train all machine learning algorithms employing the approaches as described in Sections 9.3 and 9.4.2, to yield models (i.e., prediction algorithms).

Table 9.3 Details of the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Breast Cancer, Respectively, Used by Each Prediction Algorithm

Run	1	2	3	4	5
m/l	280/482	278/482	275/482	272/482	269/482
p/n	9114/6538	9114/6538	9114/6538	9114/6538	9114/6538
$p \cap n$	5478	5478	5478	5478	5478
$P_{m/l}$	280/100	278/100	275/100	272/100	269/100
$P_{p/n}$	6538/6538	6538/6538	6538/6538	6538/6538	6538/6538
$B1_m$	280	278	275	272	269
$B1_p$	9114	9114	9114	9114	9114
$B2_{m/l}$	280/482	278/482	275/482	272/482	269/482
$B2_{p/n}$	6538/6538	6538/6538	6538/6538	6538/6538	6538/6538

Table 9.4 shows the AUCs of twelve prediction algorithms on the target test set of multiple myeloma patients. As shown in Table 9.4, BT+SVR+S performs better than the baseline prediction algorithms (i.e., B2+SVR+L, B2+SVR+S, B2+RR, B1+SVR+L, and B1+SVR+S and B+RR). In particular, BT+SVR+S achieves the highest AUC in 4 out of 5 runs. The BT+SVR+S results were consistently good compared to the other prediction algorithms in terms of AUC on the target test set as the target training set size is reduced. These results indicate that the performance of BT+SVR+S is stable.

Table 9.4 AUC Scores of the Twelve Prediction Algorithms on the Target Test Set of Multiple Myeloma Patients Where the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Breast Cancer Patients, Respectively, Are Used. The Algorithm with the Highest AUC Is Shown in Bold. Std is the Standard Deviation of the AUC Values Obtained from the Five Runs

Run	1	2	3	4	5
T+SVR+L	0.644	0.636	0.640	0.647	0.653
T+SVR+S	0.643	0.642	0.654	0.659	0.666
T+RR	0.653	0.655	0.652	0.657	0.652
BT+SVR+L	0.658	0.657	0.678	0.665	0.675
BT+SVR+S	**0.682**	0.682	**0.687**	**0.695**	**0.703**
BT+RR	0.670	**0.693**	0.668	0.681	0.659
B1+SVR+L	0.613	0.609	0.622	0.628	0.632
B1+SVR+S	0.602	0.600	0.601	0.605	0.598
B1+RR	0.614	0.611	0.603	0.607	0.606
B2+SVR+L	0.440	0.491	0.466	0.501	0.549
B2+SVR+S	0.449	0.485	0.487	0.506	0.500
B2+RR	0.499	0.505	0.511	0.511	0.516

Table 9.5 shows the P-values of a two-sample t-test on the target test set for each run, as in [54]. Each prediction algorithm with a highly statistically significant result is shown in red ($P < .001$). Each algorithm with a statistically significant result is shown in blue ($.001 \leq P < .05$). As shown in Table 9.5, the proposed prediction algorithms yield

highly statistically significant results, and these highly statistically significant results reflect the superior performance of the proposed prediction algorithms (see Table 9.4).

Table 9.5 P-Values of a Two-Sample *t*-Test for the Twelve Prediction Algorithms on the Target Test Set Where the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Breast Cancer Patients, Respectively, Are Used. Each Prediction Algorithm with $P < 0.001$ Is Considered As Highly Statistically Significant and Colored in Red. Each Prediction Algorithm with $P < 0.05$ Is Considered Statistically Significant and Colored in Blue

Run	1	2	3	4	5
T+SVR+L	6088×10^{-6}	906×10^{-6}	674×10^{-6}	442×10^{-6}	372×10^{-6}
T+SVR+S	765×10^{-6}	815×10^{-6}	498×10^{-6}	273×10^{-6}	219×10^{-6}
T+RR	188×10^{-6}	218×10^{-6}	244×10^{-6}	194×10^{-6}	313×10^{-6}
BT+SVR+L	218×10^{-6}	245×10^{-6}	2396×10^{-6}	132×10^{-6}	6888×10^{-8}
BT+SVR+S	4×10^{-5}	2172×10^{-8}	1506×10^{-8}	1378×10^{-8}	527×10^{-8}
BT+RR	3382×10^{-8}	9221×10^{-9}	4307×10^{-8}	1169×10^{-8}	6069×10^{-8}
B1+SVR+L	6×10^{-3}	7×10^{-3}	4×10^{-3}	2×10^{-3}	2×10^{-3}
B1+SVR+S	8×10^{-3}	1×10^{-2}	11×10^{-3}	9×10^{-3}	14×10^{-3}
B1+RR	261×10^{-5}	4×10^{-3}	6×10^{-3}	4×10^{-3}	5×10^{-3}
B2+SVR+L	881×10^{-3}	665×10^{-3}	489×10^{-3}	724×10^{-3}	613×10^{-3}
B2+SVR+S	784×10^{-3}	722×10^{-3}	708×10^{-3}	607×10^{-3}	633×10^{-3}
B2+RR	4992×10^{-4}	466×10^{-3}	398×10^{-3}	386×10^{-3}	342×10^{-3}

Figures 9.3(a), 9.3(b), and 9.3(c) show the predictions of BT+SVR+S, B1+RR, and B2+RR, respectively, on the target test set at the first run. The result of BT+SVR+S shown in Figure 9.3(a) was highly statistically significant ($P = 4\times10^{-5}$ from a two-sample t-test) between the trial-defined responder (i.e., sensitive) and non-responder (i.e., resistant) groups. The result of B1+RR shown in Figure 9.3(b) was statistically significant with $P = 261\times10^{-5}$ from a two-sample t-test. The result of B2+RR shown in Figure 9.3(c) was not statistically significant with $P = 49920\times10^{-5}$ from a two-sample

t-test. In Figure 9.3(d), the ROC curves reveal AUC values of 0.682, 0.614, and 0.499 for BT+SVR+S, B1+RR, and B2+RR, respectively.

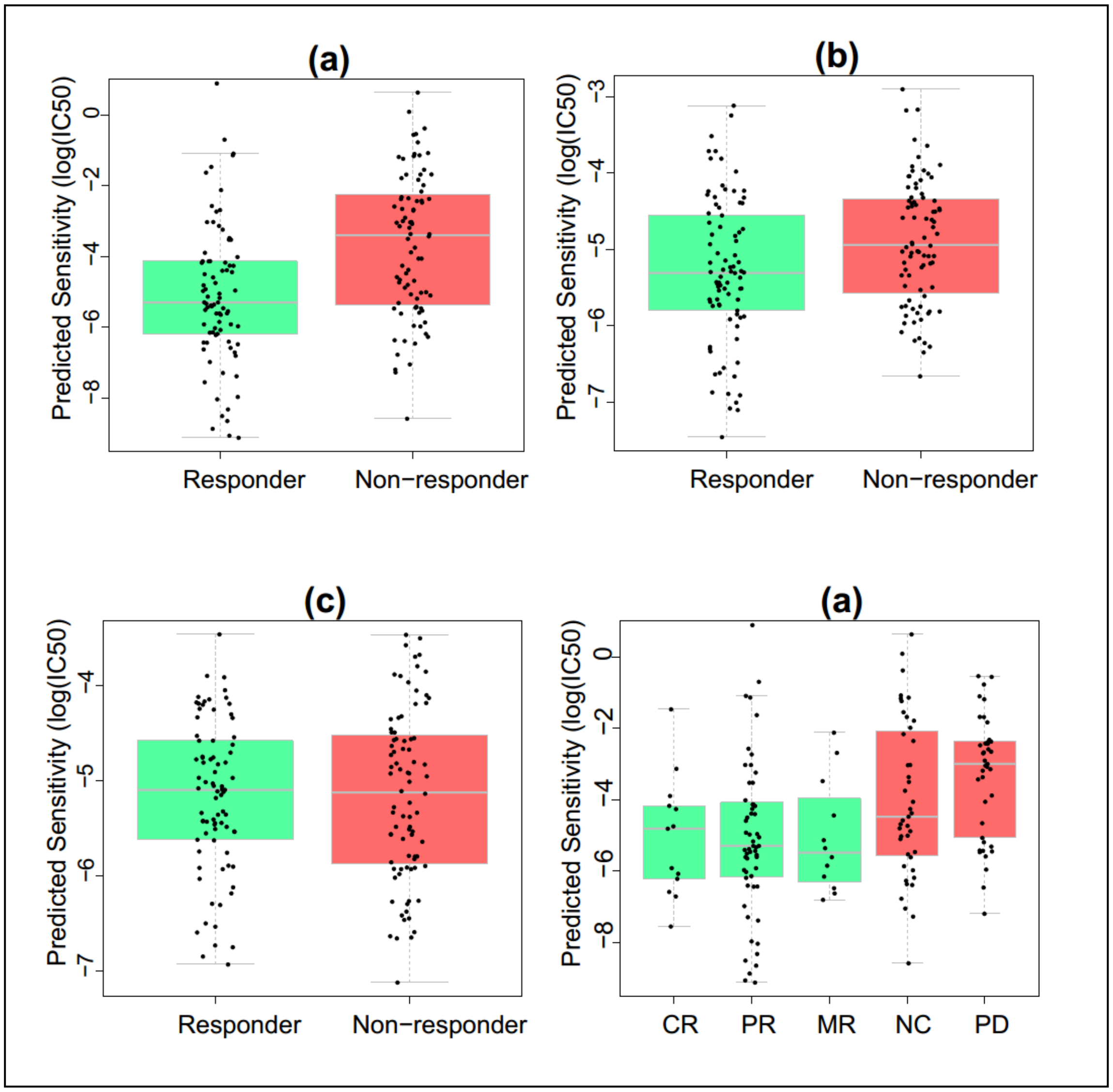


Figure 9.3 Predictions of bortezomib sensitivity on the target test set of multiple myeloma patients where the target training set and auxiliary dataset pertaining to multiple myeloma and breast cancer patients, respectively, are used. Strip charts and boxplots (a), (b), and (c) show the differences in predicted drug sensitivity to bortezomib treatment between the responder (i.e., sensitive) group and non-responder (i.e., resistant) group using BT+SVR+S, B1+RR, and B2+RR, respectively. (d) shows the ROC curves of the prediction algorithms, which reveal the proportion of true positives compared to the proportion of false positives. ROC = receiver operating characteristic.

Figure 9.4 shows the ranking of the twelve prediction algorithms based on their MAUC values. The MAUC of an algorithm is calculated by taking the mean of the AUC values the algorithm receives from the 5 runs of experiments. As shown in Figure 9.4, the proposed prediction algorithms outperform the baseline prediction algorithms w.r.t. the MAUC.

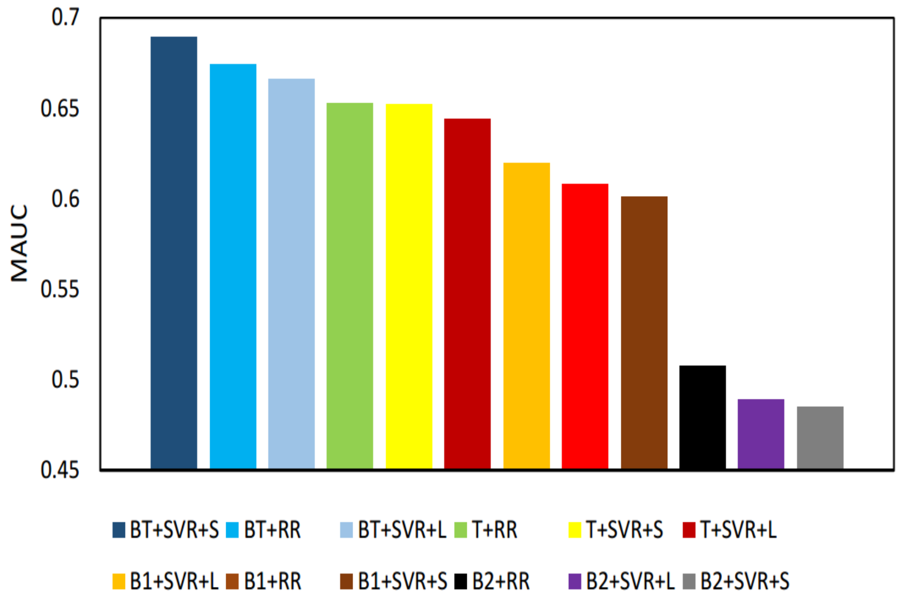


Figure 9.4 The Mean AUC (MAUC) values of the twelve bortezomib sensitivity prediction algorithms for multiple myeloma patients. The algorithms are ranked from left to right where the leftmost algorithm has the highest MAUC and the rightmost algorithm has the lowest MAUC.

9.4.3.2 Exploiting Triple-Negative Breast Cancer Auxiliary Data. Table 9.6 shows details of the target training set and the auxiliary dataset pertaining to triple-negative breast cancer patients used by each prediction algorithm. The target training set is obtained from the target task (i.e., prediction of bortezomib sensitivity in multiple myeloma patients) and the auxiliary dataset is obtained from the related task (i.e., prediction of cisplatin sensitivity in triple-negative breast cancer patients). The only difference between Table 9.3 and Table 9.6 is that Table 9.6 has different auxiliary data pertaining to triple-negative breast cancer patients, while the target test set is still the same.

Table 9.6 Details of the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Triple-Negative Breast Cancer Patients, Respectively, Used by Each Prediction Algorithm

Run	1	2	3	4	5
m/l	280/497	278/497	275/497	272/497	269/497
p/n	9114/9620	9114/9620	9114/9620	9114/9620	9114/9620
$p \cap n$	7911	7911	7911	7911	7911
$P_{m/l}$	280/100	278/100	275/100	272/100	269/100
$P_{p/n}$	9114/9114	9114/9114	9114/9114	9114/9114	9114/9114
$B1_m$	280	278	275	272	269
$B1_p$	9114	9114	9114	9114	9114
$B2_{m/l}$	280/497	278/497	275/497	272/497	269/497
$B2_{p/n}$	9114/9114	9114/9114	9114/9114	9114/9114	9114/9114

Table 9.7 shows the AUCs of the twelve prediction algorithms on the target test set of multiple myeloma patients. As shown in Table 9.7, the proposed prediction algorithms employing the boosted transfer learning approach (BT) perform better than the baseline prediction algorithms. Specifically, BT+SVR+S and BT+RR yielded the highest AUC in 4 out of 5 runs. These results indicate that the proposed prediction algorithms employing BT approach achieve high performance in terms of AUC on the target test set. The results also show the stability of the prediction algorithms employing BT approach.

Table 9.7 AUC Scores of the Twelve Prediction Algorithms on the Target Test Set of Multiple Myeloma Patients Where the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Triple-Negative Breast Cancer Patients, Respectively, Are Used. The Algorithm with the Highest AUC Is Shown in Bold. Std Is the Standard Deviation of the AUC Values Obtained from the Five Runs.

Run	1	2	3	4	5
T+SVR+L	0.601	0.599	0.604	0.617	0.618
T+SVR+S	0.62	0.621	0.629	0.634	0.632
T+RR	0.615	0.615	0.614	0.62	0.623
BT+SVR+L	0.628	0.635	0.665	0.627	**0.669**
BT+SVR+S	**0.683**	0.641	0.659	**0.695**	0.667
BT+RR	0.638	**0.654**	**0.675**	0.677	0.654
B1+SVR+L	0.613	0.609	0.622	0.628	0.632
B1+SVR+S	0.602	0.600	0.601	0.605	0.598
B1+RR	0.614	0.611	0.603	0.607	0.606
B2+SVR+L	0.528	0.528	0.538	0.536	0.523
B2+SVR+S	0.472	0.466	0.473	0.477	0.471
B2+RR	0.464	0.460	0.466	0.472	0.476

In Table 9.8, the P-values of a two-sample t-test on the target test set are reported for each run. The proposed prediction algorithms BT+SVR+S and BT+RR yield highly statistically significant results in each run (see results colored in red in Table 9.8), and these highly statistically significant results reflect the high performance of BT+SVR+S and BT+RR prediction algorithms (see Table 9.7).

Table 9.8 P-Values of a Two-Sample t-Test for the Twelve Prediction Algorithms on the Target Test Set Where the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Triple-Negative Breast Cancer Patients, Respectively, Are Used. Each Prediction Algorithm with $P < 0.001$ Is Considered Highly Statistically Significant and Colored in Red. Each Prediction Algorithm with $P < 0.05$ Is Considered Statistically Significant and Colored in Blue

Run	1	2	3	4	5
T+SVR+L	8701×10^{-6}	9608×10^{-6}	7425×10^{-6}	4303×10^{-6}	4126×10^{-6}
T+SVR+S	3877×10^{-6}	3499×10^{-6}	2547×10^{-6}	1708×10^{-6}	1897×10^{-6}
T+RR	3908×10^{-6}	4911×10^{-6}	5189×10^{-6}	363×10^{-5}	3932×10^{-6}
BT+SVR+L	2698×10^{-6}	1171×10^{-6}	188×10^{-6}	2163×10^{-6}	132×10^{-6}
BT+SVR+S	2716×10^{-8}	719×10^{-6}	185×10^{-6}	1185×10^{-8}	104×10^{-6}
BT+RR	382×10^{-6}	111×10^{-6}	2692×10^{-8}	2774×10^{-8}	115×10^{-6}
B1+SVR+L	6×10^{-3}	7×10^{-3}	4×10^{-3}	2×10^{-3}	2×10^{-3}
B1+SVR+S	8×10^{-3}	1×10^{-2}	11×10^{-3}	9×10^{-3}	14×10^{-3}
B1+RR	261×10^{-5}	4×10^{-3}	6×10^{-3}	4×10^{-3}	5×10^{-3}
B2+SVR+L	4249×10^{-4}	426×10^{-3}	3987×10^{-4}	3964×10^{-4}	4338×10^{-4}
B2+SVR+S	8505×10^{-4}	8672×10^{-4}	8341×10^{-4}	8028×10^{-4}	8263×10^{-4}
B2+RR	6622×10^{-4}	694×10^{-3}	6596×10^{-4}	6066×10^{-4}	57×10^{-2}

Figures 9.5(a), 9.5(b), and 9.5(c) show the predictions of BT+SVR+S, B1+RR, and B2+RR, respectively, on the target test set at the first run. BT+SVR+S (Figure 9.5(a)) achieved a highly statistically significant result ($P = 2716\times10^{-8}$ from a two-sample t-test). The result of B1+RR (Figure 9.5(b)) was statistically significant with ($P = 261\times10^{-5}$ from a two-sample t-test). The result of B2+RR (Figure 9.5(c)) was not statistically significant ($P = 6622\times10^{-4}$ from a two-sample t-test). In Figure 9.5(d), the ROC curves reveal AUC values of 0.683, 0.614, and 0.464 for BT+SVR+S, B1+RR, and B2+RR, respectively.

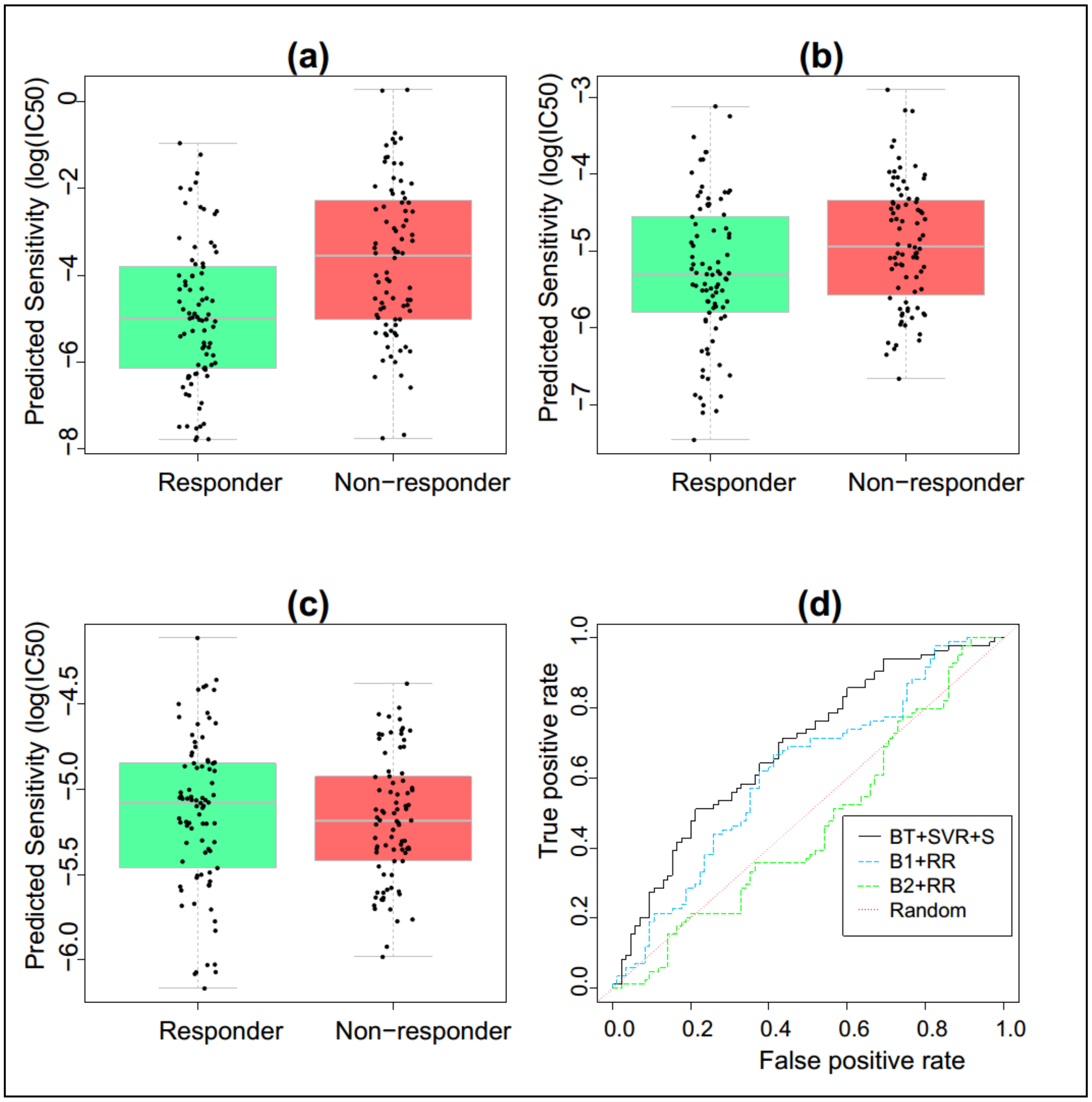


Figure 9.5 Predictions of bortezomib sensitivity on the target test set of multiple myeloma patients where the target training set and auxiliary dataset pertaining to multiple myeloma and triple-negative breast cancer patients, respectively, are used. Strip charts and boxplots (a), (b), and (c) show the differences in predicted drug sensitivity to bortezomib treatment between the responder (i.e., sensitive) group and non-responder (i.e., resistant) group using BT+SVR+S, B1+RR, and B2+RR, respectively. (d) shows the ROC curves of the prediction algorithms, which reveal the proportion of true positives compared to the proportion of false positives. ROC = receiver operating characteristic.

Figure 9.6 shows that the proposed prediction algorithms BT+SVR+S, BT+RR, BT+SVR+L, and T+SVR+S outperform the baseline prediction algorithms w.r.t. the MAUC.

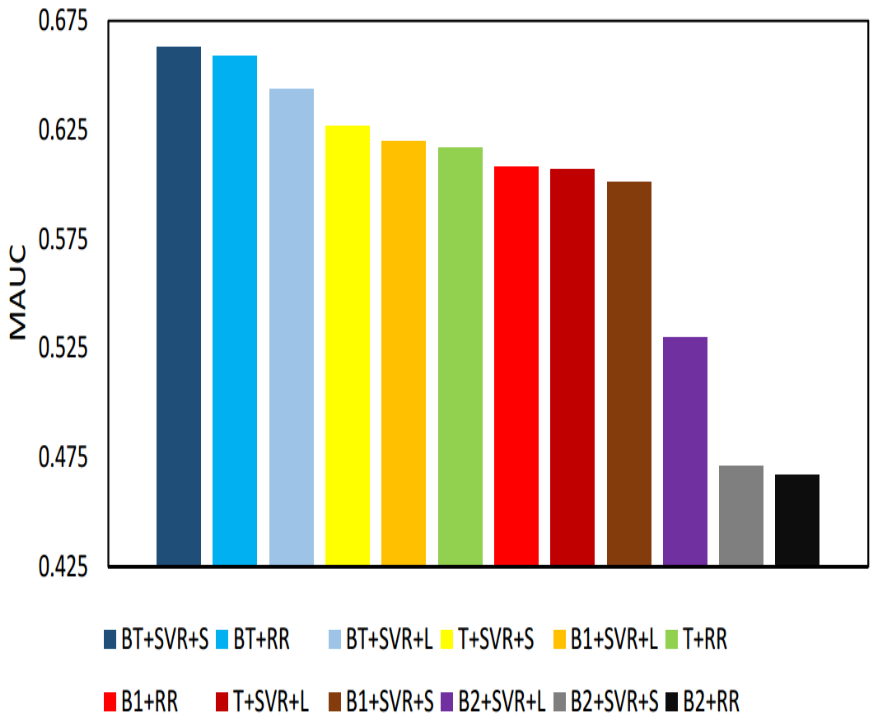


Figure 9.6 The Mean AUC (MAUC) values of the twelve bortezomib sensitivity prediction algorithms for multiple myeloma patients. The algorithms are ranked from left to right where the leftmost algorithm has the highest MAUC and the rightmost algorithm has the lowest MAUC.

9.4.3.3 Exploiting Non-Small Cell lung cancer Auxiliary Data. Table 9.9 shows details of the target training set and auxiliary dataset pertaining to non-small cell lung cancer patients used by each prediction algorithm. The target training set is obtained from the target task (i.e., prediction of bortezomib sensitivity in multiple myeloma patients) and the auxiliary dataset is acquired from the related task (i.e., prediction of erlotinib sensitivity in non-small cell lung cancer patients). Here, different auxiliary dataset pertaining to non-small cell lung cancer patients is used, while the target test set still remains the same.

Table 9.9 Details of the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Non-Small Cell Lung Cancer Patients, Respectively, Used by Each Prediction Algorithm

Run	1	2	3	4
m/l	280/258	278/258	275/258	272/258
p/n	9114/9507	9114/9507	9114/9507	9114/9507
$p \cap n$	7855	7855	7855	7855
$P_{m/l}$	280/100	278/100	275/100	272/100
$P_{p/n}$	9114/9114	9114/9114	9114/9114	9114/9114
$B1_m$	280	278	275	272
$B1_p$	9114	9114	9114	9114
$B2_{m/l}$	280/258	278/258	275/258	272/258
$B2_{p/n}$	9114/9114	9114/9114	9114/9114	9114/9114

Table 9.10 shows the AUCs of twelve prediction algorithms on the target test set of multiple myeloma patients. The proposed prediction algorithm BT+SVR+S achieved the highest AUC scores in 4 out of 5 runs. The high performance results indicate the stability and superiority of BT approach when using SVR+S.

Table 9.10 AUC Scores of the Twelve Prediction Algorithms on the Target Test Set of Multiple Myeloma Patients Where the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Non-Small Cell Lung Cancer Patients, Respectively, Are Used. The Algorithm with the Highest AUC Is Shown in Bold. Std Is the Standard Deviation of the AUC Values Obtained from the Five Runs

Run	1	2	3	4	5
T+SVR+L	0.6	0.598	0.604	0.617	0.617
T+SVR+S	0.621	0.619	0.63	0.635	0.634
T+RR	0.614	0.615	0.61	0.615	0.624
BT+SVR+L	0.67	0.653	0.623	**0.663**	0.663
BT+SVR+S	**0.673**	**0.653**	**0.651**	0.657	**0.678**
BT+RR	0.658	0.639	0.619	0.594	0.627
B1+SVR+L	0.613	0.609	0.622	0.628	0.632
B1+SVR+S	0.602	0.6	0.601	0.605	0.598
B1+RR	0.614	0.611	0.603	0.607	0.606
B2+SVR+L	0.403	0.409	0.436	0.422	0.421
B2+SVR+S	0.643	0.644	0.644	0.646	0.648
B2+RR	0.641	0.641	0.642	0.641	0.642

Table 9.11 reports the P-values of a two-sample t-test on the target test set for each run. The proposed prediction algorithm BT+SVR+S yields highly statistically significant result in each run (see results colored in red in Table 9.11), and these highly statistically significant results reflect the high performance of BT+SVR+S prediction algorithm (see Table 9.10).

Table 9.11 P-Values of a Two-Sample *t*-Test for the Twelve Prediction Algorithms on the Target Test Set Where the Target Training Set and Auxiliary Dataset Pertaining to Multiple Myeloma and Non-Small Cell Lung Cancer Patients, Respectively, Are Used. Each Prediction Algorithm with $P < 0.001$ Is Considered Highly Statistically Significant and Colored in Red. Each Prediction Algorithm with $P < 0.05$ Is Considered Statistically Significant and Colored in Blue

Run	1	2	3	4	5
T+SVR+L	8887×10^{-6}	9892×10^{-6}	7654×10^{-6}	4282×10^{-6}	4214×10^{-6}
T+SVR+S	3883×10^{-6}	3735×10^{-6}	2486×10^{-6}	1606×10^{-6}	1833×10^{-6}
T+RR	3887×10^{-6}	49×10^{-4}	6343×10^{-6}	4458×10^{-6}	3918×10^{-6}
BT+SVR+L	135×10^{-6}	368×10^{-6}	3893×10^{-6}	212×10^{-6}	18×10^{-5}
BT+SVR+S	1195×10^{-7}	261×10^{-6}	394×10^{-6}	383×10^{-6}	4695×10^{-8}
BT+RR	3887×10^{-6}	49×10^{-4}	6343×10^{-6}	4458×10^{-6}	3918×10^{-6}
B1+SVR+L	6×10^{-3}	7×10^{-3}	4×10^{-3}	2×10^{-3}	2×10^{-3}
B1+SVR+S	8×10^{-3}	1×10^{-2}	11×10^{-3}	9×10^{-3}	14×10^{-3}
B1+RR	261×10^{-5}	4×10^{-3}	6×10^{-3}	4×10^{-3}	5×10^{-3}
B2+SVR+L	939×10^{-3}	9358×10^{-4}	9073×10^{-4}	9173×10^{-4}	926×10^{-3}
B2+SVR+S	5556×10^{-6}	5455×10^{-6}	5414×10^{-6}	5493×10^{-6}	5196×10^{-6}
B2+RR	3881×10^{-6}	3855×10^{-6}	3888×10^{-6}	3905×10^{-6}	3822×10^{-6}

Figures 9.7(a), 9.7(b), and 9.7(c) show the predictions of BT+SVR+S, B1+RR, and B2+RR, respectively, on the target test set at the first run. BT+SVR+S (Figure 9.7(a)) yielded a highly statistically significant result ($P = 1195\times10^{-7}$ from a two-sample t-test). The result of B1+RR (Figure 9.7(b)) was statistically significant with ($P = 261\times10^{-5}$ from a two-sample t-test). B2+RR (Figure 9.7(c)) yielded a statistically significant result ($P = 3381\times10^{-6}$ from a two-sample t-test). In Figure 9.7(d), the ROC curves reveal AUC values of 0.673, 0.614, and 0.641 for BT+SVR+S, B1+RR, and B2+RR, respectively.

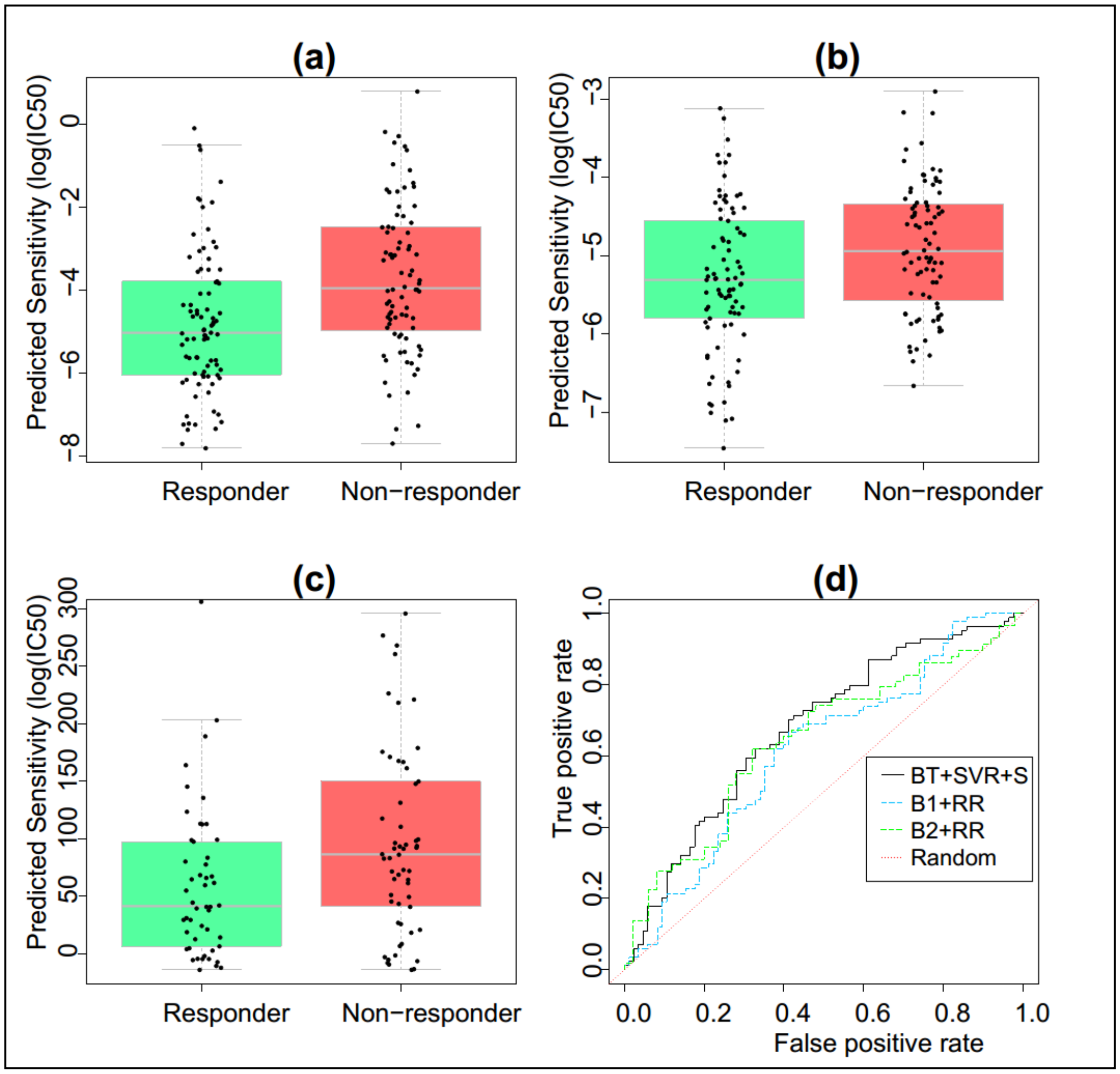


Figure 9.7 Predictions of bortezomib sensitivity on the target test set of multiple myeloma patients where the target training set and auxiliary dataset pertaining to multiple myeloma and non-small cell lung cancer patients, respectively, are used. Strip charts and boxplots (a), (b), and (c) show the differences in predicted drug sensitivity to bortezomib treatment between the responder (i.e., sensitive) group and non-responder (i.e., resistant) group using BT+SVR+S, B1+RR, and B2+RR, respectively. (d) shows the ROC curves of the prediction algorithms, which reveal the proportion of true positives compared to the proportion of false positives. ROC = receiver operating characteristic.

Figure 9.8 shows that the proposed prediction algorithms BT+SVR+S and BT+SVR+L outperform the baseline prediction algorithms w.r.t. the MAUC.

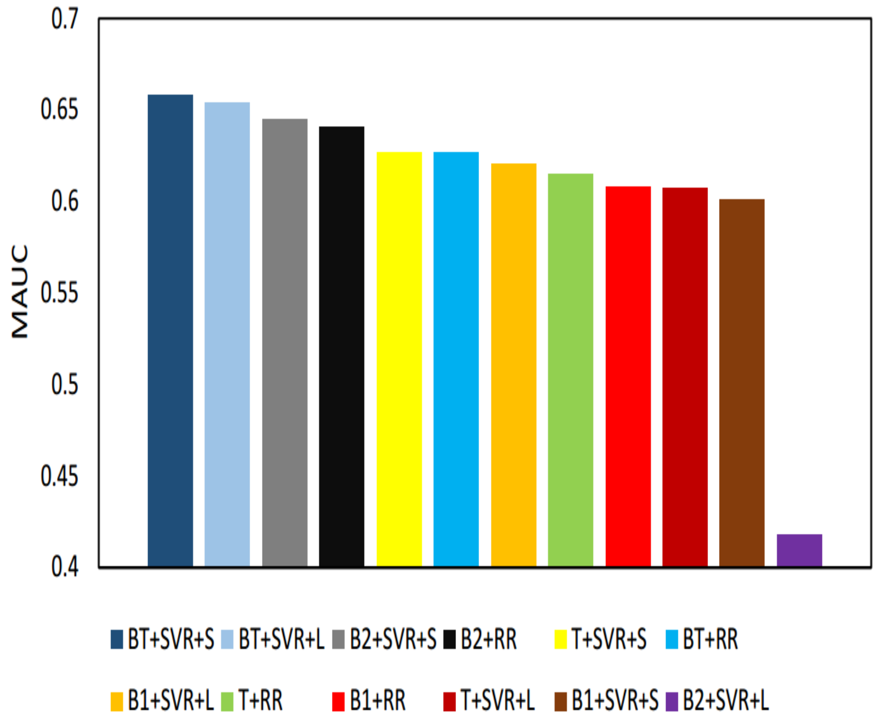


Figure 9.8 The Mean AUC (MAUC) values of the twelve bortezomib sensitivity prediction algorithms for multiple myeloma patients. The algorithms are ranked from left to right where the leftmost algorithm has the highest MAUC and the rightmost algorithm has the lowest MAUC.

9.5 Discussion

The experiments show that the proposed approaches significantly outperform existing approaches. Further, the proposed approaches are well-suited for a wide range of tasks, such as integration of different types of biological data to increase the accuracy of gene regulatory networks (GRN) inference [16, 110, 172, 173], and integration of different cancer data to improve the prediction performance in a given drug sensitivity task.

As in [54], the mapping of predicted continuous values to categorical labels was performed using the ROCR package [174]. The details of the mapping algorithm can be found in [175]. In a nutshell, the algorithm sorts the predicted continuous values in increasing order. The algorithm works iteratively by examining one value at a time, from the smallest to the largest value. When examining a particular value v, the algorithm labels v and all the values larger than or equal to v as "resistant" (i.e., positive) and all the values smaller than v as "sensitive" (i.e., negative). The algorithm compares these "resistant" and "sensitive" labels with the corresponding true labels in the target test set to build a confusion matrix. The true positive rate (TPR) and false positive rate (FPR) with respect to the value v are then calculated and plotted. After all the predicted continuous values are examined, multiple points are plotted, where the x-coordinate of a point is a FPR and the y-coordinate of the point is a TPR. These points constitute the ROC curve of the prediction algorithm.

The biological rationale behind the superior results of the approaches is that combining cancer drugs is often used to achieve enhanced therapeutic efficacy in the treatment [176]. For example, docetaxel (chemotherapy drug) is used to treat breast cancer in combination with other specific chemotherapy drugs [170] [177]. Bortezomib and docetaxel combination has been used as a therapy for breast cancer [178, 179]. Hence, the task of predicting bortezomib sensitivity in multiple myeloma patients is closely related to the task of predicting docetaxel sensitivity in breast cancer patients, where closeness plays an important role in machine learning. For example, suppose we are given an unseen example (i.e., test example). If the unseen example has an expression profile closer to given training example with corresponding response (i.e., drug value), then the unseen example is most likely to have a closer response to the response associated with the given training example. The same holds for combining bortezomib and cisplatin, which clinically led to synergistic killing of head and neck squamous cell carcinoma (HNSCC) cells [180]. In addition, erlotinib plus bortezomib showed synergistic antitumor activity against the H460 Non-small cell lung cancer (NSCLC) cell line [181].

In the proposed approaches, it is assumed that the number of features in the target training set is greater than the number of features in the auxiliary data. Then, the top q features in the target test set are selected using the highest q statistical leverage scores computed on the target training set. However, If the number of features in the auxiliary data is greater than number of features in the target training set (as the case in triple-negative breast cancer and non-small cell lung cancer auxiliary data), then the top q features are selected from the auxiliary data using the highest q statistical leverage scores computed from the auxiliary data, where q equals the number of features as in the target training set, and no further feature selection is performed on the target training and target test sets.

In this work, differences in distributions between the target training set and auxiliary data have contributed to the degraded performance on the target test set for prediction algorithms employing the second baseline approach (B2), which does not have a transfer mechanism like ours.

It is worth mentioning that the performance of other machine learning algorithms is also assessed, including random forests [121], support vector regression with a polynomial kernel of degree 2, and support vector regression with a Gaussian kernel. However, they exhibited poor performance; consequently, their results are not included in this paper.

9.6 Summary

In this paper, two approaches are presented to improve drug sensitivity prediction: a transfer learning approach and a boosted transfer learning approach. The transfer learning approach works by (1) performing a feature selection step to balance the number of features; (2) changing the representation of auxiliary data of a related task to a new representation that is closer to the target training set; and (3) combining the target training set with each one of the auxiliary data separately, and using the result as input to a standard machine learning algorithm. The boosted transfer learning approach boosts the performance of the transfer learning approach using a modified version of AdaBoost.

The proposed approaches employ two machine learning algorithms: support vector regression and ridge regression. The experimental results demonstrate the stability of the proposed transfer learning approaches. The proposed approaches outperform the baseline approaches including an existing approach as measured by their higher and statistically significant AUC scores.

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

In this dissertation, new machine learning and data mining algorithms have been developed and applied to solve two important biomedical problems: (i) gene network inference; and (ii) drug response prediction.

Future work for gene network inference includes: (i) applying the proposed learning framework in the third chapter to other unsupervised network inference tools and evaluating its performance when used with those tools; (ii) exploring and assessing the feasibility of the framework using new feature learning and data classification methods including deep learning algorithms; (iii) applying the proposed hybrid approach in the fourth chapter to some well-studied organisms such as *E. coli* and yeast; (iv) assessing the feasibility of other unsupervised methods when using the hybrid approach; (v) exploring new data cleaning and link prediction algorithms such as matrix completion [182, 183] and evaluating their performance on both the DREAM datasets and datasets from the well-studied organisms; and (vi) extending sampling and boosting techniques in the fifth chapter to miRNA-mediated regulatory networks [118, 119].

The work for drug sensitivity prediction opens possibilities for future work, e.g., (i) exploiting signaling pathways to select the most important features and incorporating them into the proposed link prediction approach in the seventh chapter; (ii) extending the transfer learning approaches in the ninth chapter to handle auxiliary data from multiple related tasks simultaneously; (iii) collaborating with domain experts, where signaling pathways are leveraged to improve the prediction performance on a given drug sensitivity prediction task; and (iv) adopting new feature representation methods to improve the proposed transfer learning approaches for other drug sensitivity prediction tasks.

APPENDIX A

EVALUATING PREDICTION ALGORITHMS ON CLINICAL TRIAL DATA OF NON-SMALL CELL LUNG CANCER AND TRIPLE-NEGATIVE BREAST CANCER

Table A.1 shows the performance of 9 erlotinib sensitivity prediction algorithms on non-small cell lung cancer data of patients. The prediction algorithms were evaluated using Spearman correlation test, where statistical significant results of prediction algorithms on the test set are shown in bold. The results indicate that prediction algorithms of the baseline were not statistically significant compared to ours. Specifically, A1+RR statistically significantly outperforming the baseline prediction algorithms, where *p* are calculated according to Spearman correlation test.

As the number of cell lines which respond to erlotinib is small and model is fitting a huge amount of noisy cell lines that affected the model, B+RR (proposed prediction algorithm via Geeleher *et al*.) results were not significant. Geeleher *et al*. [54] tackled this problem that caused poor performance as follows. They changed linear ridge regression and fitted model using logistic ridge regression model on the 15 most sensitive and 55 most resistant samples and then applied model to the test set (i.e., clinical trial data), where their model then achieved significant results (rho = 0.64 $p = 5.3 \times 10^{-4}$ and from a Spearman's correlation test). In contrast, the proposed prediction algorithms achieved statistical significant results using linear ridge regression and this shows that the proposed new feature representation is discriminative guiding the learning algorithm (RR) when incorporated with the proposed prediction algorithms.

Table A.1 Prediction of Erlotinib Sensitivity in NSCLC (Non-Small Cell Lung Cancer) Patients. Values with Statistical Significance (p<0.05) Are Shown in Bold. Results Are Shown As (p/rho) According to Spearman Correlation Test

m	258	256	253	250	247
d	9507	9507	9507	9507	9507
A1+SVR+L	**0.034**/-0.368	0.099/-0.265	0.111/-0.252	**0.047**/-0.341	**0.047**/-0.341
A1+SVR+S	0.053/-0.330	0.061/-0.315	0.078/-0.292	**0.035**/-0.367	0.065/-0.310
A1+RR	**0.007**/-0.480	**0.018**/-0.421	**0.013**/-0.441	**0.011**/-0.453	**0.010**/-0.458
m+A1	134	133	132	130	129
d+A1	19014	19014	19014	19014	19014
A2+SVR+L	0.394/-0.056	0.613/0.060	0.569/0.036	0.548/0.025	0.486/-0.007
A2+SVR+S	0.275/-0.125	0.382/-0.063	0.348/-0.082	0.290/-0.115	0.320/-0.098
A2+RR	**0.028/**-0.385	0.127/-0.236	0.070/-0.302	0.069/-0.304	0.122/-0.241
m+A2	134	133	132	130	129
d+A2	9507	9507	9507	9507	9507
B+SVR+L	0.349/-0.081	0.399/-0.053	0.486/-0.007	0.426/-0.038	0.478/-0.011
B+SVR+S	0.571/0.037	0.668/0.091	0.610/0.059	0.584/0.045	0.599/0.053
B+RR	0.285/-0.119	0.350/-0.080	0.309/-0.104	0.266/-0.130	0.273/-0.126

Table A.2 shows the performance of 3 cisplatin sensitivity prediction algorithms in triple-negative breast cancer patients' data. The p-values used to evaluate prediction algorithms are from a linear regression model. Hence, approaches that employ ridge regression are evaluated. Geeleher *et al*. [54] assessed the response of 24 triple-negative breast cancer patients to neoadjuvant cisplatin therapy. Each patient is assigned to one of four drug response categories based on RECIST [184]. B+RR did not capture variability in clinical response (see Table A.2 below). The proposed prediction algorithm achieved comparable results, which were not statistically significant.

Table A.2 Prediction of Cisplatin Sensitivity in Triple-Negative Breast Cancer Patients. Results Are Recorded According to p-Values from a Linear Regression Model

m	497	493	488	483	478
d	9620	9620	9620	9620	9620
A1+RR	0.1013	0.05781	0.08953	0.1775	0.1708
m+A1	254	252	249	247	244
d+A1	19240	19240	19240	19240	19240
A2+RR	0.192	0.1413	0.192	0.09515	0.1213
m+A2	254	252	249	247	244
d+A2	9620	9620	9620	9620	9620
B+RR	0.262	0.2055	0.185	0.2119	0.2306

APPENDIX B

EVALUATING SIGNALING PATHWAYS AS A CONSTRAINT TO GET RELIABLE FEATURE SETS

The performance of prediction algorithms employing ridge regression was assessed including, the baseline prediction algorithm of Geeleher *et al*. (B+RR) on the test sets of breast cancer and multiple myeloma.

For Breast Cancer: Hedgehog (Hh), Notch, and Wnt signaling pathways were used, which have been reported as a novel therapeutic target in breast cancer [185-187]. These pathways consist of 217 unique genes and 113 of them could be found in our dataset which were used in the proposed model. **For Multiple Myeloma (MM)**, Jak-STAT, PI3K-Akt, and NF-kappa B were used as the classic signaling pathways underlying MM [187, 188]. There are 512 unique genes in these pathways and 341 of them could be found in the dataset which were used in the proposed model. It is found found that the performance of B+RR (i.e., the baseline) and A1+RR has significantly degraded where all achieved AUC below 0.60 on both test sets for breast cancer and multiple myeloma. This shows that removing discriminative features significantly degrade the performance of all prediction algorithms including the baseline (B+RR) as these discriminative features are crucial to improve the prediction performance.

REFERENCES

[1] M. A. Hasan, and M. J. Zaki, "A Survey of Link Prediction in Social Networks," *Social Network Data Analytics*, C. C. Aggarwal, ed., Boston, MA: Springer US, 2011, pp. 243-275.

[2] A. K. Menon, and C. Elkan, “Link prediction via matrix factorization,” in the Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases, 2011, pp. 437-452.

[3] D. Liben-Nowell, and J. Kleinberg, “The link-prediction problem for social networks,” *Journal of the American Society for Information Science and Technology,* vol. 58, no. 7, 2007, pp. 1019-1031.

[4] R. De Smet, and K. Marchal, “Advantages and limitations of current network inference methods,” *Nature Reviews Microbiology,* vol. 8, no. 10, 2010, pp. 717-729.

[5] L. Cerulo, C. Elkan, and M. Ceccarelli, “Learning gene regulatory networks from only positive and unlabeled data,” *BioMed Central (BMC) Bioinformatics,* vol. 11, no. 1, 2010, pp. 1.

[6] J. T. L. Wang, “Inferring Gene Regulatory Networks: Challenges and Opportunities,” *Journal of Data Mining in Genomics & Proteomics,* vol. 6, no. 1, 2015, pp. 1.

[7] Z. Gillani, M. S. Akash, M. M. Rahaman, and M. Chen, “CompareSVM: supervised, Support Vector Machine (SVM) inference of gene regularity networks,” *BioMed Central (BMC) Bioinformatics,* vol. 15, no. 1, 2014, pp. 395.

[8] F. Mordelet, and J.-P. Vert, “SIRENE: supervised inference of regulatory networks,” *Bioinformatics,* vol. 24, no. 16, 2008, pp. i76-i82.

[9] J. Ernst, Q. K. Beg, K. A. Kay, G. Balázsi, Z. N. Oltvai, and Z. Bar-Joseph, “A semi-supervised method for predicting transcription factor–gene interactions in Escherichia coli,” *Public Library of Science (PLoS) Computational Biology,* vol. 4, no. 3, 2008, pp. e1000044.

[10] L. Getoor, and C. P. Diehl, “Link mining: a survey,” *Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations,* vol. 7, no. 2, 2005, pp. 3-12.

[11] J. Leskovec, D. Huttenlocher, and J. Kleinberg, “Predicting positive and negative links in online social networks,” in the Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 641-650.

[12] B. Barzel, and A.-L. Barabási, “Network link prediction by global silencing of indirect correlations,” *Nature Biotechnology,* vol. 31, no. 8, 2013, pp. 720-725.

[13] A. Clauset, C. Moore, and M. E. Newman, “Hierarchical structure and the prediction of missing links in networks,” *Nature,* vol. 453, no. 7191, 2008, pp. 98-101.

[14] M. Elloumi, Iliopoulos, C.S., Wang, J.T.L., and Zomaya, A.Y., "Pattern Recognition in Computational Molecular Biology," *Wiley Series on Bioinformatics: Computational Techniques and Engieering,* New York City, NY: Wiley US, 2015.

[15] S. R. Maetschke, P. B. Madhamshettiwar, M. J. Davis, and M. A. Ragan, “Supervised, semi-supervised and unsupervised inference of gene regulatory networks,” *Briefings in Bioinformatics,* 2013, pp. 195-211.

[16] T. Turki, and J. T. L. Wang, “A new approach to link prediction in gene regulatory networks,” in Proceedings of the 16th International Conference on Intelligent Data Engineering and Automated Learning, 2015, pp. 404-415.

[17] N. Patel, and J. T. L. Wang, “Semi-supervised prediction of gene regulatory networks using machine learning algorithms,” *Journal of Biosciences,* vol. 40, no. 4, 2015, pp. 731-740.

[18] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis, “Advances to Bayesian network inference for generating causal networks from observational biological data,” *Bioinformatics,* vol. 20, no. 18, 2004, pp. 3594-3603.

[19] P. Zoppoli, S. Morganella, and M. Ceccarelli, “TimeDelay-ARACNE: Reverse engineering of gene networks from time-course data by an information theoretic approach,” *BioMed Central (BMC) Bioinformatics,* vol. 11, no. 1, 2010, pp. 1.

[20] A. Madar, A. Greenfield, E. Vanden-Eijnden, and R. Bonneau, “DREAM3: network inference using dynamic context likelihood of relatedness and the inferelator,” *Public Library of Science (PloS) One*, vol. 5, no. 3, 2010, pp. e9803.

[21] A. Greenfield, A. Madar, H. Ostrer, and R. Bonneau, “DREAM4: Combining genetic and dynamic information to identify biological networks and dynamical models,” *Public Library of Science (PloS) One*, vol. 5, no. 10, 2010, pp. e13397.

[22] G. Krouk, P. Mirowski, Y. LeCun, D. E. Shasha, and G. M. Coruzzi, “Predictive network modeling of the high-resolution dynamic plant transcriptome in response to nitrate,” *Genome Biology*, vol. 11, no. 12, 2010, pp. 1-19.

[23] P. Vera-Licona, A. Jarrah, L. D. Garcia-Puente, J. McGee, and R. Laubenbacher, “An algebra-based method for inferring gene regulatory networks,” *BioMed Central (BMC) Systems Biology*, vol. 8, no. 1, 2014, pp. 1.

[24] A. F. Villaverde, J. Ross, F. Morán, and J. R. Banga, “Mider: network inference with mutual information distance and entropy reduction,” *Public Library of Science (PloS) One*, vol. 9, no. 5, 2014, pp. e96732.

[25] G. Sanguinetti, “Combining tree-based and dynamical systems for the inference of gene regulatory networks,” *Bioinformatics*, no. 2015, pp. 1614-1622.

[26] W. C. Young, A. E. Raftery, and K. Y. Yeung, “Fast Bayesian inference for gene regulatory networks using ScanBMA,” *BioMed Central (BMC) Systems Biology*, vol. 8, no. 1, 2014, pp. 47.

[27] R. Bonneau, D. J. Reiss, P. Shannon, M. Facciotti, L. Hood, N. S. Baliga, and V. Thorsson, “The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo,” *Genome Biology*, vol. 7, no. 5, 2006, pp. R36.

[28] C. Angelini, and V. Costa, “Understanding gene regulatory mechanisms by integrating ChIP-seq and RNA-seq data: statistical solutions to biological problems,” *Frontiers in Cell and Developmental Biology*, vol. 2, no. 51, 2014.

[29] T. Werner, S. Dombrowski, C. Zgheib, F. A. Zouein, H. L. Keen, M. Kurdi, and G. W. Booz, “Elucidating functional context within microarray data by integrated transcription factor focused gene-interaction and regulatory network analysis,” *European Cytokine Network*, vol. 24, no. 2, 2013, pp. 75-90.

[30] P. Brazhnik, A. de la Fuente, and P. Mendes, “Gene networks: how to put the function in genomics,” *Trends Biotechnology,* vol. 20, no. 11, 2002, pp. 467-472.

[31] T. Ideker, and N. J. Krogan, “Differential network biology,” *Molecular Systems Biology,* vol. 8, 2012, pp. 565.

[32] A. A. Margolin, K. Wang, W. K. Lim, M. Kustagi, I. Nemenman, and A. Califano, “Reverse engineering cellular networks,” *Nature Protocols,* vol. 1, no. 2, 2006, pp. 662-71.

[33] S. R. Maetschke, P. B. Madhamshettiwar, M. J. Davis, and M. A. Ragan, “Supervised, semi-supervised and unsupervised inference of gene regulatory networks,” *Briefings in Bioinformatics,* vol. 15, no. 2, 2014, pp. 195-211.

[34] J. Ernst, Q. K. Beg, K. A. Kay, G. Balazsi, Z. N. Oltvai, and Z. Bar-Joseph, “A semi-supervised method for predicting transcription factor-gene interactions in Escherichia coli,” *Public Library of Science (PloS) Computational Biology,* vol. 4, no. 3, 2008, pp. e1000044.

[35] N. Patel, and J. T. L. Wang, “Semi-supervised prediction of gene regulatory networks using machine learning algorithms,” *Journal of Biosciences,* vol. 40, no. 4, 2015, pp. 731-40.

[36] M. Altaf-Ul-Amin, F. M. Afendi, S. K. Kiboi, and S. Kanaya, “Systems biology in the context of big data and networks,” *BioMed Research International,* vol. 2014, 2014.

[37] M. Altaf-Ul-Amin, T. Katsuragi, T. Sato, and S. Kanaya, “A glimpse to background and characteristics of major molecular biological networks,” *BioMed Research International,* vol. 2015, 2015.

[38] T. Ideker, and N. J. Krogan, “Differential network biology,” *Molecular Systems Biology,* vol. 8, no. 1, 2012, pp. 565.

[39] N. Kibinge, N. Ono, M. Horie, T. Sato, T. Sugiura, M. Altaf-Ul-Amin, A. Saito, and S. Kanaya, “Integrated pathway-based transcription regulation network mining and visualization based on gene expression profiles,” *Journal of Biomedical Informatics,* vol. 61, 2016, pp. 194-202.

[40] A. A. Margolin, K. Wang, W. K. Lim, M. Kustagi, I. Nemenman, and A. Califano, “Reverse engineering cellular networks,” *Nature Protocols,* vol. 1, no. 2, 2006, pp. 662-671.

[41] T. Turki, Wang, Jason T. L., Rajikhan, Ibrahim, “Inferring gene regulatory networks by combining supervised and unsupervised methods.,” in the Proceedings of the 15th IEEE International Conference on Machine Learning and Applications, 2016.

[42] R. D. Leclerc, “Survival of the sparsest: robust gene networks are parsimonious,” *Molecular Systems Biology,* vol. 4, no. 1, 2008, pp. 213.

[43] Z. Gillani, M. Akash, M. Rahaman, and M. Chen, “CompareSVM: supervised, Support Vector Machine (SVM) inference of gene regularity networks,” *BioMed Central (BMC) Bioinformatics,* vol. 15, no. 1, 2014, pp. 395.

[44] R. L. Siegel, K. D. Miller, and A. Jemal, “Cancer statistics, 2015,” *CA: A Cancer Journal for Clinicians,* vol. 65, no. 1, 2015, pp. 5-29.

[45] A. Kamb, S. Wee, and C. Lengauer, “Why is cancer drug discovery so difficult?,” *Nature Reviews Drug Discovery,* vol. 6, no. 2, 2007, pp. 115-120.

[46] V. Marx, “Cancer: A most exceptional response,” *Nature,* vol. 520, no. 7547, 2015, pp. 389-393.

[47] N. C. Turner, and J. S. Reis-Filho, “Genetic heterogeneity and cancer drug resistance,” *The Lancet Oncology,* vol. 13, no. 4, pp. e178-e185.

[48] D. M. Roden, and A. L. George Jr, “The genetic basis of variability in drug responses,” *Nature Reviews Drug Discovery,* vol. 1, no. 1, 2002, pp. 37-44.

[49] M. W. Libbrecht, and W. S. Noble, “Machine learning applications in genetics and genomics,” *Nature Reviews Genetics,* vol. 16, no. 6, 2015, pp. 321-332.

[50] F. Sanchez-Garcia, P. Villagrasa, J. Matsui, D. Kotliar, V. Castro, U.-D. Akavia, B.-J. Chen, L. Saucedo-Cuevas, R. Rodriguez Barrueco, D. Llobet-Navas, Jose M. Silva, and D. Pe’er, “Integration of Genomic Data Enables Selective Discovery of Breast Cancer Drivers,” *Cell,* vol. 159, no. 6, pp. 1461-1475.

[51] P. Zhang, and V. Brusic, “Mathematical modeling for novel cancer drug discovery and development,” *Expert Opinion on Drug Discovery,* vol. 9, no. 10, 2014, pp. 1133-1150.

[52] D. G. Covell, “Data Mining Approaches for Genomic Biomarker Development: Applications Using Drug Screening Data from the Cancer Genome Project and the Cancer Cell Line Encyclopedia,” *Public Library of Science (PloS) One,* vol. 10, no. 7, 2015, pp. e0127433.

[53] J. C. Costello, L. M. Heiser, E. Georgii, M. Gonen, M. P. Menden, N. J. Wang, M. Bansal, M. Ammad-ud-din, P. Hintsanen, S. A. Khan, J.-P. Mpindi, O. Kallioniemi, A. Honkela, T. Aittokallio, K. Wennerberg, N. D. Community, J. J. Collins, D. Gallahan, D. Singer, J. Saez-Rodriguez, S. Kaski, J. W. Gray, and G. Stolovitzky, “A community effort to assess and improve drug sensitivity prediction algorithms,” *Nature Biotechnology,* vol. 32, no. 12, 2014, pp. 1202-1212.

[54] P. Geeleher, N. J. Cox, and R. S. Huang, “Clinical drug response can be predicted using baseline gene expression levels and in vitro drug sensitivity in cell lines,” *Genome Biology,* vol. 15, no. 3, 2014, pp. R47-R47.

[55] B. Yadav, P. Gopalacharyulu, T. Pemovska, S. A. Khan, A. Szwajda, J. Tang, K. Wennerberg, and T. Aittokallio, “From drug response profiling to target addiction scoring in cancer cell models,” *Disease Models and Mechanisms,* vol. 8, no. 10, 2015, pp. 1255-1264.

[56] M. P. Menden, F. Iorio, M. Garnett, U. McDermott, C. H. Benes, P. J. Ballester, and J. Saez-Rodriguez, “Machine Learning Prediction of Cancer Cell Sensitivity to Drugs Based on Genomic and Chemical Properties,” *Public Library of Science (PloS) One,* vol. 8, no. 4, 2013, pp. e61318.

[57] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, no. 8, 2013, pp. 1798-1828.

[58] M. Mohri, A. Rostamizadeh, and A. Talwalkar, "Foundations of machine learning," Cambridge, MA: MIT press, 2012.

[59] A. Coates, and A. Y. Ng, "Learning feature representations with k-means," *Neural Networks: Tricks of the Trade*, Berlin, Heidelberg: Springer, 2012, pp. 561-580.

[60] R. L. Siegel, K. D. Miller, and A. Jemal, “Cancer statistics, 2016,” *CA: A Cancer Journal for Clinicians,* vol. 66, no. 1, 2016, pp. 7-30.

[61] A. C. Society, "Cancer facts and figures 2016," American Cancer Society, 2016.

[62] P. L. Bedard, A. R. Hansen, M. J. Ratain, and L. L. Siu, “Tumour heterogeneity in the clinic,” *Nature*, vol. 501, no. 7467, 2013, pp. 355-364.

[63] T. Turki, and Z. Wei, “Learning approaches to improve prediction of drug sensitivity in breast cancer patients,” in the Proceedings of the 38th IEEE Annual International Conference of the Engineering in Medicine and Biology Society (EMBC), 2016, pp. 3314-3320.

[64] T. Turki, and Z. Wei, “A Noise-Filtering Approach for Cancer Drug Sensitivity Prediction,” in the Proceedings of the Neural Information Processing Systems Workshop on Machine Learning for Health (NIPS ML4HC), Barcelona, Spain, 2016.

[65] W. Du, and O. Elemento, “Cancer systems biology: embracing complexity to develop better anticancer therapeutic strategies,” *Oncogene*, vol. 34, no. 25, 2015, pp. 3215-3225.

[66] G. Riddick, H. Song, S. Ahn, J. Walling, D. Borges-Rivera, W. Zhang, and H. A. Fine, “Predicting in vitro drug sensitivity using Random Forests,” *Bioinformatics*, vol. 27, no. 2, 2011, pp. 220-224.

[67] S. J. Pan, and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, 2010, pp. 1345-1359.

[68] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big Data*, vol. 3, no. 1, 2016, pp. 1-40.

[69] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki, “Link prediction using supervised learning,” in the Proceedings of Society for Industerial and Appliem Mathematics (SIAM) Data Mining Workshop on Link Analysis, Counterterrorism and Security, 2006.

[70] M. Fire, L. Tenenboim, O. Lesser, R. Puzis, L. Rokach, and Y. Elovici, “Link prediction in social networks using computationally efficient topological features,” in the Proceedings of IEEE Third International Conference on Privacy, Security, Risk and Trust and IEEE Third International Conference on Social Computing 2011, pp. 73-80.

[71] E. D. Kolaczyk, "Statistical Analysis of Network Data: Methods and Models," New York City, NY: Springer, 2009, p. 386.

[72] F. W. Takes, and W. A. Kosters, “Computing the eccentricity distribution of large graphs,” *Algorithms,* vol. 6, no. 1, 2013, pp. 100-118.

[73] J. Ye, H. Cheng, Z. Zhu, and M. Chen, “Predicting positive and negative links in signed social networks by transfer learning,” in the Proceedings of the 22nd international conference on World Wide Web, 2013, pp. 1477-1488.

[74] M. Newman, "Networks: an introduction," New York City, NY: Oxford University Press, 2010.

[75] M. E. Newman, “The mathematics of networks,” *The New Palgrave Encyclopedia of Economics,* vol. 2, no. 2008, 2008, pp. 1-12.

[76] N. Japkowicz, and M. Shah, "Evaluating learning algorithms: a classification perspective," New York City, NY: Cambridge University Press, 2011.

[77] T. Schaffter, D. Marbach, and D. Floreano, “GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods,” *Bioinformatics,* vol. 27, no. 16, 2011, pp. 2263-2270.

[78] A. Liaw, and M. Wiener, “Classification and regression by randomForest,” *R News,* vol. 3, no. 3, 2002, pp. 18-22.

[79] F. Günther, and S. Fritsch, “neuralnet: Training of neural networks,” *The R Journal,* vol. 2, no. 1, 2010, pp. 30-38.

[80] C.-C. Chang, and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 2, no. 3, 2011, pp. 27.

[81] G. Csardi, and T. Nepusz, “The igraph software package for complex network research,” *InterJournal, Complex Systems,* vol. 1695, no. 5, 2006, pp. 1-9.

[82] I. Guyon, A. R. S. A. Alamdari, G. Dror, and J. M. Buhmann, “Performance prediction challenge,” in the Proceedings of the IEEE International Joint Conference on Neural Network, 2006, pp. 1649-1656.

[83] G. K. Kanji, "100 statistical tests," London, UK: Sage, 2006.

[84] P. A. Nuin, Z. Wang, and E. R. Tillier, “The accuracy of several multiple sequence alignment programs for proteins,” *BioMed Central (BMC) Bioinformatics,* vol. 7, no. 1, 2006, pp. 1.

[85] J. D. Thompson, F. Plewniak, and O. Poch, “A comprehensive comparison of multiple sequence alignment programs,” *Nucleic Acids Research,* vol. 27, no. 13, 1999, pp. 2682-2690.

[86] H. Chen, W.-S. Ku, H. Wang, L. Tang, and M.-T. Sun, “LinkProbe: Probabilistic inference on large-scale social networks,” in the Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE), 2013, pp. 290-301.

[87] T. Turki, and U. Roshan, “Weighted maximum variance dimensionality reduction,” in the Proceedings of the 6th Mexican Conference on Pattern Recognition, 2014, pp. 11-20.

[88] J. T. Wang, J. Liu, and J. Wang, “XML clustering and retrieval through principal component analysis,” *International Journal on Artificial Intelligence Tools,* vol. 14, no. 04, 2005, pp. 683-699.

[89] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano, “Generating realistic in silico gene networks for performance assessment of reverse engineering methods,” *Journal of Computational Biology,* vol. 16, no. 2, 2009, pp. 229-239.

[90] R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky, “Towards a rigorous assessment of systems biology models: the DREAM3 challenges,” *Public Library of Science (PloS) One,* vol. 5, no. 2, 2010, pp. e9202.

[91] T. Hothorn, and B. S. Everitt, *A handbook of statistical analyses using R*, Boca Raton, FL: CRC press, 2014.

[92] P. Zoppoli, S. Morganella, and M. Ceccarelli, "TimeDelay-ARACNE: reverse engineering of gene networks from time-course data by an information theoretic approach," *BioMed Central (BMC) Bioinformatics,* vol. 11, 2010, pp. 154.

[93] A. Greenfield, A. Madar, H. Ostrer, and R. Bonneau, "DREAM4: combining genetic and dynamic information to identify biological networks and dynamical models," *Public Library of Science (PloS) One*, vol. 5, no. 10, 2010, pp. e13397.

[94] A. Madar, A. Greenfield, E. Vanden-Eijnden, and R. Bonneau, "DREAM3: network inference using dynamic context likelihood of relatedness and the inferelator," *Public Library of Science (PloS) One*, vol. 5, no. 3, 2010, pp. e9803.

[95] V. A. Huynh-Thu, and G. Sanguinetti, "Combining tree-based and dynamical systems for the inference of gene regulatory networks," *Bioinformatics,* vol. 31, no. 10, 2015, pp. 1614-1622.

[96] F. Mordelet, and J. P. Vert, "SIRENE: supervised inference of regulatory networks," *Bioinformatics,* vol. 24, no. 16, 2008, pp. i76-i82.

[97] J. T. L. Wang, M. J. Zaki, H. Toivonen, and D. Shasha, "Data Mining in Bioinformatics," J. T. L. Wang, Zaki, M.J., Toivonen, H., and Shasha, D., ed., London, UK: Springer-Verlag London, 2005.

[98] B. Schölkopf, and A. J. Smola, "Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond," Cambridge, MA: MIT Press, 2001.

[99] R. A. Horn, and C. R. Johnson, "Matrix Analysis," Cambridge, UK: Cambridge University Press, 2012.

[100] G. H. Golub, and C. F. Van Loan, "Matrix Computations," Baltimore, MD: Johns Hopkins University Press, 1996.

[101] Y. Saad, "Numerical Methods for Large Eigenvalue Problems," Philadelphia, PA: SIAM, 2011.

[102] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, “Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion,” *Journal of Machine Learning Research,* vol. 11, 2010, pp. 3371-3408.

[103] X. Rong, "deepnet: deep learning toolkit in R," *R package*, 2014.

[104] C. C. Chang, and C. J. Lin, “LIBSVM: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology,* vol. 2, no. 3, 2011, pp. 27.

[105] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis, “kernlab - an S4 package for kernel methods in R,” *Journal of Statistical Software,* vol. 11, no. 9, 2004, pp. 1-20.

[106] L. Gondara, “Random forest with random projection to impute missing gene expression data,” in the Proceedings of the 14th IEEE International Conference on Machine Learning and Applications, 2015, pp. 1251-1256.

[107] I. Chebil, M. Elati, C. Rouveirol, and G. Santini, “SetNet: ensemble method techniques for learning regulatory networks,” in the Proceedings of the 12th IEEE International Conference on Machine Learning and Applications, 2013, pp. 34-39.

[108] B. Ouyang, L. Jiang, and Z. Teng, “A noise-filtering method for link prediction in complex networks,” *Public Library of Science (PloS) One,* vol. 11, no. 1, 2016, pp. e0146925.

[109] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical Recipes: The Art of Scientific Computing, Third Edition," New York City, NY: Cambridge University Press, 2007.

[110] T. Turki, W. Bassett, and J. T. L. Wang, “A Learning Framework to Improve Unsupervised Gene Network Inference,” in the Proceedings of the 12th International Conference on Machine Learning and Data Mining in Pattern Recognition Machine Learning (MLDM), New York, NY, 2016, pp. 28-42.

[111] H. He, and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 21, no. 9, 2009, pp. 1263-1284.

[112] H. He, and Y. Ma, "Imbalanced learning: foundations, algorithms, and applications," Hoboken, NJ: John Wiley & Sons, 2013.

[113] X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory undersampling for class-imbalance learning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 39, no. 2, 2009, pp. 539-550.

[114] G. E. Batista, R. C. Prati, and M. C. Monard, “A study of the behavior of several methods for balancing machine learning training data,” *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations Newsletter,* vol. 6, no. 1, 2004, pp. 20-29.

[115] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, “Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem,” in the Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2009, pp. 475-482.

[116] Y. Freund, and R. E. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” in the Proceedings of the Second European Conference on Computational Learning Theory, 1995, pp. 23-37.

[117] T. Turki, M. Ihsan, N. Turki, J. Zhang, U. Roshan, and Z. Wei, “Top-k Parametrized Boost,” in the Proceedings of the Second International Conference on Mining Intelligence and Knowledge Exploration, 2014, pp. 91-98.

[118] L. Zhong, J. T. Wang, D. Wen, V. Aris, P. Soteropoulos, and B. A. Shapiro, “Effective classification of microRNA precursors using feature mining and AdaBoost algorithms,” *OMICS: A Journal of Integrative Biology,* vol. 17, no. 9, 2013, pp. 486-493.

[119] L. Zhong, J. T. Wang, D. Wen, and B. A. Shapiro, “Pre-miRNA classification via combinatorial feature mining and boosting,” in the Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2012, pp. 1-4.

[120] H. Alhammady, and K. Ramamohanarao, “Using emerging patterns to construct weighted decision trees,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 18, no. 7, 2006, pp. 865-876.

[121] L. Breiman, “Random Forests,” *Machine Learning,* vol. 45, no. 1, 2001, pp. 5-32.

[122] T. Turki, and Z. Wei, “IPRed: Instance Reduction Algorithm Based on the Percentile of the Partitions,” in the Proceedings of the 26th Modern AI and Cognitive Science Conference, 2015, pp. 181-185.

[123] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, 2002, pp. 321-357.

[124] Y. Li, M. Liang, and Z. Zhang, “Regression Analysis of Combined Gene Expression Regulation in Acute Myeloid Leukemia,” *Public Library of Science (PloS) Computational Biology,* vol. 10, no. 10, 2014, pp. e1003908.

[125] H. Kim, and M. Bredel, “Feature selection and survival modeling in The Cancer Genome Atlas,” *International Journal of Nanomedicine,* vol. 8, no. Suppl 1, 2013, pp. 57-62.

[126] J. Friedman, T. Hastie, and R. Tibshirani, “Regularization paths for generalized linear models via coordinate descent,” *Journal of Statistical Software,* vol. 33, no. 1, 2010, pp. 1.

[127] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, “Pathwise coordinate optimization,” *The Annals of Applied Statistics,* vol. 1, no. 2, 2007, pp. 302-332.

[128] M. Wozniak, M. Graña, and E. Corchado, “A survey of multiple classifier systems as hybrid systems,” *Information Fusion,* vol. 16, 2014, pp. 3-17.

[129] H. Joensuu, P.-L. Kellokumpu-Lehtinen, P. Bono, T. Alanko, V. Kataja, R. Asola, T. Utriainen, R. Kokko, A. Hemminki, M. Tarkkanen, T. Turpeenniemi-Hujanen, S. Jyrkkiö, M. Flander, L. Helle, S. Ingalsuo, K. Johansson, A.-S. Jääskeläinen, M. Pajunen, M. Rauhala, J. Kaleva-Kerola, T. Salminen, M. Leinonen, I. Elomaa, and J. Isola, “Adjuvant Docetaxel or Vinorelbine with or without Trastuzumab for Breast Cancer,” *New England Journal of Medicine,* vol. 354, no. 8, 2006, pp. 809-820.

[130] M. Aujla, “Chemotherapy: Treating older breast cancer patients,” *Nature Reviews Clinical Oncology*, vol. 6, no. 6, 2009, pp. 302-302.

[131] A. Brazma, H. Parkinson, U. Sarkans, M. Shojatalab, J. Vilo, N. Abeygunawardena, E. Holloway, M. Kapushesky, P. Kemmeren, G. G. Lara, A. Oezcimen, P. Rocca-Serra, and S.-A. Sansone, “ArrayExpress—a public repository for microarray gene expression data at the EBI,” *Nucleic Acids Research*, vol. 31, no. 1, 2003, pp. 68-71.

[132] R. Edgar, M. Domrachev, and A. E. Lash, “Gene Expression Omnibus: NCBI gene expression and hybridization array data repository,” *Nucleic Acids Research*, vol. 30, no. 1, 2002, pp. 207-210.

[133] J. C. Chang, E. C. Wooten, A. Tsimelzon, S. G. Hilsenbeck, M. C. Gutierrez, R. Elledge, S. Mohsin, C. K. Osborne, G. C. Chamness, D. C. Allred, and P. O'Connell, “Gene expression profiling for the prediction of therapeutic response to docetaxel in patients with breast cancer,” *The Lancet*, vol. 362, no. 9381, pp. 362-369.

[134] J. C. Chang, E. C. Wooten, A. Tsimelzon, S. G. Hilsenbeck, M. C. Gutierrez, Y.-L. Tham, M. Kalidas, R. Elledge, S. Mohsin, C. K. Osborne, G. C. Chamness, D. C. Allred, M. T. Lewis, H. Wong, and P. O'Connell, “Patterns of Resistance and Incomplete Response to Docetaxel by Gene Expression Profiling in Breast Cancer Patients,” *Journal of Clinical Oncology*, vol. 23, no. 6, 2005, pp. 1169-1177.

[135] J. Friedman, T. Hastie, and R. Tibshirani, "glmnet: Lasso and elastic-net regularized generalized linear models. R package version 1.9–5," R Foundation for Statistical Computing Vienna, 2013.

[136] O. Bousquet, and A. Elisseeff, “Stability and generalization,” *The Journal of Machine Learning Research*, vol. 2, 2002, pp. 499-526.

[137] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi, “General conditions for predictivity in learning theory,” *Nature*, vol. 428, no. 6981, 2004, pp. 419-422.

[138] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church, “Systematic determination of genetic network architecture,” *Nature Genetics*, vol. 22, no. 3, 1999, pp. 281-285.

[139] A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, and I. S. Kohane, “Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks,” *Proceedings of the National Academy of Sciences*, vol. 97, no. 22, 2000, pp. 12182-12186.

[140] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner, “Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles,” *Public Library of Science (PloS) Biology,* vol. 5, no. 1, 2007, pp. e8.

[141] I. S. Jang, E. C. Neto, J. Guinney, S. H. Friend, and A. A. Margolin, “Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data,” in the Proceedings of the Pacific Symposium on Biocomputing, 2014, pp. 63.

[142] S. Falgreen, K. Dybkær, K. H. Young, Z. Y. Xu-Monette, T. C. El-Galaly, M. B. Laursen, J. S. Bødker, M. K. Kjeldsen, A. Schmitz, and M. Nyegaard, “Predicting response to multidrug regimens in cancer patients using cell line experiments and regularised regression models,” *BioMed Central (BMC) Cancer*, vol. 15, no. 1, 2015, pp. 235.

[143] N. Chiluka, N. Andrade, and J. Pouwelse, "A link prediction approach to recommendations in large-scale user-generated content systems," *Advances in Information Retrieval*: Springer, 2011, pp. 189-200.

[144] T. Turki, and Z. Wei, “A greedy-based oversampling approach to improve the prediction of mortality in MERS patients,” in the Proceedings of the Annual IEEE Systems Conference (SysCon), 2016, pp. 1-5.

[145] B. Settles, “Active learning literature survey,” *University of Wisconsin, Madison,* vol. 52, no. 55-66, 2010, pp. 11.

[146] P. Melville, and R. J. Mooney, “Diverse ensembles for active learning,” in the Proceedings of the Twenty-First International Conference on Machine Learning, 2004, pp. 74.

[147] R. Gilad-Bachrach, A. Navot, and N. Tishby, “Query by committee made real,” in the Proceedings of the Advances in Neural Information Processing Systems, 2005, pp. 443-450.

[148] A. Krogh, and J. Vedelsby, “Neural network ensembles, cross validation, and active learning,” in the Proceedings of the Advances in Neural Information Processing Systems, 1995, pp. 231-238.

[149] M. W. Mahoney, and P. Drineas, “CUR matrix decompositions for improved data analysis,” *in the Proceedings of the National Academy of Sciences,* vol. 106, no. 3, 2009, pp. 697-702.

[150] A. Bodor, I. Csabai, M. W. Mahoney, and N. Solymosi, “rCUR: an R package for CUR matrix decomposition,” *BioMed Central (BMC) Bioinformatics,* vol. 13, 2012, pp. 103.

[151] K. Neubert, S. Meister, K. Moser, F. Weisel, D. Maseda, K. Amann, C. Wiethe, T. H. Winkler, J. R. Kalden, R. A. Manz, and R. E. Voll, “The proteasome inhibitor bortezomib depletes plasma cells and protects mice with lupus-like disease from nephritis,” *Nature Medicine,* vol. 14, no. 7, 2008, pp. 748-755.

[152] A. Paramore, and S. Frantz, “Bortezomib,” *Nature Reviews Drug Discovery,* vol. 2, no. 8, 2003, pp. 611-612.

[153] G. Mulligan, C. Mitsiades, B. Bryant, F. Zhan, W. J. Chng, S. Roels, E. Koenig, A. Fergus, Y. Huang, P. Richardson, W. L. Trepicchio, A. Broyl, P. Sonneveld, J. D. Shaughnessy, P. Leif Bergsagel, D. Schenkein, D.-L. Esseltine, A. Boral, and K. C. Anderson, “Gene expression profiling and correlation with outcome in clinical trials of the proteasome inhibitor bortezomib,” *Blood,* vol. 109, no. 8, 2007, pp. 3177-3188.

[154] P. Bermolen, and D. Rossi, “Support vector regression for link load prediction,” *Computer Networks,* vol. 53, no. 2, 2009, pp. 191-201.

[155] Z. l. Wu, C. h. Li, J. K. y. Ng, and K. R. p. h. Leung, “Location Estimation via Support Vector Regression,” *IEEE Transactions on Mobile Computing,* vol. 6, no. 3, 2007, pp. 311-321.

[156] J. Balfer, and J. Bajorath, “Systematic Artifacts in Support Vector Regression-Based Compound Potency Prediction Revealed by Statistical and Activity Landscape Analysis,” *Public Library of Science (PloS) One,* vol. 10, no. 3, 2015, pp. e0119301.

[157] I. Jolliffe, "Principal component analysis," New York City, NY: Wiley Online Library, 2002.

[158] D. Witten, R. Tibshirani, S. Gross, and B. Narasimhan, "PMA: Penalized Multivariate Analysis," *R Package,* 2011.

[159] D. M. Witten, R. Tibshirani, and T. Hastie, “A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis,” *Biostatistics*, 2009, pp. 515-534.

[160] C. D. Sigg, and J. M. Buhmann, “Expectation-maximization for sparse and non-negative PCA,” in the Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 960-967.

[161] C. Sigg, and M. C. Sigg, "nsprcomp," *R Package*, 2013.

[162] C. D. Manning, P. Raghavan, and H. Schütze, "Introduction to Information Retrieval," New York City, NY: Cambridge University Press, 2008, p. 496.

[163] R. Batuwita, and V. Palade, “microPred: effective classification of pre-miRNAs for human miRNA gene prediction,” *Bioinformatics,* vol. 25, no. 8, 2009, pp. 989-995.

[164] Y. B. Marques, A. de Paiva Oliveira, A. T. R. Vasconcelos, and F. R. Cerqueira, “Mirnacle: machine learning with SMOTE and random forest for improving selectivity in pre-miRNA ab initio prediction,” *BioMed Central (BMC) Bioinformatics,* vol. 17, no. 18, 2016, pp. 53.

[165] M. Nakamura, Y. Kajiwara, A. Otsuka, and H. Kimura, “Lvq-smote–learning vector quantization based synthetic minority over–sampling technique for biomedical data,” *BioData Mining,* vol. 6, no. 1, 2013, pp. 16.

[166] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, “Relative-error CUR matrix decompositions,” *SIAM Journal on Matrix Analysis and Applications,* vol. 30, no. 2, 2008, pp. 844-881.

[167] H. Drucker, “Improving Regressors using Boosting Techniques,” in the Proceedings of the Fourteenth International Conference on Machine Learning 1997, pp. 107-115.

[168] L. Venkova, A. Aliper, M. Suntsova, R. Kholodenko, D. Shepelin, N. Borisov, G. Malakhova, R. Vasilov, S. Roumiantsev, A. Zhavoronkov, and A. Buzdin, “Combinatorial high-throughput experimental and bioinformatic approach identifies molecular pathways linked with the sensitivity to anticancer target drugs,” *Oncotarget,* vol. 6, no. 29, 2015, pp. 27227-27238.

[169] M. J. Garnett, E. J. Edelman, S. J. Heidorn, C. D. Greenman, A. Dastur, K. W. Lau, P. Greninger, I. R. Thompson, X. Luo, J. Soares, Q. Liu, F. Iorio, D. Surdez, L. Chen, R. J. Milano, G. R. Bignell, A. T. Tam, H. Davies, J. A. Stevenson, S. Barthorpe, S. R. Lutz, F. Kogera, K. Lawrence, A. McLaren-Douglas, X. Mitropoulos, T. Mironenko, H. Thi, L. Richardson, W. Zhou, F. Jewitt, T. Zhang, P. O/'Brien, J. L. Boisvert, S. Price, W. Hur, W. Yang, X. Deng, A. Butler, H. G. Choi, J. W. Chang, J. Baselga, I. Stamenkovic, J. A. Engelman, S. V. Sharma, O. Delattre, J. Saez-Rodriguez, N. S. Gray, J. Settleman, P. A. Futreal, D. A. Haber, M. R. Stratton, S. Ramaswamy, U. McDermott, and C. H. Benes, "Systematic identification of genomic markers of drug sensitivity in cancer cells," *Nature,* vol. 483, no. 7391, 2012, pp. 570-575.

[170] H. Joensuu, P.-L. Kellokumpu-Lehtinen, P. Bono, T. Alanko, V. Kataja, R. Asola, T. Utriainen, R. Kokko, A. Hemminki, and M. Tarkkanen, "Adjuvant docetaxel or vinorelbine with or without trastuzumab for breast cancer," *New England Journal of Medicine,* vol. 354, no. 8, 2006, pp. 809-820.

[171] L. C. Wienkers, and T. G. Heath, "Predicting in vivo drug interactions from in vitro drug discovery data," *Nature reviews Drug discovery,* vol. 4, no. 10, 2005, pp. 825-833.

[172] F. Petralia, P. Wang, J. Yang, and Z. Tu, "Integrative random forest for gene regulatory network inference," *Bioinformatics,* vol. 31, no. 12, 2015, pp. i197-i205.

[173] Y. Abduallah, T. Turki, K. Byron, Z. Du, M. Cervantes-Cervantes, and J. T. L. Wang, "MapReduce Algorithms for Inferring Gene Regulatory Networks from Time-Series Microarray Data Using an Information-Theoretic Approach," *BioMed Research International,* vol. 2017, 2017, pp. 8.

[174] T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer, "ROCR: visualizing classifier performance in R," *Bioinformatics,* vol. 21, no. 20, 2005, pp. 3940-3941.

[175] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*: Pearson, 2005.

[176] P. Nowak-Sliwinska, A. Weiss, X. Ding, P. J. Dyson, H. Van Den Bergh, A. W. Griffioen, and C.-M. Ho, "Optimization of drug combinations using feedback system control," *Nature Protocols,* vol. 11, no. 2, 2016, pp. 302-315.

[177] M. Marty, F. Cognetti, D. Maraninchi, R. Snyder, L. Mauriac, M. Tubiana-Hulin, S. Chan, D. Grimes, A. Antón, and A. Lluch, "Randomized phase II trial of the efficacy and safety of trastuzumab combined with docetaxel in patients with human epidermal growth factor receptor 2–positive metastatic breast cancer administered as first-line treatment: The M77001 study group," *Journal of Clinical Oncology*, vol. 23, no. 19, 2005, pp. 4265-4274.

[178] K. F. Chen, C. Y. Liu, Y. C. Lin, H. C. Yu, T. H. Liu, D. R. Hou, P. J. Chen, and A. L. Cheng, "CIP2A mediates effects of bortezomib on phospho-Akt and apoptosis in hepatocellular carcinoma cells," *Oncogene*, vol. 29, 2010.

[179] A. Awada, J. Albanell, P. A. Canney, L. Y. Dirix, T. Gil, F. Cardoso, P. Gascon, M. J. Piccart, and J. Baselga, "Bortezomib/docetaxel combination therapy in patients with anthracycline-pretreated advanced/metastatic breast cancer: a phase I/II dose-escalation study," *British Journal of Cancer*, vol. 98, 2008.

[180] C. Li, R. Li, J. R. Grandis, and D. E. Johnson, "Bortezomib induces apoptosis via Bim and Bik up-regulation and synergizes with cisplatin in the killing of head and neck squamous cell carcinoma cells," *Molecular Cancer Therapeutics*, vol. 7, no. 6, 2008, pp. 1647-1655.

[181] T. J. Lynch, D. Fenton, V. Hirsh, D. Bodkin, E. L. Middleman, A. Chiappori, B. Halmos, R. Favis, H. Liu, and W. L. Trepicchio, "A randomized phase 2 study of erlotinib alone and in combination with bortezomib in previously treated advanced non-small cell lung cancer," *Journal of Thoracic Oncology*, vol. 4, no. 8, 2009, pp. 1002-1009.

[182] B. Recht, "A simpler approach to matrix completion," *Journal of Machine Learning Research*, vol. 12, 2011, pp. 3413-3430.

[183] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," *Journal of Machine Learning Research*, vol. 11, 2010, pp. 2057-2078.

[184] E. Eisenhauer, P. Therasse, J. Bogaerts, L. Schwartz, D. Sargent, R. Ford, J. Dancey, S. Arbuck, S. Gwyther, and M. Mooney, "New response evaluation criteria in solid tumours: revised RECIST guideline (version 1.1)," *European Journal of Cancer*, vol. 45, no. 2, 2009, pp. 228-247.

[185] A. H. N. Kamdje, P. F. S. Etet, L. Vecchio, J. M. Muller, M. Krampera, and K. E. Lukong, "Signaling pathways in breast cancer: therapeutic targeting of the microenvironment," *Cellular Signalling*, vol. 26, no. 12, 2014, pp. 2843-2856.

[186] M. Kubo, M. Nakamura, A. Tasaki, N. Yamanaka, H. Nakashima, M. Nomura, S. Kuroki, and M. Katano, “Hedgehog signaling pathway is a new therapeutic target for patients with breast cancer,” *Cancer Research*, vol. 64, no. 17, 2004, pp. 6071-6074.

[187] M. Kanehisa, and S. Goto, “KEGG: kyoto encyclopedia of genes and genomes,” *Nucleic Acids Research*, vol. 28, no. 1, 2000, pp. 27-30.

[188] L. Chen, Q. Li, T. She, H. Li, Y. Yue, S. Gao, T. Yan, S. Liu, J. Ma, and Y. Wang, “IRE1α-XBP1 signaling pathway, a potential therapeutic target in multiple myeloma,” *Leukemia Research*, vol. 49, 2016, pp. 7-12.