

2020

Similarity-principle-based machine learning method for clinical trials and beyond

<https://hdl.handle.net/2144/41983>

"Downloaded from OpenBU. Boston University's institutional repository."

BOSTON UNIVERSITY
GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

**SIMILARITY-PRINCIPLE-BASED MACHINE LEARNING METHOD
FOR CLINICAL TRIALS AND BEYOND**

by

SUSAN HWANG

B.A., North Carolina State University, 2009
M.A., Boston University, 2013

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2020

© 2020 by
SUSAN HWANG
All rights reserved

Approved by

First Reader

Mark Yang Chang, Ph.D.
Adjunct Professor of Biostatistics

Second Reader

Robert A. Lew, Ph.D.
Associate Professor of Biostatistics

Third Reader

Michael P. LaValley, Ph.D.
Professor of Biostatistics

DEDICATION

I would like to dedicate this dissertation to my beloved grandmother BongSoon Choi who raised me and passed away when I was in middle school, my patient parents Sue Hwang and YoungKun Hwang and my one and only brother JunBum Hwang who moved back to Korea forever. I would also like to dedicate this work to my dearest dog DongE (a.k.a. Snowball Snowy) who moved to the US from Korea at the age of 14 and passed away at the age of 20 (and spent the last 6 years in Boston with me while I struggled through the MA/PhD program), my amazing first and last foster dogs Reyou and JeeSun, my other wonderful foster dogs who found loving forever families in the past couple of years (Sena/Nalu, Suri, Yurim/Cooper, Moon/Jun, Ayi/Parker, Maho, POCO), and those people who are actively volunteering and fighting to help for the voiceless.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest sincere gratitude to my advisor, Mark Chang, for his unwavering support, priceless advice as a researcher and long-lived human, immense knowledge in not only biostatistics but also in philosophy and other fields, and incredible patience throughout my whole journey as a Ph.D. student. He has been a steady source of support, motivation and encouragement and has helped shape me into the person, researcher, and statistician I am today. It is difficult to put into words just how much influence he has had on my life these past several years during my time at BU. I could not have had a better mentor and his passion in research has inspired me.

Also, a very special thank you to my dissertation committee, Bob Lew, Mike LaValley and Bingming Yi, for their valuable feedback and comments. This work would not have been possible without their support, guidance and encouragement, and I'm so grateful to have them on my committee and can't thank enough for their expertise and precious time throughout this journey.

I would also like to express my sincere gratitude to Howard Cabral, co-director of the Biostatistics Program, for his endless support, advice and encouragement in both my academic and career interests since I entered the MA/PhD program. As a hard-working student but a non-native English speaker who did not have a strong background in statistics, there were many times that I felt unsure. But by always saying "Susan's here" and smiling, I felt I belonged to the department even if I struggled.

Thank you to my academic advisor, Lisa Sullivan, and professors in the Biostatistics Department, especially Gheorghe Doros for his encouragement,

dedication, passion and patience in teaching in both class and regular office visits for questions. I feel truly grateful and privileged to have been a student and met very passionate and caring professors in this department.

Special thanks to my friend Han Chen, who generously offered to patiently tutor me, helped me prepare and pass the qualifying exams (while I was working full-time) and encouraged me through the most difficult times. Without his help, I would not have passed the qualifiers, and this dissertation journey would not have begun.

Thank you to my wonderful talented colleagues and friends: Jing Wang, Baiyun Yao, Flora Xue, Jisun Jang, Zhaoyang Teng, Wei Gao, Yaonan Zhang, Yansong Cheng, Joseph Wu, Siyan Xu, Revathi Ananthakrishnan, Aya Mitani, Danielle Enserro, Jacqueline Milton, Carlee Moser, Sarah Leatherman. I cannot thank them enough for their friendship, support and great memories.

Special thanks, as well, to Bo Yang, Ouhong Wang, Bingming Yi for the externship opportunity, encouragement, valuable advice / supports, and for allowing me to gain hands-on experience as a research extern for an academic year on the analysis of clinical trials for a rare disease, Cystic Fibrosis. Also thank you to my Vertex colleagues for their friendship, support and great memories.

Thanks to Anjana Boss, Bernard Silverman, Yangchun Du, Ying Jiang, Chih-Chin Liu, Jianjun Liu, and my other Alkermes colleagues for their encouragement, support, advice and for making my internship and 3 years of career as a full-time junior biostatistician joyful.

Special thanks also due to my academic advisor and statistics professor from North Carolina State University, Jeff Thompson (a.k.a. Dr. T), for his encouragement, advice, support, dedication, passion and patience in teaching. Without his support / time during regular office visits and wonderful recommendation letters, this long journey would never have been this far.

Also, many thanks to my study buddies and life-time friends from NC State for being my rocks, especially to my best friend and my person at NC State, YoungMi Kang, who moved back to Korea forever, and to YoungJin Park, who has helped shape me into the person, researcher and activist I am today.

Thanks to Maggie Boyd, a writing assistant from the Educational Resource Center program at BU, for her help in reviewing a partial draft of my dissertation and correcting some grammatical errors in such a limited time.

Special thanks to my parents, Sue Hwang and YoungKun Hwang, my aunts, Cindy Tak Skinner and ChinSuk Aylward, and my uncle Robert Aylward for always listening when I need a shoulder to cry on and for always supporting me and for teaching me the value of hard work, determination, and humility throughout my life, who unconditionally love, encourage, support and believe in me. I would also like to thank my family and friends, especially HyungGyun Lee, for being my rock these past few years.

Last but not least, by grace of God, I have met these wonderful people in my life and am here today still on the journey, continuing the search of my passion, hoping to make a small contribution to make a better world.

**SIMILARITY-PRINCIPLE-BASED MACHINE LEARNING METHOD
FOR CLINICAL TRIALS AND BEYOND**

SUSAN HWANG

Boston University Graduate School of Arts and Sciences, 2020

Major Professor: Mark Chang, PhD, Adjunct Professor of Biostatistics

ABSTRACT

The control of type-I error is a focal point for clinical trials. On the other hand, it is also critical to be able to detect a truly efficacious treatment in a clinical trial. With recent success in *supervised learning* (classification and regression problems), artificial intelligence (AI) and machine learning (ML) can play a vital role in identifying efficacious new treatments. However, the high performance of the AI methods, particularly the deep learning neural networks, requires a much larger dataset than those we commonly see in clinical trials. It is desirable to develop a new ML method that performs well with a small sample size (ranges from 20 to 200) and has advantages as compared with the classic statistical models and some of the most relevant ML methods.

In this dissertation, we propose a Similarity-Principle-Based Machine Learning (SBML) method based on the *similarity principle* assuming that identical or similar subjects should behave in a similar manner. SBML method introduces the attribute-scaling factors at the training stage so that the relative importance of different attributes can be objectively determined in the similarity measures. In addition, the gradient method is used in learning / training in order to update the attribute-scaling factors. The method is novel as far as we know.

We first evaluate SBML for continuous outcomes, especially when the sample size is small, and investigate the effects of various tuning parameters on the performance of SBML. Simulations show that SBML achieves better predictions in terms of mean squared errors or misclassification error rates for various situations under consideration than conventional statistical methods, such as full linear models, optimal or ridge regressions and mixed effect models, as well as ML methods including kernel and decision tree methods.

We also extend and show how SBML can be flexibly applied to binary outcomes. Through numerical and simulation studies, we confirm that SBML performs well compared to classical statistical methods, even when the sample size is small and in the presence of unmeasured predictors and/or noise variables.

Although SBML performs well with small sample sizes, it may not be computationally efficient for large sample sizes. Therefore, we propose Recursive SBML (RSBML), which can save computing time, with some tradeoffs for accuracy. In this sense, RSBML can also be viewed as a combination of unsupervised learning (dimension reduction) and supervised learning (prediction). Recursive learning resembles the natural human way of learning. It is an efficient way of learning from complicated large data. Based on the simulation results, RSBML performs much faster than SBML with reasonable accuracy for large sample sizes.

PREFACE

Artificial intelligence (AI) takes many forms: data mining, text mining, machine learning (ML), pattern recognition, statistical learning, deep learning, bioinformatics, computational biology, computational linguistics, natural language processing and robotics. Achievements in many fields, especially in deep learning for image and language processing and voice recognition seem unrelated to conventional statistics, even though both are heavily dependent on big data. Some AI studies are statistically oriented under the names of statistical learning, pattern recognition and ML. Because traditional statistics and ML both deal with data, and an increasing number of statisticians or data scientists have become involved in ML research, the meaning and scope of ML have been constantly evolving.

ML generally differs from statistical inference. Differences include: (1) ML emphasizes learning and prediction, while classical statistics focuses on hypothesis testing and estimation, (2) ML and classical statistics both deal with uncertainty, but the former is mainly algorithm-based, while the latter focuses on probability distributions, and (3) ML either simulates or gathers real world experiences/evidence, while classical statistics models the data assuming the data follow mathematically tractable probability distributions.

In ML literature, the *inputs*, *attributes* or *features* are preferred to indicate what statistical literature calls the *predictors*, *covariates*, or more classically *independent variables*. ML literature calls the *outputs* or *labels* what statistical literature calls the

outcomes, endpoints, responses or classically *dependent variables*. In this work, we will use the terms interchangeably, but we will primarily use “*predictors*” and “*outcomes*.”

This 5-chapter dissertation involves mainly three closely related research topics: Similarity-Principle-Based Machine Learning with Continuous Outcomes, Similarity-Principle-Based Machine Learning with Binary Outcomes and Recursive / Hierarchical Similarity-Principle-Based Machine Learning.

Chapter 1 provides an overview of general approaches to machine learning methods, including a general framework, concepts and types of learning. In particular, we focus more on supervised learning methods such as kernel methods, *support vector machines* and tree methods. In Chapter 2 we evaluate the performance of SBML for commonly encountered continuous outcomes and determine the basic ranges of model tuning parameters. We then expand the evaluation of SBML against other comparable methods to binary outcomes in Chapter 3, as binary outcomes are also common in clinical trials and medical fields. In Chapter 4 we expand the work to recursive or hierarchical SBML, where we use group-level data, instead of individual-level data. The recursive approach is to mimic human learning in the case of big data and to improve computational efficiency. Finally, we conclude with Chapter 5 which provides general recommendations and a summary of key findings in this work. All the necessary R code for both SBML and RSBML are provided in the Appendix.

TABLE OF CONTENTS

DEDICATION	iv
ACKNOWLEDGMENTS	v
ABSTRACT	viii
PREFACE	x
TABLE OF CONTENTS	xii
LIST OF TABLES	xvi
LIST OF FIGURES	xviii
LIST OF ABBREVIATIONS	xix
1. INTRODUCTION	1
1.1 Motivation	1
1.2 Machine Learning	3
1.3 Types of Supervised Machine Learning Methods	6
1.3.1 Kernel Methods	6
1.3.2 Support Vector Machine	7
1.3.3 Decision Tree Methods	7
1.3.4 Logistic Regression	9
1.4 Role of Similarity Principle in Scientific Discovery	10
1.5 General Simulation Settings	11

2. SIMILARITY-PRINCIPLE-BASED MACHINE LEARNING WITH	
CONTINUOUS OUTCOMES	13
2.1 Introduction.....	13
2.2 Methodology.....	13
2.2.1 Similarity Weighting.....	13
2.2.2 Data Normalization.....	16
2.2.3 Notations.....	17
2.2.4 Algorithms.....	17
2.3 Application to Rare Disease Clinical Trials.....	23
2.3.1 Cystic Fibrosis – Rare Disease.....	23
2.3.2 General Settings.....	24
2.3.3 Comparisons of Model Performance.....	26
2.4 Simulation Examples.....	30
2.4.1 Simulation Data and Setting.....	30
2.4.2 Effect of Similarity Functions.....	31
2.4.3 Effects of Tuning Parameters.....	32
2.4.4 Effect of Training Size.....	35
2.4.5 Effect of Outcome Functions.....	37
2.4.6 Effect of Noise Variables.....	38
2.5 Summary.....	40
3. SIMILARITY-PRINCIPLE-BASED MACHINE LEARNING WITH BINARY	
OUTCOMES.....	42

3.1 Introduction.....	42
3.2 SBML Methods and Algorithms.....	44
3.3 Simulations – General Settings.....	50
3.4 Effect of Learning.....	53
3.5 Effect of Penalty Coefficient	55
3.6 Effect of Initial Similarity Scores	56
3.7 Effect of Unmeasured Predictors	57
3.8 Effect of Noise Variables.....	58
3.9 Effect of Correlation Among Attributes	62
3.10 Comparisons of Different Machine Learning Methods	65
3.11 Summary	70
4. RECURSIVE SIMILARITY-PRINCIPLE-BASED MACHINE LEARNING	73
4.1 Introduction.....	73
4.2 Methodology.....	74
4.3 Algorithms	75
4.4 Simulations	77
4.4.1 Simulation Data and Setting	78
4.4.2 Effect of Learning.....	81
4.4.3 Effect of Nonlinearity	83
4.4.4 Effect of Noise Variables.....	84
4.4.5 Effect of Unmeasured Predictors	85
4.4.6 Comparison of RSBML vs. SBML.....	86

4.5 Summary	89
5. CONCLUSION AND DISCUSSION	91
APPENDIX.....	93
<i>Application and Demo using Pima Indians Diabetes Dataset</i>	93
<i>R Code for Similarity-Principle-Based Machine Learning</i>	101
<i>R Code for Recursive Similarity-Principle-Based Machine Learning</i>	104
BIBLIOGRAPHY	105
CURRICULUM VITAE.....	109

LIST OF TABLES

Table 2.1 Training Errors vs. Proportion of Training Set - CF Data	27
Table 2.2 Testing Errors vs. Proportion of Training Set - CF Data.....	28
Table 2.3 Effect of Similarity Functions on MSEs for SBML	32
Table 2.4 Effect of Tuning Parameters on MSEs for SBML with P-values as Initial Similarity Scores.....	32
Table 2.5 Effect of Tuning Parameters on MSEs for SBML with Constant Initial Similarity Scores.....	34
Table 2.6 Effect of Training Size on MSEs.....	36
Table 2.7 Effect of Outcome Functions on MSEs	37
Table 2.8 Effect of Noise Variables on MSEs	39
Table 3.1 Effect of Tuning Parameters on MSEs for SBML with P-values as Initial Similarity Scores.....	53
Table 3.2 Effect of Learning on MSEs for SBML with Constant Initial Similarity Scores	54
Table 3.3 Effect of Penalty Coefficient on MSEs for SBML.....	55
Table 3.4 Effect of Initial Similarity Scores on MSEs for SBML.....	56
Table 3.5 Effect of Unmeasured Predictors on MSEs	57
Table 3.6 Effect of Noise Variables on MSEs	58
Table 3.7 Effect of Noise Variables on MSEs with Unmeasured Predictors	59
Table 3.8 Effect of Noise Variables on MSEs given Correlation.....	60
Table 3.9 Effect of Correlations on MSEs.....	63

Table 3.10 Effect of Correlations on MSEs with Unmeasured Predictors	64
Table 3.11 Effect of Training Size on Testing MSEs - Setting #2	67
Table 3.12 Effect of Training Size on Testing MSEs - Setting #3	67
Table 3.13 Effect of Training Size on Testing MSEs - Setting #4	68
Table 3.14 Effect of Training Size on Testing MCEs - Setting #2.....	69
Table 3.15 Effect of Training Size on Testing MCEs - Setting #3.....	69
Table 3.16 Effect of Training Size on Testing MCEs - Setting #4.....	70
Table 4.1 List of True Outcome Functions.....	78
Table 4.2 Effect of Learning on Testing MSEs for RSBML.....	82
Table 4.3 Effect of True Outcome Functions on MSEs.....	83
Table 4.4 Effect of Number of Noise Variables on MSEs with RSBML.....	85
Table 4.5 Effect of Unmeasured Predictors on MSEs with RSBML	86
Table 4.6 Comparison of RSBML vs. SBML.....	87

LIST OF FIGURES

Figure 2.1 Similarity Network (<i>similarix</i>) of 5 Nodes.....	14
Figure 2.2 Flowchart (algorithms) of Similarity-Principle-Based Machine-Learning	19
Figure 2.3 Predictions - Example of Similarity Principle in Action.....	21
Figure 2.4 Scatterplots, Histograms and Correlation Matrix - CF Data	26
Figure 2.5 Training Errors vs. Proportion of Training Set - CF Data.....	27
Figure 2.6 Testing Errors vs. Proportion of Training Set - CF Data	28
Figure 2.7 Effect of Tuning Parameters on MSEs for SBML with P-values as Initial Similarity Scores.....	33
Figure 2.8 Effect of Tuning Parameters on MSEs for SBML with Constant Initial Similarity Scores.....	34
Figure 2.9 Effect of Training Size on MSEs.....	36
Figure 3.1 Predictions - Example of Similarity Principle in Action.....	47
Figure 3.2 Effect of Learning on MSEs for SBML	54
Figure 3.3 Effect of Noise Variables on MSEs.....	61
Figure 3.4 Effect of Correlations and Unmeasured Predictors on MSEs	64
Figure 4.1 RSBML Illustration.....	74
Figure 4.2 Effect of Learning on MSEs for RSBML.....	82
Figure 4.3 Effect of Number of Noise Variables on MSEs for RSBML.....	85
Figure 4.4 Comparison of RSBML vs. SBML	87

LIST OF ABBREVIATIONS

BU	Boston University
DSF	Distance-Inverse Similarity Function
ESF	Exponential Similarity Function
FLM	Full Linear (or Logistic) Model
GLM	Generalized Linear Model
LSF	Logistic-Alike Similarity Function
MCE	Misclassification Error
MSE	Mean Squared Error
MVN	Multivariate Normal
OLM	Optimal Linear (or Logistic) Model
OLS	Ordinary Least Squares
RF	Random Forest
RR	Ridge Regression
RSBML	Recursive Similarity Based Machine Learning
SBML	Similarity Based Machine Learning
SF	Similarity Function

1. INTRODUCTION

1.1 Motivation

Classical hypothesis testing began with RA Fisher (1925) and progressed into its current mathematical formulation along a path marked by the work of Neyman and Pearson (Lehmann 1933), Wald (1950) and Lehmann (2005). Now, advances in computation, featuring AI methods, have reshaped the practice of statistics (Efron 2020), but most articles in medical journals still use a p-value to designate an important result. Before high-speed computing, a typical medical study might only have one hypothesis and a sample size small enough for adding machine calculation. P-values were not intended for large samples or for multiple comparisons, painfully obvious limitations when analyzing genetic data (Efron 2020). New criteria are needed.

The use of the type-I error rate in clinical trials and related settings have made it a widely used evidential measure of medicine effectiveness, but: 1) a p-value does not tell the effect of the treatment and 2) with small sample sizes such as in rare disease clinical trials, a large p-value can discredit a potentially effective treatment. This work addresses another weakness in the p-value – namely a p-value at most indicates the average effect of an intervention but can mislead us in a precision medicine study when the target population is heterogeneous. Conventional statistical methods evaluate treatment effects on the aggregate population. A treatment with no overall effect may well significantly benefit some and significantly disbenefit all the others. AI methods can predict with reasonable probability who will benefit from the treatment and who will not; therefore, the treatment can be made available to the appropriate subgroups of subjects. In many

cases, conservative multiple hypothesis testing procedures such as the Bonferroni correction may hide significant benefits to subgroups. This problem led to the development of false discovery rates (Benjamini 1995).

From this perspective the crucial task becomes finding those subgroups likely to benefit from the investigational treatment. Ultimately, achieving this task may identify the subjects who benefit and thereby make precision medicine more feasible. Artificial intelligence (AI) and machine learning (ML) algorithms focus on learning and prediction, making them important tools we should investigate.

Current AI deep learning methods include Convolution Neural Networks (CNNs) for Image Recognition (Lu et al. 2017; Le Cun and Bengio 1995), Recurrent Neural Networks (RNNs) for Speech Recognition, Natural Language Processing (Williams et al. 1986; Li and Wu 2014) and Deep Belief Networks (DBNs) for disease or cancer diagnosis / prognosis (Carneiro et al. 2012; Ngo et al. 2016; Kim et al. 2017). These methods require big data and cannot be directly applied to smaller samples. Furthermore, traditional modelling techniques such as generalized linear models do not perform well for heterogeneous data as we see in rare disease trials.

How best to combine analytic methods raises difficult issues. Several AI or ML predictive methods applied to product development use statistical methods such as ridge regression (Hoerl and Kennard 1970), optimal regression via stepwise variable selection (Efroymson 1960; Beale 1970) and kernel methods (Hofmann et al. 2008). Other methods such as deep learning neural networks, reinforcement learning and evolutionary intelligence have been used in drug discovery when big data is available. However, their

performance is not always satisfactory because the predictions might be unreliable, especially when the sample size is small.

The kernel method (Scholkopf, 2004) fails to objectively consider the relative importance of different predictors and often does not perform well in the case of small sample sizes. On the other hand, structured local regression uses structured kernels with consideration of weights for different attributes (Hastie et al, 2001). The kernels are used for local error weighting and minimization. In this dissertation, we propose a Similarity-Principle-Based Machine Learning (SBML) method, which is based on the general *similarity principle* and objectively determines the relative importance of different predictors in the similarity measures. Mathematically, it can be viewed as a combination or generalization of the structured kernel and Nadaraya-Watson kernel-weighting method (Hastie et al, 2001). Simulations and a case study show that SBML performs better than traditional methods under most settings in our study. We hope this work will bring more attention to the AI or ML methods in clinical trials and beyond.

The rest of this chapter is organized as follows: Section 1.2 describes a brief overview of three common types of ML algorithms. Section 1.3 briefly describes a few popular supervised ML methods (including kernel methods, support vector machine, decision tree methods, logistic regression). Section 1.4 briefly introduces the notion of the similarity principle.

1.2 Machine Learning

Artificial intelligence (AI) or machine learning (ML) algorithms have garnered significant attention as of late for their potential applications in the field of drug

development (Chang 2010; Hughes et al. 2015; Lu and Zhang et al. 2017). However, methodology developments and applications in clinical trials where the sample size is limited are lacking.

In addition, the majority of AI or ML methods require big data to be effective. Unfortunately, in clinical trials, especially in oncology or rare disease trials, patients' characteristics are heterogeneous, and the number of subjects is limited. Also, in an emergency situation like the COVID-19 pandemic, where the novel virus spreads rapidly, and the number of subjects who tested positive grows exponentially throughout different countries, decisions (e.g., policies, treatment procedures, etc.) have to be made early and quickly based on limited available patient data (which may not have been seen before). Indeed, AI or ML methods with high predictive accuracy for both big data and small sample sizes are highly desirable.

Experts broadly distinguish three common types of ML algorithms: supervised, unsupervised and reinforcement learning (Ramasubramanian and Singh 2017; François-Lavet et al. 2018).

Supervised learning algorithms require that input data have labeled output data. The algorithm learns from known features of that particular data to generate an output model that successfully predicts labels for new (incoming, unlabeled) data points. Supervised learning consists of two parts: classification (e.g., medical diagnosis, fraud detection, spam email detection, image classification, etc.) and regression (e.g., score prediction, risk assessment, weather forecasting, etc.). Examples of supervised learning include linear or logistic regressions, support vector machines (SVM), kernel methods

(Vapnik 1995, 1998), Naïve Bayes (for classification problems), K-Nearest Neighbors (Altman 1992), tree methods such as CART (Classification And Regression Tree) and random forests (Quinlan 1986; Rokach 2008).

Unsupervised learning algorithms accept unlabeled data and attempt to group observations into clusters based on underlying similarities in input features. Examples of unsupervised learning include Principal Component Analysis (PCA) (Pearson 1901), and clustering algorithms such as K-Means Clustering (MacQueen 1967) and Hierarchical Clustering. Unsupervised learning methods are mainly used for: Clustering and Association (e.g., face recognition, text mining, market basket analysis, etc.).

Reinforcement learning algorithms are analogous to human and animal cognition and mainly used for robot navigation and gaming (e.g., Google Deep Mind's AlphaGo defeats Go champion Lee Sedol, etc.). Briefly, the model is “rewarded” based on its behavior, through which it learns to maximize the sum of its rewards by adapting the decisions it makes to earn as many rewards as possible.

In drug discovery, supervised learning methods have been widely used since the late 1990s, particularly in quantitative structure-activity relationships (QSAR), protein folding studies, and molecular designs, and more recently used for disease diagnosis and prognosis. However, few have applied ML in clinical trials largely due to limited sample size. In this work, we propose a supervised ML method: Similarity-Principle-Based Machine Learning (SBML) for clinical trials and beyond.

1.3 Types of Supervised Machine Learning Methods

Supervised ML algorithms such as SVM, kernel methods, KNN, decision trees and random forest methods can be applied to both regression and classification problems. Logistic regression is often used for classification problems, while linear regression is often applied for regression problems with continuous normally distributed outcomes.

1.3.1 Kernel Methods

In classical statistical models for regression and classification, the form of the mapping $y(x, w)$ from input x to output y is governed by a set of adaptive parameters w . During the learning phase, a set of training data is used to obtain estimates of these parameters. Then in the testing phase, the model is used to predict the outputs of new unseen inputs based solely on the learned parameters w . This approach is also used in nonlinear parametric models such as neural networks, but SBML and kernel methods (KM) are memory-based approaches that involve storing the entire training set (dimension reduction is possible with modifications) in order to make future predictions. These methods are generally fast to train but slow at making predictions for test data points.

A typical kernel $k(x, x_j)$, defined as a *dot product*, can be viewed as the similarity between objects that are characterized by attributes x and x_j . Once the kernel is selected and weights w_j ($j = 1, \dots, N$) for the N training subjects are determined, the predicted outcome for the new subject can be expressed as a weighted sum (linear combination) of the kernels (similarities). Through *statistical learning*, the weights can be updated based on the loss or error minimization.

$$Y = \sum_j w_j k(x, x_j) .$$

KM appears to be similar to SBML, but they actually differ at least in two ways: (1) *KM* is an over-parameterized model with N parameters, while SBML has only K attributes-scaling factors. (2) Similarities (kernels) in KM are pre-determined based on field-experts' judgments, while similarities in SBML are objectively determined through training the attribute-scaling factors.

1.3.2 Support Vector Machine

Linear discriminant analysis (LDA) is based on the construction of the hyperplane that minimizes the misclassification error. Similarly, a *support vector machine* (SVM), developed in the mid-1960s, is a generalization of LDA for constructing hyperplanes that minimize the misclassification or regression error. SVM searches for a linear decision boundary that separates members of one class from the other. In the case that such a hyperplane does not exist, SVM uses a nonlinear mapping to transform the training data into a higher dimension before seeking the linear optimal separating hyperplane. With appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. SVM has successfully been applied to handwritten digit recognition, text classification, speaker identification, etc., and it is less prone to overfitting. KM and SVM have been broadly used in bioinformatics (Schölkopf, et al 2004), and SVM has also been used in breast cancer diagnosis (Akay, 2009).

1.3.3 Decision Tree Methods

Decision tree (DT) *methods* or simply *tree methods* are among the most popular methods in statistical machine learning. They are intuitive, as well as easy to use and interpret.

There are two types of trees based on the outcome: *classification* and *regression* trees (CART). In a regression tree, the outcome is a continuous variable, while in a classification tree the outcome is a discrete variable.

When we grow the tree, we use a recursive binary split. In the regression setting, the residual sum of squares (RSS) is used as a criterion for making the splits, whereas the misclassification error (MCE) rate is the natural alternative to the RSS in the classification setting. The common impurity measures for binary classifications are: misclassification rate, p , defined as proportion of misclassification of subjects, Gini index defined as $GI = p \cdot (1-p)$, and cross-entropy defined as $D = p \cdot \ln(p)$.

For a given tree depth, a commonly used approach to obtaining an optimal tree by minimizing MCE, GI , or D is the greedy algorithm: for each parameter, try different thresholds. We can let the tree grow larger than what we need at the final state, then prune it.

A single big tree is not stable because a single error in classification can propagate to the leaves. Frequently used remedies are ensemble learning methods such as *bagging*, *boosting*, and *random forests*. Bootstrap aggregating (*bagging*) is one of the techniques for model averaging and improvement, simply forming an average of many different trees that are generated from multiple training sets with replacements. Applying this idea to the decision tree method leads us to the tree-averaging method. Similar to *bagging* is *boosting*. Weak classifiers $G_i(x)$ with a value of either 1 and -1 from n samples are those whose misclassification error rates are only slightly better than random guessing. The

predictions from all of them are then combined through a weighted majority vote to produce the final prediction.

Random forests combine the simplicity of decision tree methods with flexibility resulting in a vast improvement in accuracy. For classification problems, a random forest is an ensemble classifier that consists of many decision trees and outputs the class that has a majority vote by the individual trees. Advantages of random forests are as follows: 1) it reduces the risk of overfitting; 2) hence, for large data, it produces highly accurate predictions; 3) it can estimate missing data and maintain accuracy when a large proportion of data is missing; 4) training time is less, and it runs efficiently on large data.

1.3.4 Logistic Regression

Logistic regression is probably one of the most commonly used methods for the analysis and predictions of binary outcomes. We model a transformation of the data, ‘logit function’ which takes the following form:

$$\ln(\text{odds}) = \ln \frac{p}{1-p}, \text{ for } 0 < p < 1,$$

where ‘ln’ represents the natural-log, p represents the probability of the event ($Y = 1$).

Y_i only takes 0 or 1 for given values of k predictors, and the expected value or mean of Y equals p_i . Logistic regression assumes linear relationship between the logit function and the predictors:

$$\ln \frac{p_i}{1-p_i} = \text{Logit } E[Y_i] = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik},$$

where p_i represents the probability of the event ($Y_i = 1$), expressed as

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik})}{1 + \exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik})} = \frac{1}{1 + \exp(-\beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \dots - \beta_k x_{ik})}$$

1.4 Role of Similarity Principle in Scientific Discovery

The *similarity principle* (Chang 2012) posits that identical or similar things should behave in a similar manner. Grouping similar objects together arises from subjective prior knowledge. For instance, we believe that groups of subjects with same or similar diseases, races, genders, ages, and genotypes will likely have similar responses to a treatment or medical intervention, because such associations have occurred with related interventions. If the subjects are similar in more aspects, they will respond to a drug more similarly (Chang 2012, 2014, 2016).

The key is how to define the similarity of groups of objects or subjects in terms of pre-selected sets of attributes. The selection of attributes to define similar groups should also depend on the outcomes of interest. For example, the attributes (inclusion and exclusion criteria) selected for oncology and cardiovascular trials may be different.

The *similarity principle* is implicitly applied in drug discovery. For example, SAR (structure activity relationship), which means compounds with similar substructures will behave similarly, is the key foundation to many aspects of drug discovery. Additionally, we believe if a drug is shown to be toxic to animals, the toxicity will also likely manifest in humans due to the similarity between humans and the animals. For instance, a new chemical compound is tested on animals first, and after it has shown to be effective or efficacious and well tolerated, it will be tested on humans in clinical trials. Similarly, if a drug is shown to decrease cancer cell activities in animals, we believe it will likely decrease cancer cell activities in humans as well to a certain degree. Indeed, the

utilization of allometric scaling is common to extrapolate animal data to determine pharmacokinetic parameters in humans.

An implicit but reasonable underlying assumption these examples illustrate is access to a large detailed scientific knowledge base about the diseases of interest that suggest what patient characteristics matter most and how they might cluster. Heart disease research has developed many well-defined etiological biochemical mechanisms but no precise model describing how much weight to assign to various clinical, demographic, environmental and genetic factors. Implicitly guided by science, similarity methods organize a myriad of characteristics into a large yet manageable number of clusters rather than invoking a mindless clustering algorithm.

A key challenge is how to determine the weights of these attributes that define similarities. Subjects differ in age, gender, race, height, weight, etc. If properly determined, we can use the similarity weighting of the outcome of the subjects in the clinical trial to predict the outcome of future subject(s).

To make the *similarity principle* (Chang 2012) operational, we will discuss how to objectively determine the weights of each attribute in SBML and how to update the attribute-scaling factors by learning / training in the next section.

1.5 General Simulation Settings

For large sample size, one training dataset and one test dataset may be sufficient for evaluation of different methods. However, for small dataset as in most clinical trials, one training set and one test set may not give a stable evaluation. Therefore, we simulate 100 trials or samples for each scenario under consideration and average the errors. In this

work, the mean squared errors (MSEs) presented as errors refer to the normalized average MSEs based on 100 samples, which indicate that the average of 100 MSEs is divided by the average of 100 variances of Y . Calculating an average of 100 samples/simulations is to avoid variability caused by each sample, and 100 is good enough to reduce sample variability in comparison among different methods.

2. SIMILARITY-PRINCIPLE-BASED MACHINE LEARNING WITH CONTINUOUS OUTCOMES

2.1 Introduction

In this chapter, we are going to implement the *similarity principle* explained in Section 1.4 and propose a novel machine learning technique called Similarity-Principle-Based Machine Learning (SBML) for continuous outcomes. The chapter is organized as follows. Section 2.2 introduces the details of the proposed method (for continuous outcomes) including notations, methods and algorithms. In Section 2.3, we apply our method and compare the prediction accuracy with other traditional methods under rare disease clinical trials settings. In Section 2.4, we conduct extensive simulations under various settings and hyperparameters to compare the testing errors and to find optimal values for the tuning parameters. In Section 2.5, we discuss and summarize our findings, including recommendations of the values for the tuning parameters for SBML.

2.2 Methodology

2.2.1 Similarity Weighting

First, we note that a regression method directly models the relationship of the outcome of interest (i.e., dependent variable) with independent variables, while our SBML method indirectly models the relationship of the values of the outcome of interest (i.e., outcome values of 1 dependent variable) among different subjects. Their outcomes are weighted through dependent variables (e.g., baseline characteristics).

Based on the *similarity principle*, we can predict the outcome \hat{Y}_i for a new i^{th} subject, in terms of the similarity-weights and observed outcomes (O_j) of all the other subjects:

$$\hat{Y}_i = \sum_{j=1}^N W_{ij} O_j, \quad i = 1, 2, \dots, N \quad (1)$$

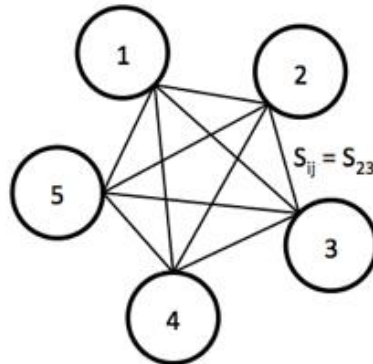
where the weights are the normalized similarity scores: $W_{ij} = \frac{S_{ij}}{\sum_{m=1}^N S_{im}}$.

Similarity between subjects or between events can be expressed in a matrix and visualized using a similarity network, called *similarix*. In a *similarix*, nodes (vertices) represent subjects (objects, or events) of interest. The weights associated with the links (edges, lines) in the network are the similarity scores between the paired subjects.

$$\mathbf{S} = \begin{bmatrix} S_{11} = 1 & S_{12} & S_{13} & \cdots & S_{1N} \\ S_{21} & S_{22} = 1 & S_{23} & \cdots & S_{2N} \\ S_{31} & S_{32} & S_{33} = 1 & \cdots & S_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ S_{N1} & S_{N2} & S_{N3} & \cdots & S_{NN} = 1 \end{bmatrix}_{N \times N},$$

where the similarity between the i^{th} and j^{th} subject, $S_{ij} = S_{ji}$, ranges from 0 to 1. An example of similarity network (*similarix*) of 5 nodes is illustrated in Figure 2.1.

Figure 2.1 Similarity Network (*similarix*) of 5 Nodes



The key is to determine appropriate similarity scores. There are several existing similarity measures available, such as a radial basis function (RBF) kernel on two samples \mathbf{x} and \mathbf{x}' , represented as feature vectors in some input space, which is defined as the following:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2\right) , \quad (2)$$

where $\|\mathbf{x} - \mathbf{x}'\|^2$ may be recognized as the squared Euclidean distance between the two feature vectors.

For instance, a kernel method (Schölkopf et al. 2004) in classification problems use a similarity function (kernel) between the unlabeled input and each of the training inputs. A kernelized binary classifier typically computes a weighted sum of similarities as

$$Y_j = \text{sign} \sum_{i=1}^N w_i O_i k(x_i, x_j) , \quad (3)$$

where N is the number of subjects or data points in the training set, x_i is from the training set, and x_j is from the test set. When the Kernel method is used for a continuous variable, it is called a Kernel smoother.

In this work, we focus on the elliptical basis function (EBF) as our similarity function:

$$S(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{R}(\mathbf{x} - \mathbf{x}')\|^2\right) , \quad (4)$$

where the attribute-scaling factors \mathbf{R} can be learned and updated through training.

It is important to know that 1) the existing kernel methods fail to recognize the relative importance of different attributes. SBML uses the attribute-scaling factors to determine the relative importance among different attributes. 2) SBML only needs to determine a small set of attribute-scaling factors \mathbf{R} ; in contrast, the kernel methods need to determine

a larger number of weights, which is approximately the squared to the number of subjects in the training set. This is because weights in kernel methods are independent to each other, and the weights in SBML are not independent to each other, but determined by the attribute-scaling factors \mathbf{R} . 3) Regression methods are parametric approaches, and kernel methods are instance approaches, while SBML is a combination of the two.

In SBML, there are the same number of weights, but they are not independent; rather, they are related by the scaling factors for the attributes. In other words, for SBML, only a limited number of attribute-scaling factors (i.e., features or parameters) need to be determined. Kernel methods and SBML may look similar, but the major difference is that our similarity scores (S_{ij}) are determined by data, and we only need to estimate K scaling parameters. Kernel methods use similarity as attributes to define subjects, while SBML uses the *similarity principle* and similarity scores as weights in prediction.

2.2.2 Data Normalization

Data normalization is a method to standardize all variables in the dataset so that they all have a mean of 0 and a standard deviation of 1. The standardization is achieved for each variable by subtracting its mean and then dividing by its standard deviation. The standardization makes the invented methodology robust and easy to apply the SBML. For example, if the range of age (in years) is from 18 to 70, standardized age will range from a negative value to a positive value with a mean of 0. Also, once we estimate the attribute-scaling factors or values of the tuning parameters using the standardized data, we can apply them to different datasets, which may have variables in different units.

2.2.3 Notations

Suppose N is the training size and M is the test size, such that $i = 1, \dots, N$ (if training set) or M (if test set); $j = 1, \dots, N$ (i.e., j^{th} person is only in the training set); $k = 1, 2, \dots, K$, where K denotes the number of features in the model. In the training stage, both i^{th} and j^{th} persons belong to the training set, and X_{ik} denotes an element in a $N \times K$ matrix in the training set, similar to a design matrix in simple linear regressions, where 1st column X_{i1} represents main effect (i.e., $k = 1$ for treatment, coded as 1 or 0), and 2nd through K^{th} column (X_{i2}, \dots, X_{iK}) represents other baseline attributes (e.g., age, gender, race, etc.). O_i is the observed outcome for i^{th} subject, where $i = 1, 2, \dots, N$ (if training set) or M (if test set). \hat{Y}_i is the predicted outcome for i^{th} subject.

2.2.4 Algorithms

The SBML algorithm is outlined in Figure 2.2 and elaborated as follows.

Step 1) Normalize the Training dataset as described in Section 2.2.2.

Step 2) Assign Initial Similarity Scores (S_k^0):

e.g., $S_k^0 =$ p-values from regression model or arbitrary number between 0 and 1, say 0.5.

That is, if we have 5 attributes (X_1, X_2, X_3, X_4, X_5) in the model, then the vector for the initial similarity scores can be $(p_1, p_2, p_3, p_4, p_5)$, where p_k is a p-value for the k^{th} variable from the regression model, or it can be $(0.5, 0.5, 0.5, 0.5, 0.5)$.

Step 3) Determination / Estimation of Initial Scaling Factors (R_k^0) – Solve for R_k^0 based on initial similarity scores, e.g., $S_k^0 =$ p-values:

It is difficult to determine the attribute-scaling factors \mathbf{R} directly from prior knowledge or current data. It is much easier to calculate \mathbf{R} through the similarity scores S_{ij} as follows.

When a pair of subjects (the k^{th} pair) is the same in terms of all the attributes under consideration, except one (the k^{th}) attribute, then the summation will disappear on the right-hand side of Eq. (6). Then the attribute-scaling factor R_k can be solved explicitly; for instance, if we use Exponential Similarity Function (ESF) as in Eq. (9), R_k can be solved using Eq. (5) as follows

$$R_k^0 = \frac{-\ln(S_k^0)}{|X_{ik} - X_{jk}|} = \frac{-\ln(S_k^0)}{IQR(X_k)}, \quad k = 1, 2, \dots, K \quad (5)$$

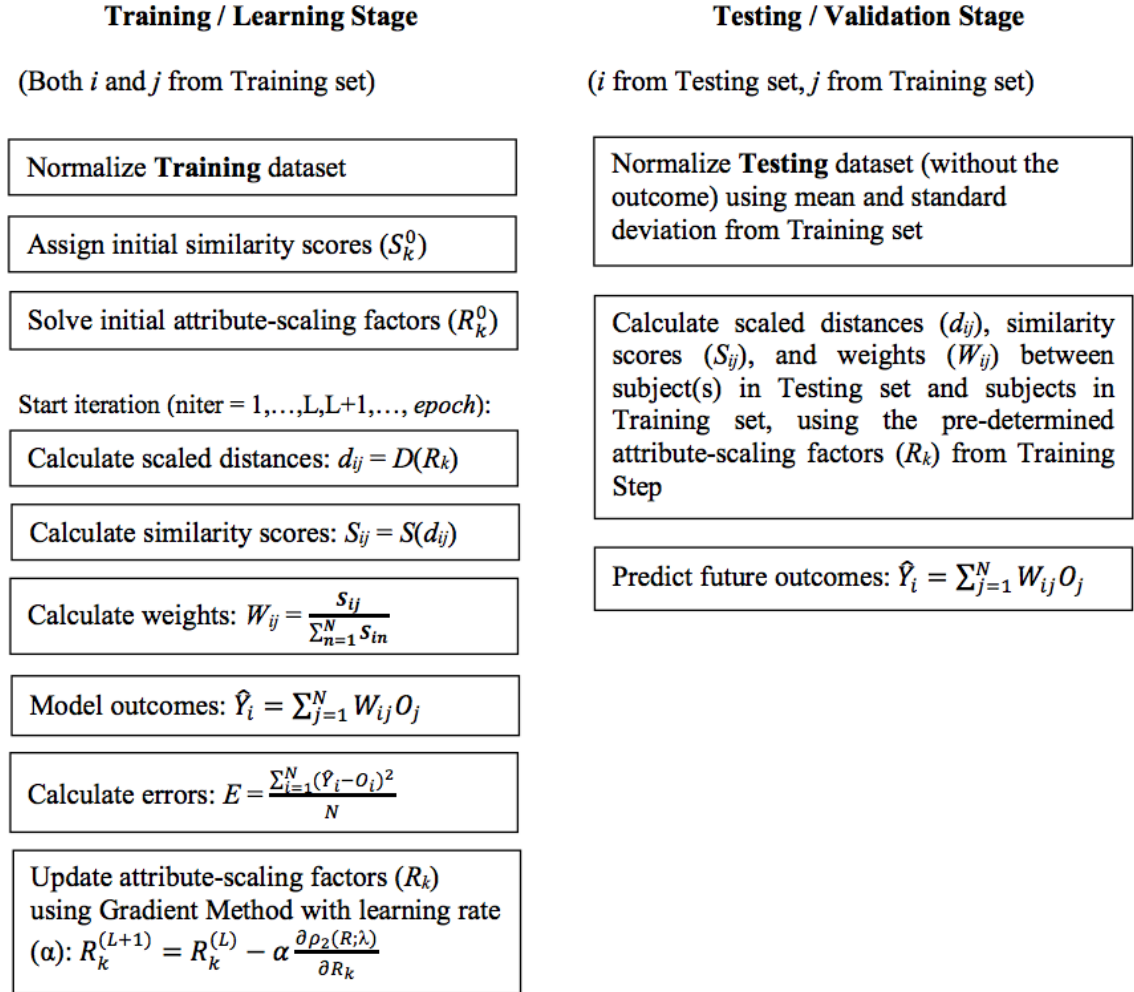
where the superscript, “0”, indicates the initial values, $|X_{ik} - X_{jk}|$ can be substituted with the inter-quartile range (IQR) between 1st and 3rd quartiles of the k^{th} attribute. The IQR can be considered as the difference between two typical subjects in the data regarding the corresponding k^{th} attribute. For this reason, we choose the initial K (real or virtual) pairs of subjects this way and determine their K initial similarity scores.

Step 3.1) When limited training data are available, we need to use our prior knowledge, which will be vague and difficult to specify. In other words, based on prior knowledge of similarity scores between some selected subjects, we solve for $\mathbf{R} = (R_1, R_2, \dots, R_K)$ using a defined similarity function: $\mathbf{R} = \text{function}(\mathbf{S})$, where $\mathbf{S} = (S_1, S_2, \dots, S_K)$.

Step 3.2) P-value-based similarity scores:

When some training data are available, we obtain p-values from a statistical model (a linear model for a continuous outcome, a logistic model for a binary or ordinal outcome, and a cox model for a time-to-event outcome) and assign these p-values to the initial similarity scores between the K selected subjects and then determine \mathbf{R} using the defined similarity function. Here we see that the p-values can be interpreted as similarity scores.

Figure 2.2 Flowchart (algorithms) of Similarity-Principle-Based Machine-Learning



Note: the iteration of learning stops when the loss function is minimized or when the pre-determined maximum number of iterations (*epoch*) is reached.

Step 4) Determination of Scaled Distance: Let d_{ij} be the scaled distance between 2

subjects (say i and j) as a function of the attribute-scaling factor R_k ($k=1, 2, \dots, K$):

$$d_{ij} = \left[\sum_{k=1}^K (v_{ijk})^\rho \right]^{\frac{1}{\rho}} = \left[\sum_{k=1}^K (R_k |X_{ik} - X_{jk}|)^\rho \right]^{\frac{1}{\rho}}, k = 1, 2, \dots, K; \rho = 1 \text{ or } 2 \quad (6)$$

where $v_{ijk} = R_k |X_{ik} - X_{jk}|$ is the scaled absolute difference of the k^{th} attribute between 2 subjects (i and j).

Step 5) Similarity Functions / Measures: Define the similarity score S_{ij} between 2 subjects (i and j) as a function of the scaled distance d_{ij} :

$$S_{ij} = S(d_{ij}), \quad \text{where } S_{ij} \in [0,1] \quad (7)$$

The similarity score ranges from 0 (completely different) to 1 (identical) inclusively. The common requirements for a similarity function $S_{ij}(d_{ij})$ are $S_{ij}(0) = 1$ and $S_{ij}(\infty) = 0$.

Example 1) Exponential Similarity Function / Score (ESF):

$$S_{ij} = \exp(-d_{ij}^\eta) \quad (8)$$

For ESF with $\eta = 1$ and $\rho = 2$, the similarity function in Eq. (8) becomes

$$S_{ij} = \exp\left(-\sqrt{\sum_{k=1}^K (R_k |X_{ik} - X_{jk}|)^2}\right), \quad \eta = 1 \text{ and } \rho = 2 \quad (9)$$

Example 2) Logistic-Alike Similarity Function (LSF):

$$S_{ij} = \frac{2}{1 + \exp(d_{ij}^\eta)}, \quad \eta > 0 \quad (10)$$

Example 3) Distance-Inverse Similarity Function (DSF):

$$S_{ij} = \frac{1}{1 + d_{ij}^\eta}, \quad \eta > 0 \quad (11)$$

Step 6) Determination of Weight: Define the weight W_{ij} between 2 subjects (i and j) as a function of the similarity score S_{ij} such that it is the normalized similarity score:

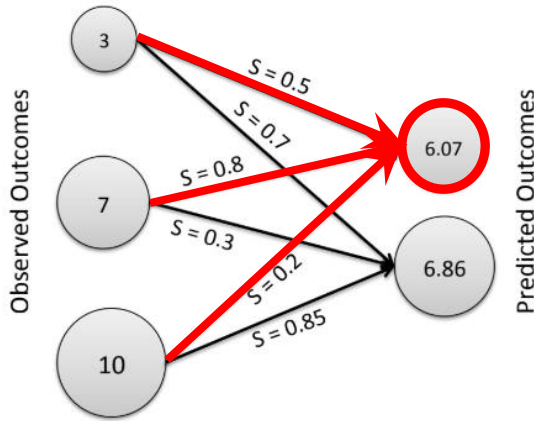
$$W_{ij} = \frac{S_{ij}}{\sum_{n=1}^N S_{in}} \quad (12)$$

Step 7) Prediction of Outcome: We can predict the outcome \hat{Y}_i for the i^{th} subject (based on similarity-weighted mean, i.e., based on the similarity-weights and observed outcomes of its direct neighbors or all the other subjects):

$$\hat{Y}_i = \sum_{j=1}^N W_{ij} O_j, \quad i = 1, 2, \dots, N \quad (13)$$

where O_j is the observed outcome for the j^{th} subject. Figure 2.3 illustrates how the predicted outcomes of 2 nodes are calculated from the 3 observed outcomes using the similarity scores. Also, from Eq. (13), as the number of subjects of type i increases, the estimation of \hat{Y}_i approaches to O_i . This is an important feature of learning.

Figure 2.3 Predictions - Example of Similarity Principle in Action



$$\hat{Y}_1 = \frac{(3 * 0.5 + 7 * 0.8 + 10 * 0.2)}{(0.5 + 0.8 + 0.2)} = 6.07$$

Step 8) Calculation of Error:

The mean squared error (MSE) can be calculated using the following formulation:

$$E = \frac{\sum_{i=1}^N (\hat{Y}_i - O_i)^2}{N} = \frac{1}{N} \sum_{i=1}^N \left(\sum_{j=1}^N W_{ij} O_j - O_i \right)^2 \quad (14)$$

Step 9) Iteration of Learning: Learning is basically updating the attribute-scaling factors

R. We can use the gradient method. To reduce the overfitting, we can minimize the *loss function* instead of MSE. There are several commonly used loss functions available.

$$\text{Ridge Loss Function} = \rho_2(\mathbf{R}; \lambda) = E + \lambda \|\mathbf{R}\|_2^2 \quad (15)$$

where λ is a tuning parameter for the penalty term and $\|\mathbf{R}\|_2^2 = \sqrt{\sum_{k=1}^K R_k^2}$.

When $\lambda = 0$, the loss function reduces to MSE without the penalty term. The large value of λ puts a heavy penalty on large values of R_k .

Since $\frac{\partial \|\mathbf{R}\|_2^2}{\partial R_k} = 2R_k$, the derivative of the ridge loss function is as follows:

$$\frac{\partial \rho_2(\mathbf{R}; \lambda)}{\partial R_k} = \frac{\partial E}{\partial R_k} + \frac{\lambda \partial \|\mathbf{R}\|_2^2}{\partial R_k} = \frac{\partial E}{\partial R_k} + 2\lambda R_k \quad (16)$$

The gradient method to update the attribute-scaling factor R_k from iteration L to iteration L+1 can now be specified as follows:

$$R_k^{(L+1)} = R_k^{(L)} - \alpha \frac{\partial \rho_2(\mathbf{R}; \lambda)}{\partial R_k} \quad (17)$$

The derivative of error, $\frac{\partial E}{\partial R_k}$, in Eq. (16) can be derived for the ESF:

$$\left\{ \begin{array}{l} \frac{\partial E}{\partial R_k} = \frac{2}{N} \sum_{i=1}^N \left((\hat{Y}_i - O_i) \sum_{j=1}^N O_j \frac{\partial W_{ij}}{\partial R_k} \right), \text{ where} \\ \frac{\partial W_{ij}}{\partial R_k} = \frac{1}{\sum_{n=1}^N S_{in}} \frac{\partial S_{ij}}{\partial R_k} - \frac{S_{ij} \sum_{n=1}^N \frac{\partial S_{in}}{\partial R_k}}{(\sum_{n=1}^N S_{in})^2} \\ \frac{\partial S_{in}}{\partial R_k} = -\eta S_{in} d_{in}^{\eta-1} \frac{\partial d_{in}}{\partial R_k} \\ \frac{\partial d_{in}}{\partial R_k} = \left(\frac{R_k}{d_{in}} \right)^{\rho-1} |X_{ik} - X_{jk}|^\rho, d_{in} \neq 0; \text{ otherwise, } \frac{\partial d_{in}}{\partial R_k} = 0 \end{array} \right. \quad (18)$$

The iteration of learning stops when the loss function is minimized or when the pre-determined maximum number of iterations (*epoch*) is reached.

Step 10) Prediction of Future Outcome: Once the attribute-scaling factors \mathbf{R} is determined using the training set, we can predict the outcome for the future subjects using Eq. (13).

To evaluate the prediction / performance of the method, we define the error for the i^{th} subject in the test set as

$$E_i = (\hat{Y}_i - O_i)^2 \quad i = 1, 2, \dots, M \text{ in test set} \quad (19)$$

If there are M future subjects to predict (i.e., $i = 1, 2, \dots, M$ in the future or test set, while $j = 1, 2, \dots, N$ in the training set), the mean square error (MSE) is

$$E = \frac{\sum_{i=1}^M E_i}{M} = \frac{\sum_{i=1}^M (\hat{Y}_i - O_i)^2}{M} = \frac{1}{M} \sum_{i=1}^M \left(\sum_{j=1}^N W_{ij} O_j - O_i \right)^2 \quad (20)$$

where the weights W_{ij} are determined based on the similarity scores S_{ij} between the i^{th} subject in the test set and the j^{th} subject in the training set.

Further details of the algorithm proposed in this section are illustrated step-by-step in the Appendix using a partial data of the *PimaIndiansDiabetes* dataset, which is originally taken from the National Institute of Diabetes and Digestive and Kidney Diseases and is available in *mlbench* package in R.

2.3 Application to Rare Disease Clinical Trials

2.3.1 Cystic Fibrosis – Rare Disease

Cystic Fibrosis (CF) is a rare, inherited, autosomal recessive genetic, life-threatening disorder that causes lung, pancreatic and digestive problems. The body produces thick and sticky mucus that can clog the lungs and obstruct the pancreas. CF damages multiple organs and systems in the body including respiratory, pancreatic, gastrointestinal,

reproductive and integumentary. CF is considered a rare disease affecting less than 200,000 people annually in the US and most commonly affects Caucasians.

2.3.2 General Settings

In order to evaluate the performance of different methods without bias and in a robust and fair manner, the target population should be well-defined in a statistical sense. This can be done using simulations to generate well-defined populations, although it may not reflect the population in the real world. In this section, we use published population data from real clinical trials and generate the target population. Due to confidentiality concerns, no actual trial data could be used. To evaluate the performance of different methods, published populations data had been used, and simulations were conducted to generate individual level data to meet the population level data. The information about the rare disease trials is mainly from the five publications (Ramsey et al. 2011; Flume et al. 2012; Davies et al. 2013; Moss et al. 2015; Ratjen et al. 2017) in different clinical trials with repeated measures from baseline and post-treatment. The sampling pool data with a total number of 260 subjects are generated as individual subject data.

For CF studies, the primary efficacy endpoint of percent predicted forced expiratory volume in the first second ($ppFEV_1$) has been widely used to evaluate lung function changes, and the following four predictors are considered for the analysis: treatment, age, sex and baseline $ppFEV_1$.

The prediction accuracy of SBML will be compared against traditional methods such as full linear model (FLM), optimal linear model (OLM) and the ridge regression (RR), mixed effect model for repeated measures (MMRM) with unstructured variance-

covariance, as well as a couple of popular machine learning methods, namely support vector machine (SVM) with radial basis function (RBF) kernel and random forests (RF).

In the stepwise algorithm for OLM, Bayesian Information Criteria (BIC) was used. For RR, the tuning parameter λ is determined by cross-validation. The R packages ‘ridge’ and ‘glmnet’ are used in the simulations. For MMRM, ppFEV₁ measures from previous visits are used to predict the outcome at the final visit, while all other methods only use baseline ppFEV₁ as one of the predictors.

For SBML, we use the following tuning parameters with the exponential similarity function (ESF): a constant value of 0.5 as the initial similarity scores (S^0) for each of the predictors, i.e., (0.5, 0.5, 0.5, 0.5), a fixed learning rate (α) of 0.125, a penalty coefficients (λ) ranging from -0.5 to 0.5, and 5 as the maximum iterations of learning (i.e., $epoch = 5$), unless specified otherwise.

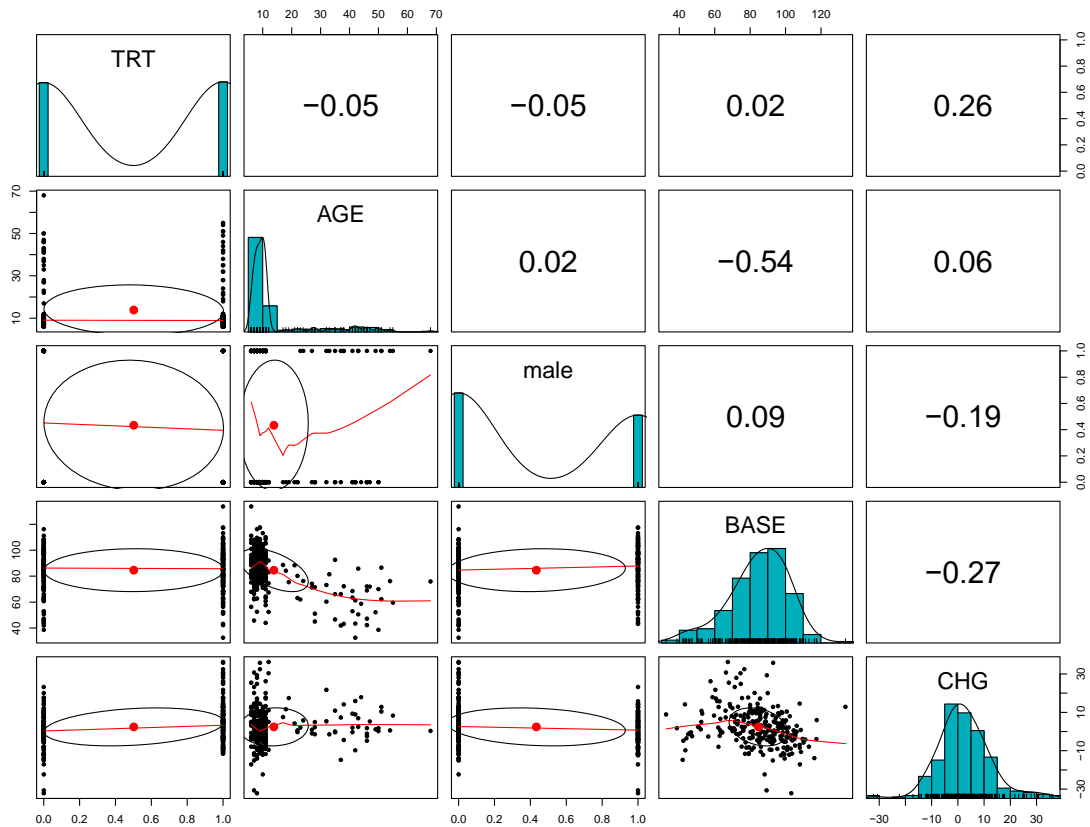
As we alluded earlier, the target population needs to be well defined in model evaluations. The reason we use bootstrapping with replacement from the empirical distribution of the small trial datasets are stated as follows: for big data, sampling with or without replacement has little impact on the comparison of different methods. However, for a small sample size such as data from rare disease trials, splitting sample may not be appropriate since the two split samples may have totally different distributions, and no method should be required to predict well on the outcomes for population, while trained on a population with totally different distribution. For example, as an extreme case, suppose there are 10 total subjects; among 10 total subjects, 5 are female and 5 are male. When we split them into a training set and a test set, all 5 females end up in the training

set, and all 5 males in the test set. To predict breast cancer for all 5 males based on the model trained based on 5 females will not be accurate.

2.3.3 Comparisons of Model Performance

Figure 2.4 contains a matrix with scatterplots of the data on the lower half of the matrix, histograms and variable names on the diagonals depicting the distributions of values for each variable of interest, as well as correlation coefficients (r) on the upper half.

Figure 2.4 Scatterplots, Histograms and Correlation Matrix - CF Data



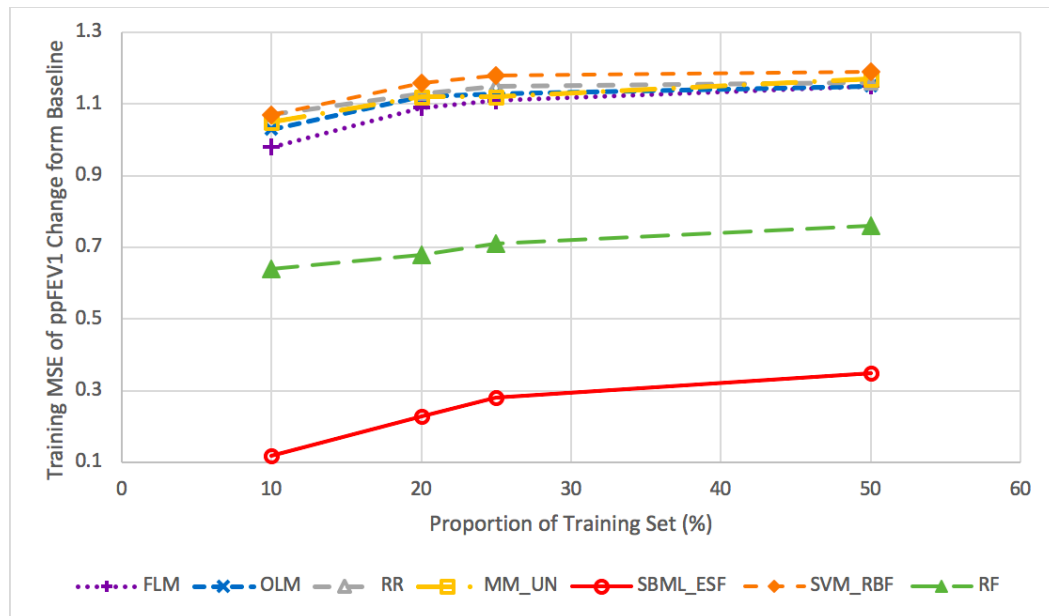
This scatterplot matrix shows mild correlations between TRT and CHG, BASE and CHG, and most importantly, between AGE and BASE, which may cause collinearity issues. But

the data structure is simple, and only a few attributes have effects on the outcome CHG – thus SBML may not show full strength.

Table 2.1 Training Errors vs. Proportion of Training Set - CF Data

Proportion of Training Set, %	FLM	OLM	RR	MMRM	SVM	RF	SBML
10 ($N_{train} = 26$)	0.979	1.034	1.066	1.05	1.074	0.639	0.121
20 ($N_{train} = 53$)	1.089	1.116	1.125	1.12	1.155	0.687	0.235
25 ($N_{train} = 66$)	1.115	1.135	1.146	1.12	1.176	0.708	0.285
50 ($N_{train} = 132$)	1.148	1.153	1.158	1.17	1.187	0.76	0.353

Figure 2.5 Training Errors vs. Proportion of Training Set - CF Data



The evaluations of different methods are summarized in Table 2.1 and Figure 2.5 for the training sets. As the proportion of training size increases from 10% through 50% of the total population, training errors increase for all methods. SBML outperforms other methods and yields the smallest training errors across different proportions of training set (0.121, 0.235, 0.285 and 0.353 for 10%, 20%, 25% and 50%, respectively).

Among other methods (FLM, OLM, RR, MMRM, SVM and RF), RF method seems to yield the 2nd smallest training errors, across different proportions of training set. However, it should be noted that, regardless of differences in the training errors, the predictability and accuracy of the methods, measured by the ‘testing’ errors (not ‘training’ errors), are the key in the context of machine learning approaches.

Table 2.2 Testing Errors vs. Proportion of Training Set - CF Data

Proportion of training set (%)	FLM	OLM	RR	MMRM	SVM	RF	SBML
10 ($N_{train} = 26$)	1.427	1.452	1.349	1.39	1.326	1.264	1.350
20 ($N_{train} = 53$)	1.306	1.348	1.295	1.30	1.263	1.192	1.246
25 ($N_{train} = 66$)	1.261	1.286	1.264	1.26	1.228	1.156	1.185
50 ($N_{train} = 132$)	1.217	1.230	1.218	1.21	1.140	1.050	1.014

Note: For SBML, $S^0 = 0.5$ (except for 10%, $S^0 = p$ -values), $\lambda = 0.5$ (except for 50%, $\lambda = -0.5$).

Figure 2.6 Testing Errors vs. Proportion of Training Set - CF Data

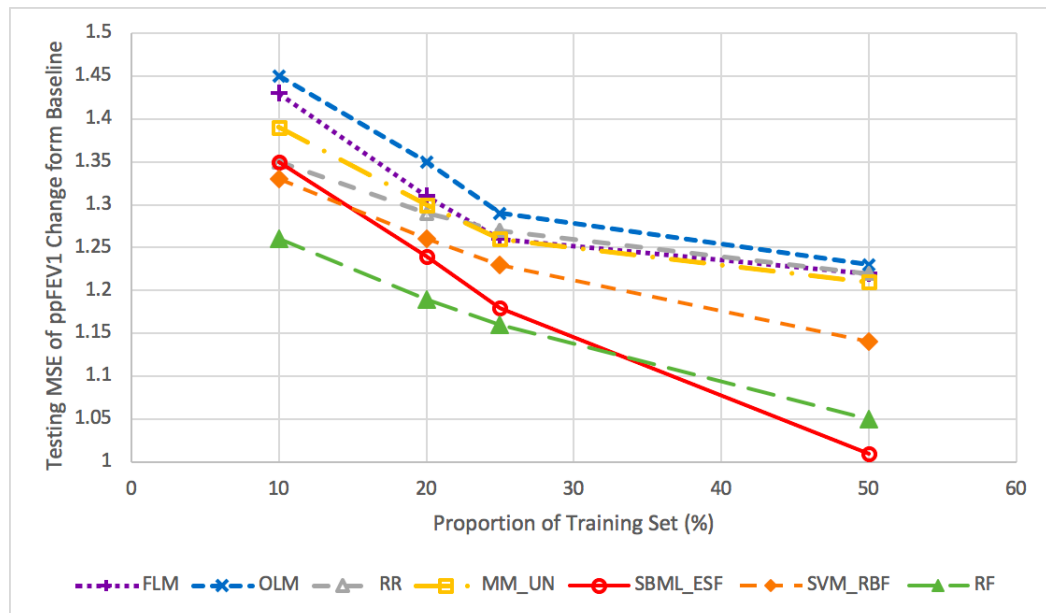


Table 2.2 summarizes the testing errors, and the corresponding plot is shown in Figure 2.6. In contrast to the training errors, the testing errors tend to decrease as the

proportion of the training size increases. Overall, in most cases, SBML performs better than other methods across different training sizes, for about 3-7% smaller than other methods, except for RF when the proportion of training size is small (i.e., $\leq 25\%$).

For example, when the proportion of the training size is 50%, the testing error for SBML is the smallest (1.014), followed by RF (1.05) and SVM (1.14), while the testing errors for the classical methods range from 1.21 to 1.23. When the proportion of the training size decreases to 25%, the testing errors increase for all of the methods (1.185 for SBML, 1.156 for RF, 1.228 for SVM, and for other classical methods ranging from 1.26 to 1.286). When the proportion of the training size decreases to 10%, the testing errors further increase to 1.35 for SBML, 1.264 for RF, 1.326 for SVM, and for other classical methods ranging from 1.39 to 1.452.

It is noteworthy that RF is an ensemble method that consists of many decision trees (i.e., default R package uses 500 trees) and the tuning parameters are computed using computerized cross-validation to optimize the prediction, therefore resulting in a vast improvement in accuracy. Whereas due to limited resources (e.g., time constraints), SBML uses manual cross-validation, and therefore the tuning parameters may not be optimized as the tuning parameters for SBML have not yet been fully explored.

Another interesting point is that MMRM is expected to predict better as it utilizes outcome measures at other time points that are close to the final visit. However, the results show no advantage in prediction. Instead, SBML without using measures from other visits except baseline produces better results regardless of the proportion of the training size. Furthermore, as we increase training size, testing error for SBML

dramatically decreases, and the gap of testing MSEs between SBML and other methods gets larger, except for RF.

2.4 Simulation Examples

We explore the effects of different parameters on the predictability of SBML, including similarity functions (SF), training size (N_{train}), penalty coefficient (λ), initial similarity scores (S^0) and the maximum number of iterations (*epoch*). We also evaluate the performance of SBML and compare against other classical statistical methods, including full linear model (FLM), optimal linear model (OLM) and the ridge regression (RR), as well as the two popular machine learning methods.

2.4.1 Simulation Data and Setting

Suppose we generate a simulated standard multivariate normal (MVN) data with 7 independent attributes, X_1 through X_7 , and let the true outcome function be defined as

$$Y_1 = X_1 + X_2X_3 + X_4^2 + X_5.$$

If we include all 7 variables (X_1 through X_7) as predictors in the model, then the FLM or ridge regression model follows the following form:

$$f(x) = \beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_3 + \beta_4X_4 + \beta_5X_5 + \beta_6X_6 + \beta_7X_7,$$

which implies that X_6 and X_7 , that are included in the analysis (but were not included in the true outcome function), are purely noise variables.

If we only include variables X_2 through X_6 as predictors in the model (i.e., we exclude X_1 from the analysis), then the FLM or ridge regression model takes the following form:

$$f(x) = \beta'_0 + \beta'_2X_2 + \beta'_3X_3 + \beta'_4X_4 + \beta'_5X_5 + \beta'_6X_6,$$

which implies that X_7 is an important yet unmeasured predictor since it is included in the calculation of true outcome function, and X_6 , that is included in the analysis model (but not included in the true outcome function), is purely noise.

One hundred sets of random samples from the distribution are drawn for each scenario of the parameters. We assume no correlation among the attributes ($r = 0$) and there are no missing data. Three different training sizes ($N_{train} = 60, 120, 240$) and a fixed testing size (N_{test}) of 50 are used; the reason we fix the testing size is to evaluate the effect of training size through different scenarios. Then we analyze the data and evaluate model performance on prediction for different parameters and methods (e.g., FLM, OLM, SVM, RF and SBML). For SBML, we use a learning rate (α) of 0.125.

2.4.2 Effect of Similarity Functions

To understand the effect of different similarity functions (SFs) for SBML on the testing MSEs, we evaluate the performance of three different similarity functions (SFs): exponential (ESF), logistic-alike (LSF), distance-inverse (DSF), as described in *Step 5* in Section 2.2.4. For SBML, we use p-values as initial similarity scores (S^0), a learning rate (α) of 0.125 and a penalty coefficient (λ) of 1. Among the 3 similarity functions, ESF produced the smallest testing errors across various training sizes, LSF generated similar results to those of ESF, and DSF produced the largest testing errors (Table 2.3). When there are 60 subjects in the training set, the testing errors for SBML are 0.784, 0.788 and 0.906, based on ESF, LSF and DSF, respectively, ESF yielding the smallest testing errors. The same trend is observed when there are 120 or 240 subjects in the training set. Another important trend is that as we decrease the training size, the testing errors for

SBML decrease for both ESF and LSF, whereas it increases for DSF. Therefore, for the rest of this work, further simulations are performed based on ESF to investigate the effects of other parameters on testing errors for SBML.

Table 2.3 Effect of Similarity Functions on MSEs for SBML

N_{train}	SBML _{ESF}	SBML _{LSF}	SBML _{DSF}
60	0.784	0.788	0.906
120	0.786	0.788	0.888
240	0.791	0.791	0.869

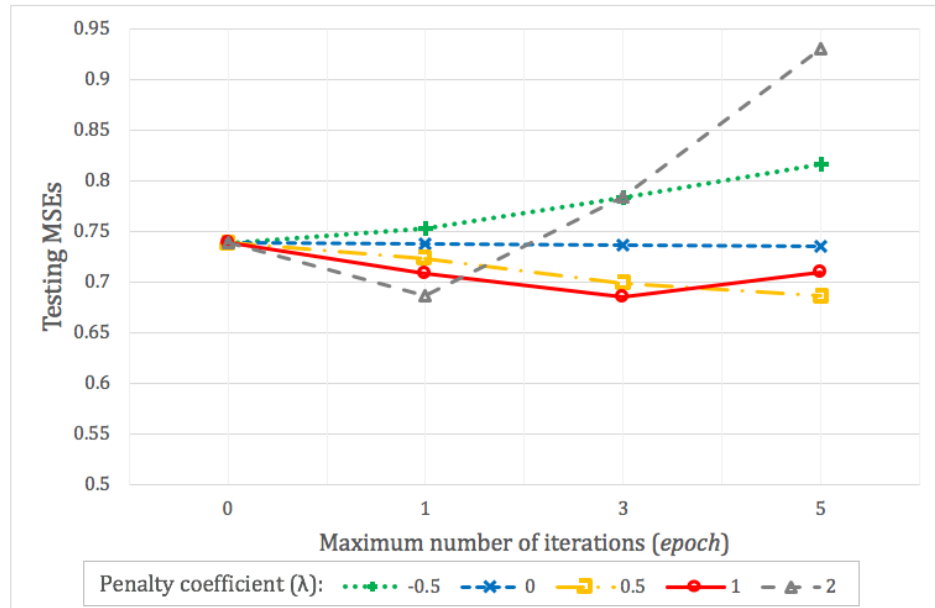
2.4.3 Effects of Tuning Parameters

Learning in SBML involves and may vary by tuning parameters such as initial similarity scores (S^0), a learning rate (α), a penalty coefficient (λ) and maximum iterations of learning (*epoch*). To study the effect of *epoch* and λ , we first choose p-values from FLM as the initial similarity scores (S^0), a training size (N_{train}) of 60 and a testing size (N_{test}) of 50 in the simulations. According to Table 2.4 and Figure 2.7, a positive value of λ (0.5, 1, 2) seems a good choice, and a smaller λ requires a larger *epoch*. As an example, the testing error for SBML reduces from 0.739 (for *epoch* = 0) to 0.686 with learning (for *epoch* = 5 and $\lambda = 0.5$). In contrast, a negative value of λ (-0.5) leads the testing error to increase as we increase the maximum number of iterations (*epoch*).

Table 2.4 Effect of Tuning Parameters on MSEs for SBML with P-values as Initial Similarity Scores

<i>epoch</i>	$\lambda = -0.5$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
0	0.739	0.739	0.739	0.739	0.739
1	0.753	0.738	0.723	0.709	0.687
3	0.784	0.737	0.699	0.685	0.784
5	0.817	0.735	0.686	0.710	0.931

Figure 2.7 Effect of Tuning Parameters on MSEs for SBML with P-values as Initial Similarity Scores



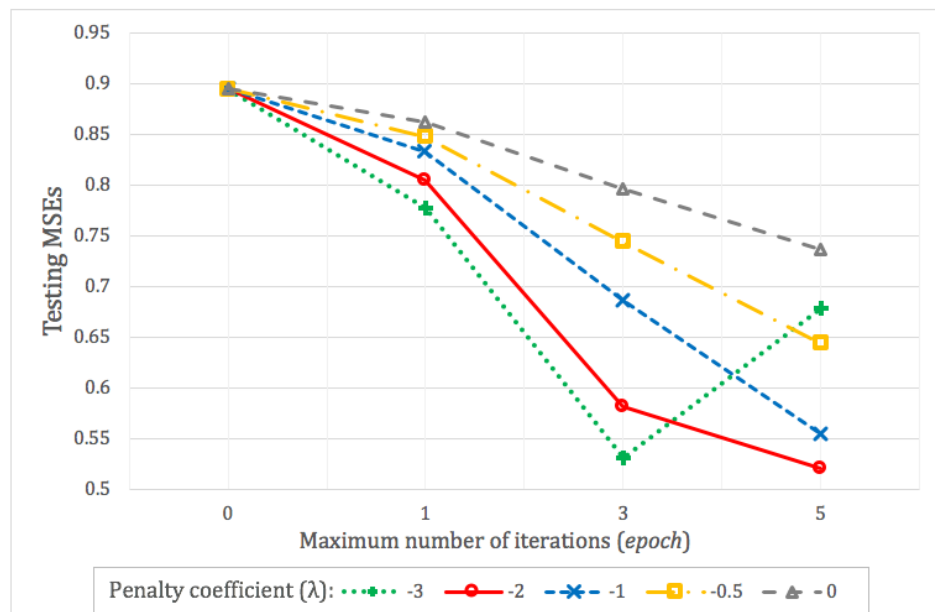
Previously we saw that the *epoch* seems to have minor impact on the testing error for SBML using p-values as the initial similarity scores; although, when $epoch = 0$, SBML usually provides better results compared to other methods.

To study the effect of initial similarity scores, we change them from p-values to a constant similarity scores of 0.5 for all the predictors, i.e., $S^0 = (0.5, 0.5, \dots, 0.5)$. The results are summarized in Table 2.5 and shown in Figure 2.8.

In contrast to the results with p-values as the initial similarity scores (Table 2.4), the testing errors for SBML with a constant value of 0.5 as the initial similarity scores (Table 2.5) dramatically improve through the learning process (*epoch* increases from 0 to 5). Especially, for negative values of penalty coefficient ($\lambda = -3$), a local minimum of the testing error has been reached at $epoch = 3$, while for other three values of penalty coefficient ($\lambda = -2, -1, -0.5, 0$), a local minimum of the testing error has not been reached yet – more iterations can be performed but can be time-consuming.

Table 2.5 Effect of Tuning Parameters on MSEs for SBML with Constant Initial Similarity Scores

<i>epoch</i>	$\lambda = -3$	$\lambda = -2$	$\lambda = -1$	$\lambda = -0.5$	$\lambda = 0$
<i>0</i>	0.895	0.895	0.895	0.895	0.895
<i>1</i>	0.778	0.805	0.833	0.848	0.862
<i>3</i>	0.531	0.581	0.686	0.744	0.796
<i>5</i>	0.679	0.520	0.555	0.644	0.737

Figure 2.8 Effect of Tuning Parameters on MSEs for SBML with Constant Initial Similarity Scores

According to Table 2.4 and Table 2.5, for $epoch = 0$, p-values as the initial similarity scores initially provide a smaller testing error of 0.739, compared to the constant value of 0.5 as the initial similarity scores (0.895). However, through the learning process (i.e., as we increase $epoch$), the best result from Table 2.4 is 0.685 (based on $S^0 = p\text{-values}$, $\lambda = 1$, $epoch = 3$) and the best result from Table 2.5 is 0.52 (based on $S^0 = 0.5$, $\lambda = -2$, $epoch = 5$).

To summarize, 1) if we use p-values as the initial similarity scores for SBML, learning may not help reduce the testing errors; however, if we have resource constraints

(e.g., time, computing power, etc.), using p-values as the initial similarity scores without any iteration of learning may provide reasonable predictions, although it may not achieve the best performance; 2) if we use a constant value of 0.5 as the initial similarity scores for SBML, (although we start with higher testing errors in the beginning of iterations), learning dramatically helps reduce the testing errors, and the best performance may be achieved when the optimal values of the tuning parameters (especially λ and *epoch*) are determined using cross-validation.

2.4.4 Effect of Training Size

To investigate the effect of the training size (N_{train}), we initially run simulations with different training sizes (60, 120, 240), given a testing size (N_{rest}) of 50, a constant value of 0.5 for the initial similarity scores (S^0) for all predictors, penalty coefficient (λ) of -0.5, with different maximum number of iterations ($epoch = 0, 2, 5, 10, 15$) and using the exponential similarity function (ESF) for SBML. Suppose the true outcome function is

$$Y_1 = X_1 + X_2X_3 + X_4^2 + X_5,$$

and suppose we only include X_2 through X_7 in the model (that is, X_1 is an unmeasured predictor, where X_6 and X_7 are 2 noise variables).

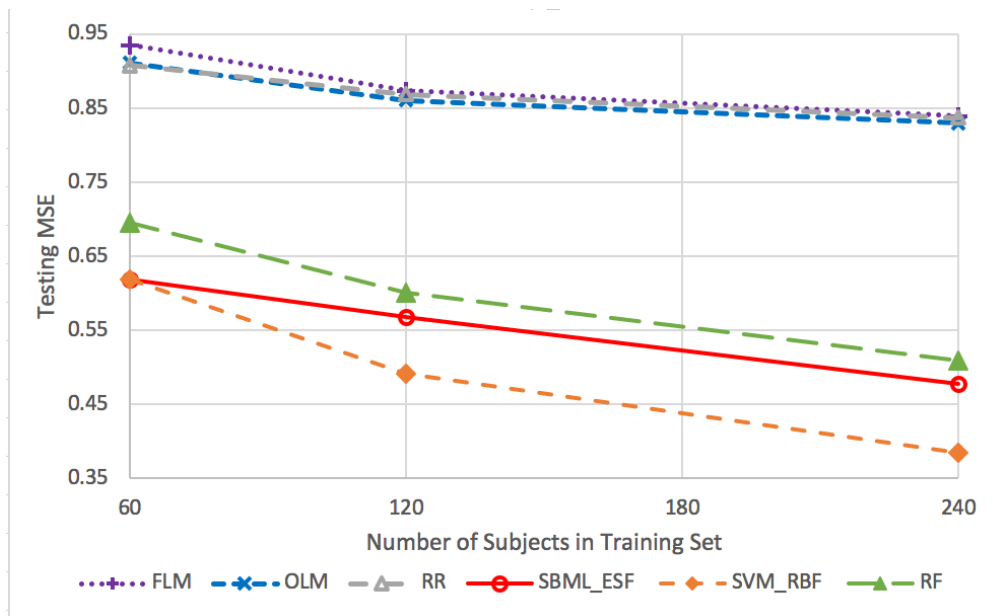
Table 2.6 and Figure 2.9 show the testing MSEs for the simulation across different methods. Given the training size is 240, SBML yields the smallest testing errors (0.478 with $epoch = 15$) compared to other methods, except SVM method (0.84, 0.83, 0.838, 0.385, 0.655 and 0.509 for FLM, OLM, RR, SVM, decision tree (DT) and RF methods, respectively). That is, SVM achieves the best performance when training size is large, followed by SBML, RF and then by DT and other traditional methods.

Decreasing the training size to 60, both SBML and SVM achieve the smallest testing errors (0.619 with $epoch = 10$) compared to other methods (0.936, 0.912, 0.908, 0.933 and 0.695 for FLM, OLM, RR, DT and RF methods, respectively).

Table 2.6 Effect of Training Size on MSEs

N_{train}	FLM	OLM	RR	SVM	DT	RF	SBML
60	0.936	0.912	0.908	0.619	0.933	0.695	0.619
120	0.874	0.861	0.869	0.491	0.779	0.601	0.568
240	0.840	0.830	0.838	0.385	0.655	0.509	0.478

Figure 2.9 Effect of Training Size on MSEs



In other words, when we increase the training size, say from 60 to 240, we dramatically gain prediction accuracy by 23%, 38%, 27% and 30% for SBML, SVM, RF and DT methods, and for the traditional methods (e.g., FLM, OLM, RR), we only achieve less than 10% as we increase the training size from 60 to 240.

Importantly, DT method loses precision quickly (i.e., MSE increases from 0.655 to 0.933) as the training size reduces; it seems to perform well compared to traditional

methods, when the training size is large (120, 240), but poorly when training size reduces to 60. This overfitting issue is the main reason that random forest (RF) or boosting/bagging techniques are preferred over the single tree method.

In summary, the prediction accuracy increases as we increase the training size across all methods, and SBML performs well compared to other traditional methods. Under the current setting, SBML (*epoch* = 10 or 15) eventually yields smaller testing errors than other traditional methods including RF method, regardless of the training size, and SVM yields the smallest testing errors among other methods when the training size is large.

2.4.5 Effect of Outcome Functions

To understand the effect of true outcome functions (Y) on the testing errors, we compare several true outcome functions as follows, keeping the same setting as in Section 2.4.4:

$$Y_1 = X_1 + X_2X_3 + X_4^2 + X_5$$

$$Y_2 = X_1 + X_2^2 + 2\sin(X_3)X_4^2 + X_5$$

$$Y_3 = X_1 + X_2^2 + 2\sin(X_3)X_4^2 + X_5^4$$

Table 2.7 Effect of Outcome Functions on MSEs

N_{train}	Y	FLM	OLM	RR	SVM	DT	RF	SBML
60	Y_1	0.936	0.912	0.908	0.619	0.933	0.695	0.619
	Y_2	0.866	0.856	0.845	0.727	0.962	0.723	0.683
	Y_3	1.181	1.152	1.079	0.778	0.844	0.737	0.464
120	Y_1	0.874	0.861	0.869	0.491	0.779	0.601	0.568
	Y_2	0.815	0.81	0.815	0.615	0.838	0.654	0.642
	Y_3	1.067	1.050	1.025	0.735	0.680	0.653	0.425
240	Y_1	0.840	0.830	0.838	0.385	0.655	0.509	0.478
	Y_2	0.749	0.742	0.755	0.494	0.664	0.498	0.546
	Y_3	1.008	1.007	1.007	0.745	0.635	0.626	0.423

Table 2.7 summarizes the effect of outcome functions on prediction accuracy given 3 different training sizes. Given the training size is 240, SVM seem to provide the smallest testing errors (0.385 and 0.494) for the outcome functions Y_1 and Y_2 , followed by SBML and RF; but when the nonlinearity increase (i.e., Y_3), SBML yields the smallest testing error of 0.423, and the testing error for SVM dramatically increase to 0.745. Similar trend is observed for the training size of 120. When the training size reduces to 60 and as we increase nonlinearity, SBML outperforms other methods by providing the smallest MSEs of 0.619, 0.683 and 0.464 for the outcome functions Y_1 , Y_2 and Y_3 , respectively. That is, SBML can handle nonlinear data better than other methods under various scenarios, except for SVM with a large training size.

2.4.6 Effect of Noise Variables

In this section, we evaluate the effect of noise variable(s), and to mimic the real-world setting, given the same setup as in Section 2.4.5 with the true outcome function as

$$Y_3 = X_1 + X_2^2 + 2\sin(X_3)X_4^2 + X_5^4,$$

we compare 2 cases: 1) same setting as in Section 2.4.4, where X_1 is an unmeasured predictor and X_6 and X_7 are 2 noise variables, and 2) with 5 additional noise variables added in the model. For instance, FLM takes the following forms for the two cases:

$$\text{Case 1: } f_1(x) = \beta_0 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7$$

$$\text{Case 2: } f_2(x) = \beta'_0 + \beta'_2 X_2 + \beta'_3 X_3 + \beta'_4 X_4 + \beta'_5 X_5 + \beta'_6 X_6 + \dots + \beta'_{12} X_{12}$$

The testing results in Table 2.8 show that given 3 training sizes, as we increase the number of noise variables from 2 to 7, all methods lose prediction accuracy, although

the degree may be slightly different. For instance, testing errors for RR and DT methods are robust and stable, regardless of number of noise variables given a training size.

SBML also yields robust and consistent testing errors, except when the size is 60, where the testing error for SBML increases from 0.633 to 0.749. Given the training size is 240, as we increase the number of noise variables from 2 to 7, the testing error for SBML increases only by 0.027 (from 0.251 to 0.278). In contrast, SVM seems to be sensitive to the number of noise variables, regardless of training size; the testing error for SVM increases from 0.809 to 0.922, from 0.711 to 0.835, and from 0.615 to 0.705, for the training size of 60, 120 and 240, respectively.

Table 2.8 Effect of Noise Variables on MSEs

N_{train}	# Noise Variables	FLM	OLM	RR	SVM	DT	RF	SBML
60	2	1.309	1.278	1.145	0.809	0.969	0.783	0.633
	7	1.437	1.354	1.139	0.922	1.007	0.846	0.749
120	2	1.175	1.158	1.091	0.711	0.714	0.608	0.357
	7	1.257	1.211	1.095	0.835	0.715	0.679	0.395
240	2	1.039	1.034	1.018	0.615	0.536	0.472	0.251
	7	1.064	1.047	1.020	0.705	0.536	0.542	0.278

FLM and OLM also tend to be a bit sensitive to the number of noise variables the model includes, when training size is 60 and 120. For example, when the training size is 60, the testing error for FLM increases from 1.309 to 1.437, and for OLM, it increases from 1.278 to 1.354 – FLM and OLM provide more robust and stable testing error when size is 240; both FLM and OLM yield the testing errors of 1.03 with 2 noise variables and around 1.05-1.06 with 7 noise variables.

Most importantly, SBML consistently outperforms and achieves the best prediction accuracy compared to other methods including SVM and RF methods, even in the presence of 5 additional noise variables.

2.5 Summary

In this chapter, we 1) proposed a new machine learning (ML) approach based on the *similarity principle*, 2) introduced the attribute-scaling factors to reflect the relative importance of each attribute, 3) derived the formulation for partial derivatives for the learning using the gradient method, and 4) showed through extensive simulations that SBML performing reasonably well under various settings, which include rare disease trials. In kernel methods, the similarity is subjectively predefined. In contrast, SBML defines the similarity objectively through attribute-scaling factors by using the data. Therefore, SBML produces better predictions when we have small training data or when the data is more complex.

The proposed SBML algorithm has some distinct features. In contrast to other ML methods that require big data, SBML can consistently produce reliable and robust prediction against other methods, even with small training data. When data displays complex structures or nonlinearity, SBML shows even better results than other methods we have discussed in this chapter. In training SBML, we recommend normalizing the data for predictors and for outcomes that are continuous, so the tuning parameters to be determined can be applied to different dataset, regardless of different measurement units. In addition, we suggest the following tuning parameters for normalized data: a constant value of 0.5 as the initial similarity scores for all predictors, a learning rate (α) around

0.125, a penalty coefficient (λ) ranging from -2 to 2, and the maximum number of iterations (*epoch*) ranging from 5 to 20. The tuning parameters can be determined via cross-validation. If you have limited resources such as computing power or time, p-values from FLM can be used as the initial similarity scores; it can produce a reasonably reliable result for SBML without further learning (i.e., updating the attribute-scaling factors).

SBML is a novel ML approach that has great potential in improving prediction accuracy that can also be applied to many other fields, not only drug development, and it can be beneficial at the exploratory stage, when we do not know which variables to include in the model or whether the relationship is linear or nonlinear. Further improvements can be made for this method. We hope this work attracts more people to SBML and other AI or ML approaches in clinical trials.

3. SIMILARITY-PRINCIPLE-BASED MACHINE LEARNING WITH BINARY OUTCOMES

3.1 Introduction

In Chapter 2, we discussed SBML for continuous outcomes. However, there are many situations where the outcomes are binary. Many variables in the field of medicine or biology are binary in nature. For instances, tumor response is often a variable of interest in oncology, whereas in cardiovascular diseases (CVD), the event of death or myocardial infarction (MI) in 30 days are measured as an outcome. Similarly, in mycosis or other infections, we may be interested in cured or not cured, finally, many diagnosis tests (e.g., COVID-19 or pregnancy to name a few) results are binary (positive or negative).

Logistic regression is one of the most commonly used methods for the analysis and predictions when outcomes are binary. According to Smith et al. (1988), standard statistical techniques such as discriminant analysis, regression analysis, and factor analysis have been used to provide this ability. However, even with the existence of hidden functional relationships that can provide forecasting ability, standard statistical techniques may be unsuccessful, and they may provide disappointing results when: 1) the sample size is small; 2) the form of the underlying functional relationship is not known; 3) the underlying functional relationships involve complex interactions and intercorrelations among a number of variables. These conditions are not unusual in medical problems.

Therefore, we will extend SBML to binary outcomes and compare its performance against the logistic model as well as other machine learning (ML) methods, such as support vector machine (SVM) with radial basis function (RBF) kernel and random forest (RF) methods, when applicable. The details of such ML methods can be found in Chapter 1.

In this chapter, we evaluate the effects of various tuning parameters on the predictability of SBML including true outcome functions (Y) to assess nonlinearity, correlations among attributes (r), training size (N_{train}), initial similarity scores (S^0), maximum number of iterations/learning ($epoch$), penalty coefficient (λ), noise variables and unmeasured predictors.

We also evaluate the performance of SBML and compare it with other classical statistical methods, including full logistic model (FLM), optimal logistic model (OLM) using stepwise BIC, and other machine learning (ML) methods such as SVM with RBF kernel and RF methods for binary outcomes, if applicable.

The metrics for evaluating the model performance for testing errors are mean squared errors (MSEs) for predicted probabilities and misclassification error rates (MCEs) for binary events (yes/no). Depending on the outcome or disease indications, we can use sensitivity (True Positive Rate, $P(T+|D+)$) and specificity (True Negative Rate, $P(T-|D-)$). If our primary goal is to correctly identify positives, we should choose a method with the highest sensitivity. On the other hand, if the goal is to identify negatives, we should put more emphasis on specificity. To be conservative, we use misclassification error rates (MCEs), which is the sum of both false positives and false negatives.

3.2 SBML Methods and Algorithms

The methods and algorithms for the binary outcome are analogous to those for continuous outcomes. If we are interested in predicting probabilities of the event, then mean squared errors (MSEs) can be used as a testing error metric. Otherwise, if we are interested in the predicted event (yes/no), misclassification error rates (MCEs) can be used as a testing error metric. For other error measures such as *entropy*, and *GINI-index*, the efficient algorithms are not the focus of this work. The algorithms for MSE-based SBML are similar to the algorithms for continuous outcome (Section 2.2.4), except a few steps where they involve the predicted outcome \hat{Y}_i and error E in the *Steps* (7) and (8).

Step 1) Normalize the Training dataset as described in Section 2.2.2.

Step 2) Assign Initial Similarity Scores (S_k^0):

e.g., $S_k^0 =$ p-values from a regression model or arbitrary number between 0 and 1, say 0.5.

That is, if we have 5 attributes (X_1, X_2, X_3, X_4, X_5) in the model, then the vector for the initial similarity scores can be $(p_1, p_2, p_3, p_4, p_5)$, where p_k is a p-value for the k^{th} variable from the regression model, or it can be $(0.5, 0.5, 0.5, 0.5, 0.5)$.

Step 3) Determination / Estimation of Initial Scaling Factors (R_k^0) – Solve for R_k^0 based on initial similarity scores, e.g., $S_k^0 =$ p-values:

It is difficult to determine the attribute-scaling factors \mathbf{R} directly from prior knowledge or current data. It is much easier to calculate \mathbf{R} through the similarity scores S_{ij} follows.

When a pair of subjects (the k^{th} pair) is the same in terms of all the attributes under consideration, except one (the k^{th}) attribute, then the summation will disappear on the

right-hand side of Eq. (6). Then the attribute-scaling factor R_k can be solved explicitly; for instance, if we use ESF as in Eq. (9), R_k can be solved using Eq. (5) as follows

$$R_k^0 = \frac{-\ln(S_k^0)}{|X_{ik} - X_{jk}|} = \frac{-\ln(S_k^0)}{IQR(X_k)}, \quad k = 1, 2, \dots, K \quad (5)$$

where the superscript, “0”, indicates the initial values, $|X_{ik} - X_{jk}|$ can be substituted with the interquartile range (IQR) between 1st and 3rd quartiles of the k^{th} attribute, which can be considered as the difference between two typical subjects in the data regarding the corresponding k^{th} attribute. For this reason, we choose the initial K (real or virtual) pairs of subjects this way and determine their K initial similarity scores.

Step 3.1) When limited training data are available, we need to use our prior knowledge, which will be vague and difficult to specify. In other words, based on prior knowledge of similarities scores between some selected subjects, we solve for $\mathbf{R} = (R_1, R_2, \dots, R_K)$ using a defined similarity function: $\mathbf{R} = \text{function}(\mathbf{S})$, where $\mathbf{S} = (S_1, S_2, \dots, S_K)$.

Step 3.2) P-value-based similarity scores:

When some training data are available, we obtain p-values from a statistical model and assign these p-values to the initial similarity scores between the K selected subjects and then determine \mathbf{R} using the defined similarity function. Here we see that the p-values can be interpreted as similarity scores.

Step 4) Determination of Scaled Distance: Let d_{ij} be the scaled distance between 2 subjects (say i and j) as a function of the attribute-scaling factor R_k ($k=1, 2, \dots, K$):

$$d_{ij} = \left[\sum_{k=1}^K (v_{ijk})^\rho \right]^{\frac{1}{\rho}} = \left[\sum_{k=1}^K (R_k |X_{ik} - X_{jk}|)^\rho \right]^{\frac{1}{\rho}}, \rho = 1 \text{ or } 2 \quad (6)$$

where $v_{ijk} = R_k |X_{ik} - X_{jk}|$ is the scaled absolute difference of the k^{th} attribute between 2 subjects (i and j).

Step 5) Similarity Functions / Measures: Define the similarity score S_{ij} between 2 subjects (i and j) as a function of the scaled distance d_{ij} :

$$S_{ij} = S(d_{ij}), \quad \text{where } S_{ij} \in [0,1] \quad (7)$$

The similarity score ranges from 0 (completely different) to 1 (identical) inclusively. The common requirements for a similarity function $S_{ij}(d_{ij})$ are $S_{ij}(0) = 1$ and $S_{ij}(\infty) = 0$.

Example 1) Exponential Similarity Function / Score (ESF):

$$S_{ij} = \exp(-d_{ij}^\eta) \quad (8)$$

For ESF with $\eta=1$ and $\rho=2$, the similarity function in Eq. (8) becomes

$$S_{ij} = \exp\left(-\sqrt{\sum_{k=1}^K (R_k |X_{ik} - X_{jk}|)^2}\right), \quad \eta = 1 \text{ and } \rho = 2 \quad (9)$$

Example 2) Logistic-Alike Similarity Function (LSF):

$$S_{ij} = \frac{2}{1 + \exp(d_{ij}^\eta)}, \quad \eta > 0 \quad (10)$$

Example 3) Distance-Inverse Similarity Function (DSF):

$$S_{ij} = \frac{1}{1 + d_{ij}^\eta}, \quad \eta > 0 \quad (11)$$

Step 6) Determination of Weight: Define the weight W_{ij} between 2 subjects (i and j) as a function of the similarity score S_{ij} such that it is the normalized similarity score:

$$W_{ij} = \frac{S_{ij}}{\sum_{n=1}^N S_{in}} \quad (12)$$

Step 7) Prediction of Outcome:

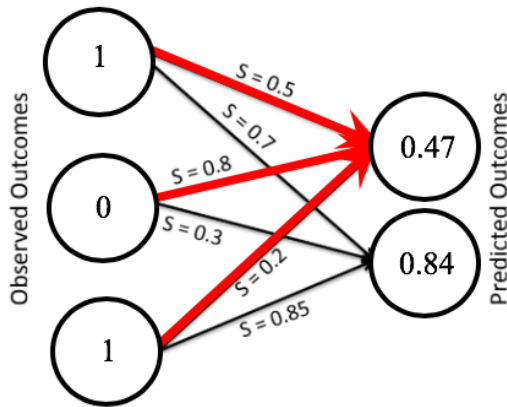
We can predict the probability of the outcome being 1 for the i^{th} subject:

$$\hat{Y}_i = \sum_{j=1}^N W_{ij} O_j, \quad i = 1, 2, \dots, N \quad (13)$$

where O_j is the observed binary outcome (i.e., takes either 1 or 0) for the j^{th} subject.

Figure 3.1 illustrates how the predicted outcomes of 2 nodes are calculated from the 3 observed outcomes using the similarity scores.

Figure 3.1 Predictions - Example of Similarity Principle in Action



$$\hat{Y}_1 = \frac{(1 * 0.5 + 0 * 0.8 + 1 * 0.2)}{(0.5 + 0.8 + 0.2)} = 0.47$$

Step 8) Calculation of Error:

As mentioned above, the error can be in terms of mean squared error (MSE) for classification probability, misclassification error rate (MCE), and *Gini-index*, but we focus on the error of classification probability as the gradient method can be easily used in the learning process. If the observed outcome O_i is 1 then the probability is 1; if the observed outcome O_i is 0, the probability is 0. These outcomes (in the probability sense)

will be compared with the predicted probability outcomes to determine the errors as elaborated below.

8.1) The error in terms of the mean squared error (MSE) for classification probability can be calculated using the following formulation:

$$E = \frac{\sum_{i=1}^N (\hat{Y}_i - O_i)^2}{N} = \frac{1}{N} \sum_{i=1}^N \left(\sum_{j=1}^N W_{ij} O_j - O_i \right)^2 \quad (14)$$

8.2) The error in terms of the misclassification error rate (MCE) for binary classification can be calculated using the following formulation:

$$MCE = \frac{\sum_{i=1}^N (\hat{Y}_i - O_i)^2}{N} = \frac{1}{N} \sum_{i=1}^N \left(\text{round} \left(\sum_{j=1}^N W_{ij} O_j \right) - O_i \right)^2 \quad (21)$$

Probabilistic error MSE is mathematically simpler because we can use the gradient method as we did for the continuous outcome. For simplicity, we will use MSEs as the error measure in our simulations. In some cases, we will also evaluate and compare misclassification errors (MCEs) across different methods and scenarios.

Step 9) Iterations of Learning:

Learning is basically updating the attribute-scaling factors \mathbf{R} . We can use the gradient method. To reduce the overfitting, we can minimize the *loss function* instead of MSE.

There are several commonly used loss functions available.

$$\text{Ridge Loss Function} = \rho_2(\mathbf{R}; \lambda) = E + \lambda \|\mathbf{R}\|_2^2 \quad (15)$$

where λ is a tuning parameter for the penalty term and $\|\mathbf{R}\|_2^2 = \sqrt{\sum_{k=1}^K R_k^2}$.

When $\lambda = 0$, the loss function reduces to MSE without the penalty term. The large value of λ puts a heavy penalty on large values of R_k .

Since $\frac{\partial \|\mathbf{R}\|^2}{\partial R_k} = 2R_k$, the derivative of the ridge loss function is as follows:

$$\frac{\partial \rho_2(\mathbf{R}; \lambda)}{\partial R_k} = \frac{\partial E}{\partial R_k} + \frac{\lambda \partial \|\mathbf{R}\|^2}{\partial R_k} = \frac{\partial E}{\partial R_k} + 2\lambda R_k \quad (16)$$

The gradient method to update the attribute-scaling factor R_k from iteration L to iteration L+1 can now be specified as follows:

$$R_k^{(L+1)} = R_k^{(L)} - \alpha \frac{\partial \rho_2(\mathbf{R}; \lambda)}{\partial R_k} \quad (17)$$

The derivative of error, $\frac{\partial E}{\partial R_k}$, in Eq. (16) can be derived for the exponential similarity function:

$$\left\{ \begin{array}{l} \frac{\partial E}{\partial R_k} = \frac{2}{N} \sum_{i=1}^N \left((\hat{Y}_i - O_i) \sum_{j=1}^N O_j \frac{\partial W_{ij}}{\partial R_k} \right), \text{ where} \\ \frac{\partial W_{ij}}{\partial R_k} = \frac{1}{\sum_{n=1}^N S_{in}} \frac{\partial S_{ij}}{\partial R_k} - \frac{S_{ij} \sum_{n=1}^N \frac{\partial S_{in}}{\partial R_k}}{(\sum_{n=1}^N S_{in})^2} \\ \frac{\partial S_{in}}{\partial R_k} = -\eta S_{in} d_{in}^{\eta-1} \frac{\partial d_{in}}{\partial R_k} \\ \frac{\partial d_{in}}{\partial R_k} = \left(\frac{R_k}{d_{in}} \right)^{\rho-1} |X_{ik} - X_{jk}|^\rho, d_{in} \neq 0; \text{ otherwise, } \frac{\partial d_{in}}{\partial R_k} = 0 \end{array} \right. \quad (18)$$

The iteration of learning stops when the loss function is minimized or when the pre-determined maximum number of iterations (*epoch*) is reached.

Step 10) Prediction of Future Outcome:

Once the attribute-scaling factors \mathbf{R} is determined using the training set, we can predict the outcome for the future subjects using Eq. (13).

To evaluate the prediction / performance of the method, define the error for the i^{th} subject in the test set as

$$E_i = (\hat{Y}_i - O_i)^2 \quad i = 1, 2, \dots, M \text{ in test set} \quad (19)$$

If there are M future subjects to predict (i.e., $i = 1, 2, \dots, M$ in the future or test set, while $j = 1, 2, \dots, N$ in the training set), the mean square error (MSE) is

$$E = \frac{\sum_{i=1}^M E_i}{M} = \frac{\sum_{i=1}^M (\hat{Y}_i - O_i)^2}{M} = \frac{1}{M} \sum_{i=1}^M \left(\sum_{j=1}^N W_{ij} O_j - O_i \right)^2 \quad (20)$$

where the weights W_{ij} are determined based on the similarity scores S_{ij} between the i^{th} subject in the test set and the j^{th} subject in the training set.

3.3 Simulations – General Settings

We simulate and construct the design matrix (X) from standard multivariate normal (MVN) data, first with 5 independent attributes, X_1 through X_5 , and then evaluate the effect of correlation among attributes on testing errors across different methods. We assume no correlation among the attributes ($r = 0$), unless specified otherwise.

We start with a true binary outcome function defined as the following, unless specified otherwise:

$$Y_1 = \{ \text{sgn}(X_1 + X_2 X_3 + X_4^2 + X_5) + 1 \} / 2$$

To evaluate the effect of nonlinearity, we run simulations and evaluate the prediction performance using the following true outcome function later in the chapter:

$$Y_3 = \{ \text{sgn}(X_1 + X_2^2 + 2 \sin(X_3) X_4^2 + X_5^4) + 1 \} / 2.$$

We begin with a set of random samples consisting of a training size of 40 and a testing size of 40, unless specified otherwise. Later in the chapter, we will run

simulations using training size of 60, 120 and 240, fixing the testing size as 50; this is to evaluate the effect of training size. Then we analyze the data and evaluate model performance on prediction for different parameters and methods. For the model specification, we begin with all of the 5 predictors included in the analysis. Then later in the chapter, we evaluate the effect of including more noise variables as well as the effect of unmeasured predictors on the testing errors across different methods.

The full logistic model (FLM) is specified as follows:

$$\ln(\text{odds}) = \ln \frac{p}{1-p} = \text{Logit } E[Y] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5,$$

where p is the probability of the event (i.e., $Y = 1$). If a noise variable is added for the analysis, then the above model specification will have an additional term (e.g., $\beta_6 X_6$ etc.). In the stepwise algorithm for optimal logistic model (OLM), Bayesian Information Criteria (BIC) was used.

For SBML, we focus on the exponential similarity function (ESF) in this chapter and begin with p-values from the full logistic model (FLM) as the initial similarity scores (S^0) – the reason is because the measure of similarity between 2 typical subjects can be viewed as new interpretation of p-values as described in Section 2.2.4.

When SBML involves learning in the training stage, we implement 1) a learning rate (α) between 0.1 and 1, however, in this work, we fix as 0.125; 2) a penalty coefficient (λ) ranging from -1 to 1; and 3) the maximum number of iterations (*epoch*) ranging up to 20. For SBML without learning (*epoch* = 0), the learning rate (α) and

penalty coefficient (λ) are meaningless. The simulation settings for this chapter can be any combination of these tuning parameters, and we assumed there are no missing data.

Setting #1 (for Section 3.4 – Section 3.9):

Description	Notation	Value for Setting #1
True outcome function	Y_1	$\{\text{sgn}(X_1 + X_2X_3 + X_4^2 + X_5) + 1\} / 2$
Model specification	$f(x)$	include all predictors (X_1 through X_5)
Correlation among attributes	r	0
Initial similarity scores for SBML	S^0	p -values, i.e., $(p_1, p_2, p_3, p_4, p_5)$
Maximum number of iterations	$epoch$	0
Training size	N_{train}	40
Testing size	N_{test}	40

Note: p_k is a p -value for the k^{th} variable from the full logistic regression model

Setting #2 (base setting for for Section 3.10):

Description	Notation	Value for Setting #2
True outcome function	Y_1	$\{\text{sgn}(X_1 + X_2X_3 + X_4^2 + X_5) + 1\} / 2$
Model specification	$f(x)$	include all predictors (X_1 through X_5)
Correlation among attributes	r	0
Initial similarity scores for SBML	S^0	a constant value of 0.5 for each attribute, i.e., $(0.5, 0.5, 0.5, 0.5, 0.5)$
Training size	N_{train}	60, 120, 240
Testing size	N_{test}	50
Penalty coefficient	λ	-1

3.4 Effect of Learning

To understand the effect of learning (*epoch*) for SBML on the prediction accuracy, we start with the Setting #1 specified in Section 3.3. The testing MSEs for SBML without learning (*epoch* = 0), OLM, and FLM are 0.734, 0.767, and 0.773, respectively, and SBML yields the smallest testing MSEs.

Table 3.1 summarizes the extensive simulation results of SBML with learning (*epoch* = 2, 3, 5) on testing MSEs under various values of the penalty coefficient (λ).

Table 3.1 Effect of Tuning Parameters on MSEs for SBML with P-values as Initial Similarity Scores

<i>epoch</i>	$\lambda = -1$	$\lambda = -0.5$	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 1$
0	0.734	0.734	0.734	0.734	0.734
2	0.761	0.742	0.733	0.738	0.760
3	0.792	0.751	0.733	0.746	0.791
5	0.873	0.779	0.733	0.769	0.860

Among the 5 different values of λ (ranging from -1 to 1, with an increment of 0.5), it seems $\lambda = 0$ yields the smallest testing MSE for SBML (0.733), regardless of the number of iterations. However, 0.733 is about the same as the testing MSE for SBML without learning (0.734), which is already smaller than the testing MSEs for FLM (0.773).

For λ of ± 0.5 and ± 1 , the testing MSEs seem to increase with learning (i.e., as we increase *epoch*). Furthermore, $\lambda = \pm 1$ provides much higher testing errors as we increase *epoch*. Therefore, we recommend λ to be somewhere close to 0 (say, with a range of -0.5 to 0.5) to reach the local minimum of testing MSEs, under Setting #1.

In conclusion, it is important to note that learning (i.e., increasing *epoch* from 0 to 2, 3, 5) does not necessarily help reduce the testing MSEs for SBML, when p-values are used as the initial similarity scores, regardless of λ values (e.g., almost no change for $\lambda =$

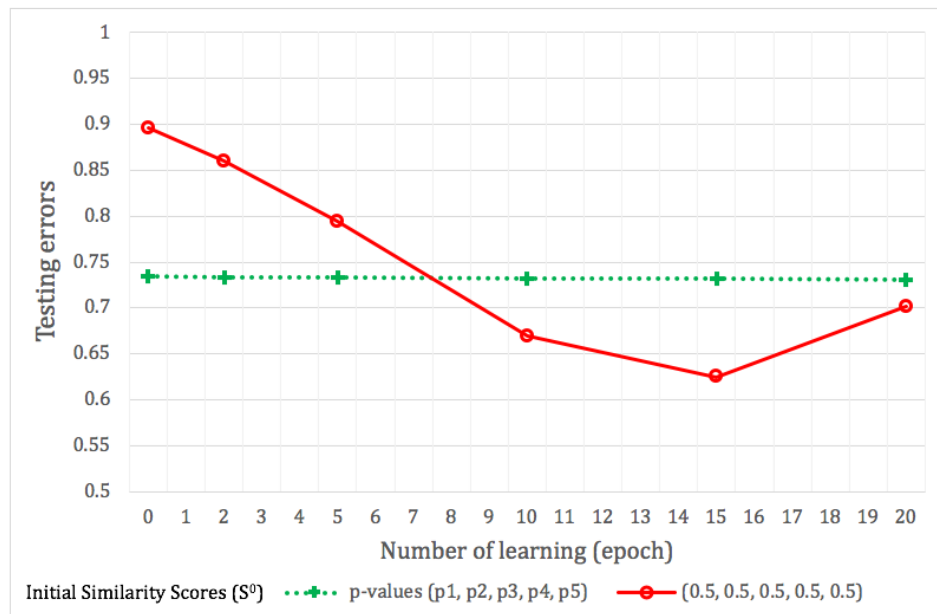
0, and learning makes predictions worse for other values of λ ; that is, λ only affects SBML with $epoch > 0$). In addition, when the absolute value of λ is greater than 0, the testing MSEs seem to increase as we increase the number of iterations under Setting #1.

Now, to evaluate the effect of learning for SBML on the testing MSEs, we slightly modify Setting #1. That is, we use a constant value of 0.5 (instead of p-values from FLM) as the initial similarity scores. Here we use a penalty coefficient (λ) of -0.5.

Table 3.2 Effect of Learning on MSEs for SBML with Constant Initial Similarity Scores

<i>epoch</i>	SBML
<i>0</i>	0.896
<i>2</i>	0.860
<i>5</i>	0.794
<i>10</i>	0.670
<i>15</i>	0.625
<i>20</i>	0.702

Figure 3.2 Effect of Learning on MSEs for SBML



The results from Table 3.2 and Figure 3.2 show that as we increase the number of iterations (*epoch*), the testing MSE for SBML decreases (0.896 for *epoch* = 0 to 0.625 for *epoch* = 15), and then it increases to 0.702 when the number of iterations increases from 15 to 20. That is, SBML with learning eventually helps reduce the testing MSEs, when we use a constant value of 0.5, i.e., (0.5, 0.5, 0.5, 0.5, 0.5), as the initial similarity scores.

3.5 Effect of Penalty Coefficient

Previously, we evaluated the effect of learning for SBML on the testing MSEs, 1) using the p-values as the initial similarity scores and various values of λ (Table 3.1); and 2) using a constant value of 0.5 as the initial similarity scores and $\lambda = -0.5$ (Table 3.2). In this section, we evaluate the SBML prediction performance given three values of the penalty coefficient (λ), using a constant value of 0.5 as the initial similarity scores, assuming the rest of the parameters are the same as the Setting #1.

Table 3.3 Effect of Penalty Coefficient on MSEs for SBML

λ	<i>epoch</i> = 0	<i>epoch</i> = 2	<i>epoch</i> = 5
-0.5	0.896	0.860	0.794
0	0.896	0.893	0.889
0.5	0.896	0.923	0.953

As shown in Table 3.3, the negative value of penalty coefficient ($\lambda = -0.5$) for SBML yields the smallest testing MSEs of 0.794 for 5 iterations of learning. Also, given $\lambda = -0.5$, testing MSE decreases as we increase the number of iterations; however, with a positive value of penalty coefficient ($\lambda = 0.5$), learning makes the prediction worse (testing MSE rather increases from 0.896 to 0.923, then increases again to 0.953). Consequently, finding an appropriate value of penalty coefficient (λ) is critical as it can change the direction of testing MSEs through learning in the training stage, and it needs

to be determined by cross-validation. That is, penalty coefficient (λ) and the maximum number of iterations (*epoch*) are important tuning parameters for SBML with learning.

3.6 Effect of Initial Similarity Scores

As seen previously (Table 3.1, Figure 3.2), using p-values as the initial similarity scores and $\lambda = 0$ for SBML provided robust testing MSEs (0.73) regardless of the maximum number of iterations (*epoch* = 0, 2, 3, 5) and across 5 different values of penalty coefficient ($\lambda = -1, -0.5, 0, 0.5, 1$). Learning does not seem to improve prediction for SBML when the p-values from FLM are used as the initial similarity scores; whereas learning helps reduce the testing MSEs for SBML when a constant value of 0.5 is used as the initial similarity scores.

Table 3.4 Effect of Initial Similarity Scores on MSEs for SBML

<i>Initial Similarity Scores</i>	SBML
$(p_1, p_2, p_3, p_4, p_5)$	0.734
$(0.5, 0.5, 0.5, 0.5, 0.5)$	0.625

Table 3.4 shows the testing MSEs for SBML with *epoch* = 15 for the two types of initial similarity scores. Previously in Figure 3.2, when SBML uses a constant value of 0.5 as the initial similarity scores, we see the testing MSE yields much larger testing MSE (0.896) compared to that of SBML using p-values as the initial similarity scores (0.734) in the beginning of the training (i.e., without any learning, *epoch* = 0). As we increase the number of iterations (*epoch*) for SBML using a constant value of 0.5 as the initial similarity scores, testing MSE dramatically reduces and yields much smaller testing MSE (0.670 for *epoch* = 10, 0.625 for *epoch* = 15) compared to that of SBML using p-values as the initial similarity scores (0.734). However, learning can be

computationally extensive or expensive (i.e., takes too long for *epoch* larger than 10), depending on the training size, and more iterations do not necessarily guarantee minimum testing MSEs for SBML. Thus, using p-values as the initial similarity scores with *epoch* = 0 for SBML may help us save time with some tradeoffs of prediction accuracy.

3.7 Effect of Unmeasured Predictors

In this section, we investigate the effect of unmeasured predictors (i.e., excluding important attributes from the model specification), and the results are summarized in Table 3.5. Suppose we miss out (or do not collect) 2 important attributes (X_4, X_5), thus conducting analyses using only 3 important attributes (i.e., X_1, X_2, X_3). Then the testing MSE for SBML (using p-values as the initial similarity scores without learning) increases by 23% from 0.725 (with all 5 important features) to 0.893 (with only 3 important features). Similarly, the testing MSEs for OLM and FLM increase by 17% (from 0.758 to 0.885) and 15% (from 0.784 to 0.905), respectively, which is not surprising. With 2 unmeasured predictors, SBML yields smaller testing MSEs compared to FLM (0.893 vs. 0.905), and slightly larger compared to OLM (0.893 vs. 0.885), but the difference is minimal.

Table 3.5 Effect of Unmeasured Predictors on MSEs

<i>Models</i>	SBML	OLM	FLM
X_1, X_2, X_3, X_4, X_5	0.725	0.784	0.758
X_1, X_2, X_3	0.893	0.885	0.905

In summary, the testing MSEs for the model with 2 unmeasured predictors (X_4, X_5) are larger than the model with all 5 important predictors, which is expected. When

we correctly specify the model (i.e., select appropriate predictors), SBML makes a better improvement than FLM or OLM. However, this conclusion is drawn under the current setting, and different values of the tuning parameters can lead to different conclusions.

3.8 Effect of Noise Variables

To understand the effect of noise variable(s), we start with the Setting #1 and repeat the analysis by adding 1) 1 noise variable, 2) 2 noise variables, and 3) 10 noise variables. Table 3.6 and Figure3.3A summarize the simulation results.

Table 3.6 Effect of Noise Variables on MSEs

Number of Noise Variables	SBML	OLM	FLM
0	0.725	0.758	0.784
1	0.746	0.812	0.835
2	0.769	0.877	0.910
10	0.981	1.219	1.310

When the model includes all 5 important attributes with no noise variable, the testing MSE for SBML (0.725) is smaller than the testing MSEs for OLM or FLM (0.758 and 0.784, respectively). Then if we add 1 noise variable, then the testing MSE increases a little bit from 0.725 to 0.746 for SBML, and the testing MSEs increase from 0.758 to 0.812 for OLM and from 0.784 to 0.835 for FLM. Then if we add 1 additional noise variable for a total of 2 noise variables, then the testing MSE for SBML increases from 0.746 to 0.769, and the testing MSEs increase from 0.812 to 0.877 for OLM and from 0.835 to 0.91 for FLM. If we add 8 additional noise features for a total of 10 noise variables, then the testing MSE for SBML increases from 0.769 to 0.981, and the testing MSEs increase from 0.877 to 1.219 for OLM and from 0.91 to 1.31 for FLM. Thus, the trend is that testing MSEs for SBML are always smaller than the testing MSEs for OLM

and FLM, regardless of the number of noise variables. Also, the gap in the testing MSEs for SBML and OLM or FLM gets larger as we increase the number of noise variables.

Effect of Noise Variables with Unmeasured Predictors

In reality, there often exists unmeasured predictors. To further understand the joint effect of noise variables and unmeasured predictors (or in the absence of important features), we further analyze scenarios where only 3 important attributes (X_1, X_2, X_3) are included in the model (i.e., X_4 and X_5 are included in the true outcome function Y but are excluded in the model, and no noise variable is included in the analysis), then compared the predictions by adding 1 noise variable, 2 noise variables, and 10 noise variables (Table 3.7, Figure 3.3B).

Table 3.7 Effect of Noise Variables on MSEs with Unmeasured Predictors

Number of Noise Variables	SBML	OLM	FLM
0	0.893	0.885	0.905
1	0.903	0.908	0.941
2	0.908	0.939	0.992
10	0.959	1.304	1.407

Note: 2 unmeasured predictors (X_4 and X_5)

With only 3 important variables included in the base model (i.e., missing 2 important attributes X_4 and X_5), the testing MSE for SBML is 0.893, which is slightly smaller than FLM (0.905) but slightly larger than the one for OLM (0.885). OLM yields a smaller testing MSE compared to SBML. Then if we add 1 noise variable, the testing MSEs increase to 0.903, 0.908 and 0.941, for SBML, OLM and FLM, respectively. With only 1 noise variable, SBML beats OLM with a minimal difference. Then if we add an additional noise variable for a total of 2 noise variables, the testing MSEs increase to

0.908, 0.939 and 0.992, for SBML, OLM and FLM, respectively. With 2 noise variables, SBML starts to predict better compared to OLM and FLM. With 8 additional noise features for a total of 10 noise variables, the testing MSEs increase to 0.959, 1.304 and 1.407, for SBML, OLM and FLM, respectively. Again, SBML outperforms and predicts better than OLM and FLM.

Similar to the noise effect as seen previously with the base model with all 5 important attributes (i.e., assuming independent attributes without unmeasured predictors), the trend of noise effect with 2 unmeasured predictors is that testing MSEs for SBML are almost always smaller than the testing MSEs for OLM and FLM. The gap in the testing MSEs for SBML versus OLM or FLM gets larger as we increase the number of noise variables (Figure 3.3B), meaning that SBML is pretty robust regardless of the number of noise variables in the model.

Effect of Noise Variables with Correlation among Attributes

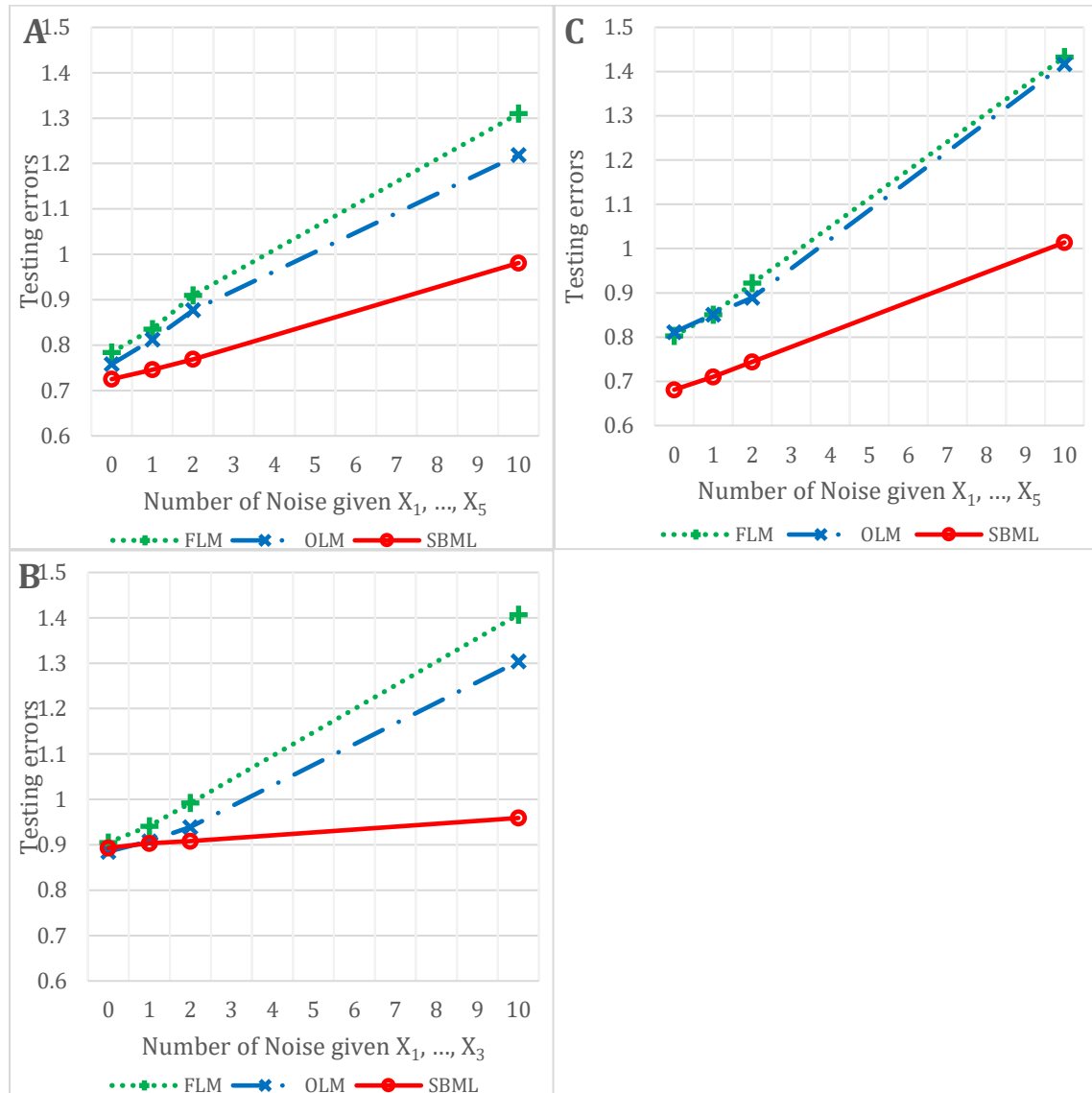
We will explore the effect of noise variables assuming the attributes are correlated, say with $r = 0.5$ (instead of $r = 0$). Using Setting #1, we will repeat the analysis by adding 1, 2, and 10 noise variables. Table 3.8 and Figure 3.3C summarize the simulation results (and we will compare to Table 3.6 and Figure 3.3A).

Table 3.8 Effect of Noise Variables on MSEs given Correlation

Number of Noise Variables	SBML	OLM	FLM
0	0.681	0.811	0.803
1	0.710	0.851	0.851
2	0.744	0.889	0.922
10	1.014	1.417	1.433

Note: $r = 0.5$

Figure 3.3 Effect of Noise Variables on MSEs



Note: Setting #1, (3.3A) $r = 0$, (3.3B) $r = 0$, (3.3C) $r = 0.5$.

Similar to the 2 previous cases, the testing MSE for SBML increases from 0.681 (with no noise variable) to 1.014 (with 10 noise variables); by adding 10 noise variables, the testing MSEs increase by 49%, 75% and 78% for SBML, OLM and FLM, respectively. We found that SBML yields the smallest testing error compared to OLM

and FLM, regardless of the number of noise variables. Also, SBML is less sensitive to the number of noise variables than other methods.

In summary, when we compare the testing MSEs from the scenarios Table 3.6 and Figure 3.3A to the testing MSEs from the scenarios Table 3.8 and Figure 3.3C, our findings are not surprising:

- 1) SBML yields the smallest testing MSEs, regardless of the number of noise variables and methods.
- 2) testing MSEs increase as we increase the number of noise variables for all methods.
- 3) SBML is less sensitive to the number of noise variables than other methods.
- 4) As we add mild correlations among attributes (r from 0 to 0.5), SBML yields smaller testing MSEs under almost all scenarios.

Different values of the tuning parameters may lead to different conclusions, and we expect the prediction may be much better for SBML when we use a constant value of 0.5 as the initial similarity scores with about 10 iterations of learning in the training stage.

3.9 Effect of Correlation Among Attributes

In reality, variables are often correlated to each other (e.g., if you're a taller male, you tend to weigh more than a shorter female). Thus, we investigate the performance or prediction based on 3 positive values of correlation structures for simplicity. For simplicity, with 5 attributes (X_1, X_2, X_3, X_4, X_5), we can have the correlation matrix Σ as follows,

$$\Sigma = \begin{pmatrix} 1 & r & r & r & r \\ r & 1 & r & r & r \\ r & r & 1 & r & r \\ r & r & r & 1 & r \\ r & r & r & r & 1 \end{pmatrix}, \text{ where } -1 \leq r \leq 1.$$

In this work, we only study the effect of 3 values of positive correlation coefficients.

Table 3.9 Effect of Correlations on MSEs

<i>Correlation coeff.</i>	SBML	OLM	FLM
0	0.725	0.758	0.784
0.5	0.681	0.811	0.803
0.75	0.628	0.877	0.838

Note: S^0 = p-values, no hidden predictor.

The results showing the effect of correlations are summarized in Table 3.9 and Figure 3.4A. Given Setting #1 ($r = 0$), the testing MSEs are 0.725, 0.758 and 0.784, for SBML, OLM and FLM, respectively. SBML performs better than the traditional approaches and yields the smallest testing MSE in this case. When the correlations among attributes increase to 0.5 (i.e., $r = 0.5$), the testing MSE for SBML decreases by 6% (to 0.681). In contrast, the testing MSEs for OLM and FLM increase by 7% and 2% (to 0.811 and 0.803), respectively. When the correlations among attributes further increase to 0.75 (i.e., $r = 0.75$), the testing MSE for SBML further decreases to 0.628, while OLM and FLM testing MSEs further increase to 0.877 and 0.838, respectively.

In summary, when we include all important attributes (X_1, X_2, X_3, X_4, X_5) in the model, SBML yields the smallest testing MSEs compared to OLM and FLM regardless of the different values of correlation coefficient we tried, and we observed that as we increase correlations among attributes, SBML further reduces testing MSEs, whereas in contrast, FLM and OLM increase testing MSEs.

Effect of Correlation When There Are Unmeasured Predictors

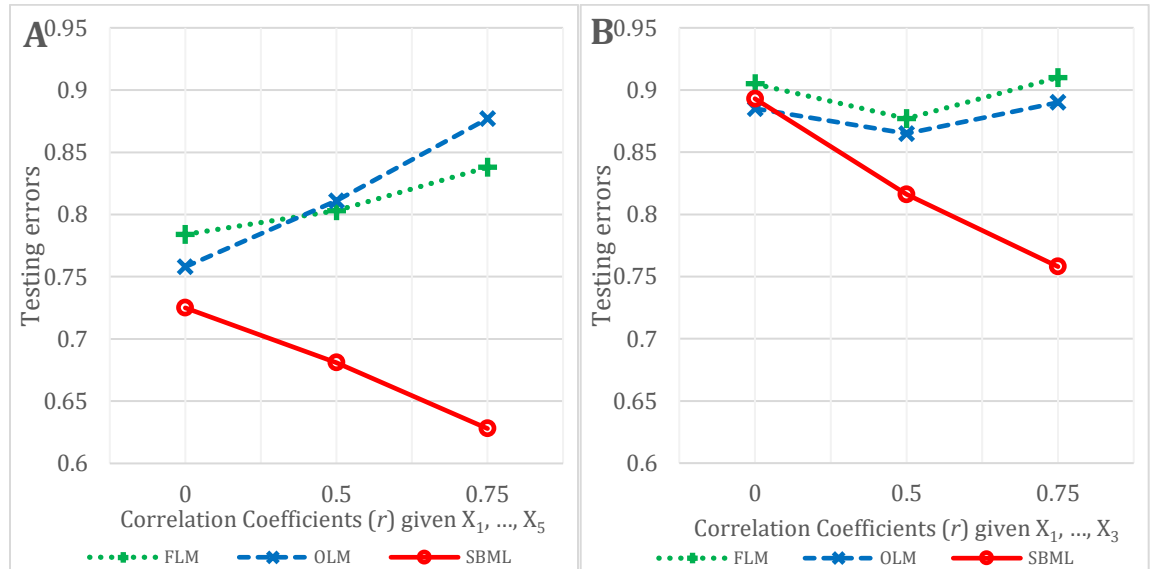
Similarly, we evaluate the effect of 3 positive values of correlation coefficient on the performance of testing MSEs, when we exclude 2 important attributes (X_4, X_5) from the model. Table 3.10 and Figure 3.4B show the results when we only use 3 important attributes (X_1, X_2, X_3) and have 2 unmeasured predictors (X_4, X_5).

Table 3.10 Effect of Correlations on MSEs with Unmeasured Predictors

<i>Correlation coeff.</i>	SBML	OLM	FLM
0	0.893	0.885	0.905
0.5	0.816	0.865	0.877
0.75	0.758	0.890	0.910

Note: S^0 = p-values, 2 unmeasured predictors (X_4, X_5)

Figure 3.4 Effect of Correlations and Unmeasured Predictors on MSEs



Note: S^0 = p-values, (3.4.A) no hidden predictor, (3.4.B) 2 hidden predictors

For $r = 0$, testing MSEs are 0.893, 0.885 and 0.905 for SBML, OLM and FLM, respectively. OLM yields slightly smaller testing MSEs in this case. When the correlations among the attributes increase to $r = 0.5$, the testing MSEs decrease to 0.816, 0.865 and 0.877 for SBML, OLM and FLM, respectively. There is a 9% deduction in

testing MSEs for SBML, whereas only a 2-3% deduction in testing MSEs for OLM and FLM, meaning SBML is the smallest testing MSE in this case. When the correlations among the attributes further increase to $r = 0.75$, the testing MSE for SBML further reduces to 0.758; while the testing MSEs for OLM and FLM increase to 0.89 and 0.91, respectively.

In summary, SBML yields the smallest testing MSEs compared to OLM and FLM, when there are moderate to strong correlations among the attributes, and an increase in the correlation reduces MSEs for SBML, while it increases MSEs for OLM and FLM. Furthermore, when there are 2 unmeasured predictors (X_4, X_5), the testing MSEs are higher in general than in the case in Table 3.9 where we include all 5 important attributes in the model, regardless of correlation and across different methods.

3.10 Comparisons of Different Machine Learning Methods

Previously, we used Setting #1 (e.g., a fixed training size of 40 and a testing size of 40, etc.). In this section, we run simulations based on Setting #2 (e.g., a training size of 60, 120 and 240, fixing a testing size at 50, etc.); the reason we fix the testing size is to study the effect of training size. Also, we use a constant value of 0.5 as the initial similarity scores ($S^0 = 0.5$) with learning for SBML, and only the best SBML testing errors will be reported among different values of the maximum number of iterations (*epoch* up to 15) and/or penalty coefficient ($\lambda = -1$ or -0.5). In addition, we compare SBML with 2 popular ML methods (e.g., SVM with RBF kernel and RF methods) as well as the traditional methods (e.g., FLM and OLM), and we evaluate both MSEs for

predicted probabilities of the event and misclassification errors (MCEs) for binary events (yes/no) as metrics for testing errors.

Setting #2 (base setting for Section 3.10):

- True outcome function: $Y_1 = \{ \text{sgn}(X_1 + X_2X_3 + X_4^2 + X_5) + 1 \} / 2$
- Model specification: include all predictors (X_1 through X_5)
- Correlation among attributes: ($r = 0$, i.e., independent)
- Initial similarity scores for SBML (S^0): a constant value of 0.5 for each attribute
- Maximum number of iterations (*epoch*): up to 20
- Penalty coefficient (λ): -1
- Training size (N_{train}) = 60, 120, 240; Testing size (N_{test}) = 50

Setting #3: same as Setting #2, except

- True outcome function: $Y_3 = \{ \text{sgn}(X_1 + X_2^2 + 2\sin(X_3)X_4^2 + X_5^4) + 1 \} / 2$

Setting #4: same as Setting #2, except

- Penalty coefficient (λ): -0.5
- Model specification: with 1 unmeasured predictor and 2 noise variables (i.e., model include X_2, \dots, X_7 , so it is missing X_1 and including X_6, X_7 instead)

MSEs

Table 3.11 shows the testing MSEs for Setting #2. Regardless of the number of training size, we observe that SBML yields smaller testing MSEs (0.622, 0.457, 0.401 for training size of 60, 120, 240, respectively) compared to those of FLM, OLM and RF methods, and that SVM with kernel method yields the smallest testing MSEs (0.553,

0.419, 0.362 for training size of 60, 120, 240, respectively). The difference in testing MSEs between SBML and SVM is about 10% regardless of the training size.

Table 3.11 Effect of Training Size on Testing MSEs - Setting #2

	Testing Errors (MSEs)				
N_{train}	FLM	OLM	SVM	RF	SBML
60	0.702	0.689	0.553	0.694	0.622
120	0.654	0.642	0.419	0.579	0.457
240	0.642	0.640	0.362	0.504	0.401

Second, we use Setting #3 – same as Setting #2, but we use Y_3 as the true outcome function instead of Y_1 (i.e., making it more nonlinear by adding sine functions, etc.).

When the nonlinearity increases, the testing MSEs with more nonlinearity in the outcome function Y_3 become larger (about 8% to 57% across all methods) than the testing MSEs for the other outcome function Y_1 in Setting #2, regardless of training size. Furthermore, SBML and RF get robust as we increase the training size, regardless of the true outcome functions; while the testing MSEs for FLM, OLM and SVM get much larger. For instance, for the training size of 240, the testing MSE for SVM increases from 0.362 (Setting #2) to 0.535 (Setting #3), whereas the testing MSE for SBML stays around 0.4.

Table 3.12 Effect of Training Size on Testing MSEs - Setting #3

	Testing Errors (MSEs)				
N_{train}	FLM	OLM	SVM	RF	SBML
60	0.902	0.886	0.765	0.749	0.728
120	0.849	0.838	0.659	0.665	0.603
240	0.794	0.789	0.535	0.554	0.463

Table 3.13 shows the testing MSEs for Setting #4 (same as Setting #2, but here, we consider a model with 1 unmeasured predictor and 2 noise variables, and λ of -0.5).

When the training size is 120 and 240, SVM yields the smallest testing MSEs (0.731 for training size of 120 and 0.677 for training size of 240), followed by RF and SBML.

When the training size decreases to 60, both SBML and RF yield the smallest testing MSEs of 0.855, followed by SVM with RBF kernel method (0.897). In conclusion, when we misspecify the model to miss 1 unmeasured predictor and include 2 noise variables instead, testing MSEs increase from 30% up to 80% (compared to Setting #2).

Table 3.13 Effect of Training Size on Testing MSEs - Setting #4

N_{train}	Testing Errors (MSEs)				
	FLM	OLM	SVM	RF	SBML
60	0.939	0.899	0.897	0.855	0.855
120	0.854	0.834	0.731	0.764	0.773
240	0.852	0.846	0.677	0.707	0.724

MCEs

Now, let's compare misclassification errors (MCEs) using binary events (yes/no) as the outcome using the same 3 cases we considered for MSEs (Settings #2, #3, #4). First, the testing MCEs for Setting #2 are presented in Table 3.14. Regardless of the number of training size, we observe that SBML yields smaller testing MCEs (0.198, 0.139, 0.123 for training size of 60, 120, 240, respectively) compared to those of FLM, OLM and RF methods, and that SVM yields the smallest testing MCEs (0.168, 0.111, 0.082 for training size of 60, 120, 240, respectively). The difference in testing MCEs between SBML and SVM decreases from 50% to 18% as we decrease the training size from 240 to 60.

Table 3.14 Effect of Training Size on Testing MCEs - Setting #2

	Testing Errors (MCEs)				
<i>N_{train}</i>	FLM	OLM	SVM	RF	SBML
60	0.219	0.215	0.168	0.215	0.198
120	0.216	0.210	0.111	0.172	0.139
240	0.207	0.209	0.082	0.144	0.123

Second, Table 3.15 shows the testing results for Setting #3. When the nonlinearity increases, unlike the MSEs, the testing MCEs for the binary event (yes/no) do not get much larger, except in SVM; for MCEs, the testing MCEs for FLM and OLM are about the same as the ones for the true outcome function (Y_1) in the 1st scenario (Setting #2), and the testing MCEs for the 2 ML methods, SBML and RF, gets smaller, with SBML yielding the smallest MCEs regardless of the training size. SBML yields the smallest testing errors (both MSEs and MCEs) when there are more nonlinear relationships in the data; whereas under previous Setting #2 (for Y_1), SVM yields the smallest testing errors (both MSEs and MCEs).

Table 3.15 Effect of Training Size on Testing MCEs - Setting #3

	Testing Errors (MCEs)				
<i>N_{train}</i>	FLM	OLM	SVM	RF	SBML
60	0.206	0.207	0.171	0.171	0.162
120	0.212	0.208	0.151	0.153	0.140
240	0.207	0.203	0.117	0.126	0.111

Third, Table 3.16 shows the testing MCEs for Setting #4. Regardless of the training size, we observe that SBML yields smaller testing MCEs (0.294, 0.262, 0.227 for training size of 60, 120, 240, respectively) compared to those of FLM and OLM, and that SVM yields the smallest testing MCEs (0.288, 0.235, 0.205 for training size of 60, 120,

240, respectively), followed by RF method. However, the testing MCEs of SBML, RF and SVM methods are about the same when the number of the training size is 60 or 240.

Table 3.16 Effect of Training Size on Testing MCEs - Setting #4

N_{train}	Testing Errors (MCEs)				
	FLM	OLM	SVM	RF	SBML
60	0.313	0.307	0.288	0.292	0.294
120	0.289	0.278	0.235	0.250	0.262
240	0.284	0.283	0.205	0.221	0.227

3.11 Summary

We showed how SBML can be flexibly extended to binary outcomes where we can treat 1) the predicted probability of the event as a continuous outcome and evaluate its performance with mean squared errors (MSEs); and/or 2) the event (yes/no) as a binary outcome and evaluate its prediction accuracy with misclassification errors (MCEs). When we use the binary event (yes/no) as the outcome, we can also use sensitivity or specificity instead of MCEs, which are the sum of both false positives and false negatives, depending on the disease indications.

When we use predicted probabilities as the outcome, with SBML using a constant value of 0.5 as the initial similarity scores, the negative values of penalty coefficient (e.g., $\lambda = -1, -0.5$) and about 10 to 20 iterations yielded the smallest testing MSEs compared to FLM and OLM. Given a constant value of 0.5 as the initial similarity scores for each of K predictors, SBML with learning eventually helped reduce the testing MSEs; given p -values as the initial similarity scores, SBML with learning did not reduce variance.

Consequently, finding appropriate values of tuning parameters is critical as the direction of testing MSEs can change, and the tuning parameters need to be determined by cross-validation. However, learning can be computationally extensive or expensive; and more iterations do not necessarily guarantee minimum testing MSEs for SBML.

The testing MSEs for the models with 2 unmeasured predictors (X_4, X_5) were larger than for the models with all 5 important predictors, which was expected. When we selected appropriate predictors, SBML made more improvements than FLM or OLM. In the presence of noise variables, SBML (without learning, $S^0 = p$ -values) provided smaller testing MSEs than OLM and FLM, regardless of the number of noise variables in the models. In addition, the gap in the testing MSEs for SBML and OLM or FLM gets larger as we increase the number of noise variables. Similar trends were observed for the noise effect 1) in the absence of 2 important predictors and 2) even in the presence of mild correlation among attributes. In the absence of 2 important predictors, SBML seemed robust and outperformed FLM and OLM as we increased the number of noise variables. For Setting #1 with mild correlations ($r = 0.5$), testing MSEs for both FLM and OLM increased regardless of the number of noise variables, while testing MSEs for SBML slightly reduced in general (compared to Setting #1 with $r = 0$). Most importantly, testing MSEs for SBML are much smaller compared to FLM and OLM under all scenarios.

SBML yielded the smallest testing MSEs compared to OLM and FLM among different correlations we tried. As we increased correlations, SBML further reduced testing MSEs; in contrast, FLM and OLM increased testing MSEs. When we had 2 important predictors excluded from the model: 1) the testing MSEs in general were

higher than when we included all 5 predictors, regardless of correlation across different methods; and 2) SBML yielded the smallest testing MSEs compared to OLM and FLM, when there are mild/strong correlations. Additionally, when compared against other ML methods, SBML seems to perform better when there is more nonlinearity in the data, especially when the training size is small, and predicts relatively well compared to SVM with RBF kernel, given manual cross-validation method.

The performance of SBML seems satisfactory and promising compared to existing approaches, especially when we use a constant value of 0.5 as the initial similarity scores with about 10 iterations. The conclusions are drawn under our simulation settings, and different data structures or different values of the tuning parameters can lead to different conclusions. Nonetheless, we expect the prediction of SBML to be further improved by optimizing the tuning parameters via cross-validation in the training stage, given computationally efficient programming power.

4. RECURSIVE SIMILARITY-PRINCIPLE-BASED MACHINE LEARNING

4.1 Introduction

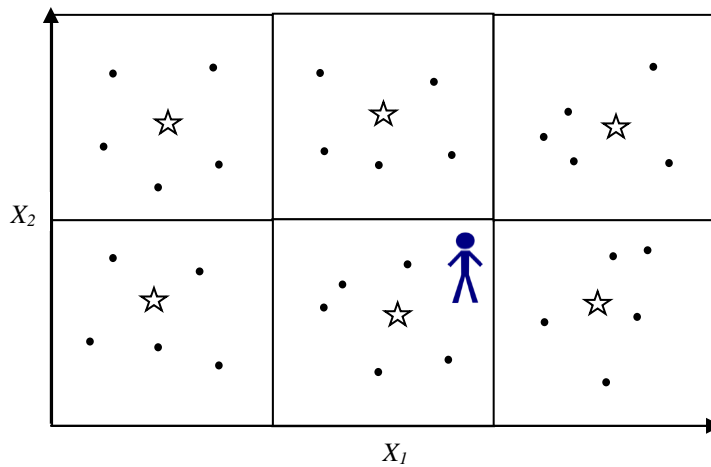
So far, we saw SBML performing well even when the data is not large. However, for big data, the computing time can be extensive. In order to apply SBML to big data, we propose the hierarchical or recursive SBML (RSBML) to allow computational efficiency and the utilization of diverse data sources to achieve a better prediction. Recursive learning is an efficient way to learn from complicated data when the differences are often difficult to precisely define. For instance, two trials conducted at different times or in different countries may differ in medical practice or standard of care, or on account of race or other unknown characteristics. In this case, RSBML can be viewed as a meta-analysis. RSBML can also be viewed as a combination of SBML and dimensionality reduction methods, and it can continue on more than two levels.

As illustrated in Figure 4.1, recursive (hierarchical) learning is a much more efficient way of learning in the real world than learning everything from one-level SBML. Suppose the dots represent individual subjects, and each star represents the center or representative (e.g., mean, median) of each group. If a new subject comes in (person figure, treated as a Test set), it may be computationally efficient when we only compare this new subject with the 6 group representatives instead all 30 subjects.

The idea of RSBML is based on human learning in an effort to capture efficiency and save time. The human brain has limited capacity and can sometimes memorize only the forest and not the individual trees for efficiency. The recursive / constructive learning process also has been tested in cognitive psychology for memory training. According to

Pardilla-Delgado, E & Payne, J D (2017), “Remembering the gist of experience (instead of or along with individual details) is arguably an adaptive process and this task has provided a great deal of knowledge about the constructive, adaptive nature of memory.” In a similar manner, RSBML resembles the nature of human learning by distilling from the group averages.

Figure 4.1 RSBML Illustration



4.2 Methodology

Recursive SBML involves hierarchical learning at multiple levels, the individual and group levels. Let's illustrate RSBML using clinical trials as an example. First, SBML is applied to the individual subjects within each group or trial to obtain the attribute-scaling factors and weights as described in earlier chapters. Then SBML is applied again to the trial level, in which similarities between different trials are considered. To determine the similarities between the trials, aggregated attributes need to be used, such as mean outcome, mean age, and the mode or proportion of female participants in each clinical trial. New variables (often necessary) can be added to further characterize or

distinguish the different trials. Finally, the weights of SBML at different levels are combined to predict the individual subject's results. Specifically, in the aggregated level, the predicted outcome for the subject t is modeled based on SBML:

$$\hat{Y}_t = \sum_{l=1}^L W^*_{tl} \hat{Y}_l(\bar{X}_l), \quad t = 1, 2, \dots, L \quad (22)$$

where W^*_{tl} is determined by the similarity score S^*_{tl} between subject t and virtual subject l (i.e., a group representative, who has the group attributes \bar{X}_l such as group mean age, group mean weight, etc.). Here, $(\bar{X}_l, \hat{Y}_l(\bar{X}_l))$ is the training set for determining the attribute-scaling factors R^* in weights W^*_{tl} .

To predict the outcome of the group representative using only the subjects in the group l :

$$\hat{Y}_l(\bar{X}_l) = \sum_{j=1}^{N_l} W_{lj} O_j, \quad l = 1, 2, \dots, L \quad (23)$$

That is,

$$\hat{Y}_t = \sum_{l=1}^L W^*_{tl} \hat{Y}_l(\bar{X}_l) = \sum_{l=1}^L W^*_{tl} \sum_{j=1}^{N_l} W_{lj} O_j, \quad t = 1, 2, \dots, L \quad (24)$$

where for a given group l , weight W_{lj} is determined in the same way as W_{ij} for the paired individuals in a single group or trial, i.e., by using the similarity score between the i^{th} and j^{th} subjects in the l^{th} group. N_l is the number of subjects in the group l . The computation time for SBML is $O(N^2)$, while it is $O(N^{3/2})$ for RSBML.

4.3 Algorithms

We can use the following algorithm to run simulations to evaluate the performance of RSBML as well as to optimize the tuning parameters for RSBML.

Stage 1: SBML to make up a Training set in Stage 2 for RSBML

For simplicity and the purpose of illustration, let's assume we have 10 groups and that each group has 20 subjects as a training set. For simplicity, we will use the same number of subjects in each group and assume there is no missing data.

For *Group 1*, calculate the means of the baseline characteristics of subjects; this will serve as a representative for Group 1 (*Test set for Group 1 in Stage 1*). Using the 20 subjects as a *Training set for Group 1 in Stage 1*, and the group representative as a *Test set for Group 1 in Stage 1* (with group means of baseline characteristics as predictors), we can predict the group representative's outcome using methods such as the simple SBML or GLM. This predicted outcome and the group means will serve as the 1st subject in the *Training set in Stage 2* in the RSBML setting later. We repeat this process of *Stage 1 Simple SBML* for the rest of the groups. By doing so, we are preparing for the *Training set in Stage 2* for the RSBML.

Consequently, at the end of *Stage 1*, we will have 10 records (1 record representing each group) which will serve as a *Training set* of *Stage 2*. To reiterate, within each group, the group representative record has been served as 1) the only single subject in the *Test set for Group 1 in Stage 1*; and once its outcome has been predicted, then as 2) one of the 10 representative records in the *Training set* of *Stage 2* (treating the predicted outcome in *Stage 1* as the observed outcome in *Stage 2*).

Stage 2: RSBML

In the previous *Stage 1*, we made up a 'virtual' *Training set* (based on representatives from each group) to be used in *Stage 2* (Recursive part of SBML). Now, let's assume we

have 5 new subjects that we want to predict future outcomes; these 5 new subjects will serve as a *Test set* of *Stage 2* and *Stage 3*. That is, instead of using the individual-level data obtained in *Stage 1*, we will use the group-level data (as a *Training set* of *Stage 2*) to predict the outcomes of these 5 new subjects in the *Test set* of *Stage 2*. This way, we can save computation time.

Stage 3: SBML to compare prediction accuracy

To evaluate whether RSBML saves computation time without losing prediction accuracy, we can combine the individual data of 200 subjects from *Stage 1*, and use them as a *Training set* for simple SBML, then run Simple SBML to predict the outcomes of the same 5 new subjects from *Test set* of *Stage 2*. We then can compare the testing errors from this *Stage 3* (Simple SBML as described in Chapter 1 for a continuous outcome and Chapter 2 for a binary outcome) with testing errors obtained in *Stage 2* (RSBML).

4.4 Simulations

We explore the effects of nonlinearity, noise variables, unmeasured predictors and the maximum number of iterations for learning for RSBML (*epoch*). We also evaluate the performance of RSBML and compare with the generalized linear model (GLM) for continuous outcomes. We only compare to GLM in this section because 1) we already compared to other classical and popular ML methods in earlier chapters and 2) the main purpose of this chapter is to evaluate the performance of RSBML against SBML in terms of prediction accuracy and computation time.

4.4.1 Simulation Data and Setting

We generate standard multivariate normal (MVN) data with 7 independent attributes, X_1 through X_7 , then we later generate additional noise variables if needed. The true outcome functions (Y) discussed in this chapter are listed below in Table 4.1.

Table 4.1 List of True Outcome Functions

True outcome functions
$Y_1 = X_1 + X_2 + X_3$
$Y_2 = X_1 + X_2X_3 + X_3^2$
$Y_3 = X_1 + X_2X_3 + 3X_3^2 + X_4$
$Y_4 = X_1 + X_2X_3 + 5X_3^2 + X_4$
$Y_5 = X_1 + X_2X_3 + 2X_3^2 + \sin(X_4)$
$Y_6 = X_1 + X_2X_3 + 2X_3^2 + 2\sin(X_4)$
$Y_7 = X_1 + X_2X_3 + 2X_3^2 + X_7$
$Y_8 = X_1 + X_2X_3 + 2X_3^2 + X_7^2$
$Y_9 = X_1 + X_2X_3 + 2X_3^2 + X_4$
$Y_{10} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5$
$Y_{11} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5 + X_6$
$Y_{12} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5 + X_6 + X_7$
$Y_{13} = X_1 + X_2X_3 + 2X_3^2 + X_4^2$

Regardless of the true outcome function, the performance will be evaluated across two different models: $f_1(x)$ with selected variables (X_1, X_2, X_3, X_7) and $f_2(x)$ with all 7 variables ($X_1, X_2, X_3, X_4, X_5, X_6, X_7$). For instance, regardless of the true outcome function, the GLM for $f_1(x)$ with selected variables is as follows:

$$f_1(x) = \beta'_0 + \beta'_1X_1 + \beta'_2X_2 + \beta'_3X_3 + \beta'_7X_7.$$

Similarly, the GLM for $f_2(x)$ with all 7 variables is as follows:

$$f_2(x) = \beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_3 + \beta_4X_4 + \beta_5X_5 + \beta_6X_6 + \beta_7X_7.$$

For Y_1 and Y_2 :

- $f_1(x)$ indicates that the model captures all 3 important predictors (X_1, X_2, X_3) and includes 1 noise variable (X_7).
- $f_2(x)$ indicates that the model captures all 3 important predictors (X_1, X_2, X_3) and includes 4 noise variables (X_4, X_5, X_6, X_7).

For Y_3 through Y_6, Y_9 and Y_{13} :

- $f_1(x)$ indicates that the model captures only 3 important predictors (X_1, X_2, X_3), i.e., missing 1 important predictor (X_4); and includes 1 noise variable (X_7).
- $f_2(x)$ indicates that the model captures all 4 important predictors (X_1, X_2, X_3, X_4) and includes 3 noise variables (X_5, X_6, X_7).

For Y_7 and Y_8 :

- $f_1(x)$ indicates that the model captures all 4 important predictors (X_1, X_2, X_3, X_7) and includes 0 noise variable.
- $f_2(x)$ indicates that the model captures all 4 important predictors (X_1, X_2, X_3, X_7) and includes 3 noise variables (X_4, X_5, X_6).

For $Y_{10} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5$:

- $f_1(x)$ indicates that the model captures only 3 out of 5 important predictors (i.e., X_1, X_2, X_3), missing 2 important predictors (X_4, X_5) and includes 1 noise variable (X_7).
- $f_2(x)$ indicates that the model captures all 5 important predictors (X_1, X_2, X_3, X_4, X_5) and includes 2 noise variables (X_6, X_7).

For $Y_{I1} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5 + X_6$:

- $f_1(x)$ indicates that the model captures only 3 out of 6 important predictors (i.e., X_1, X_2, X_3), missing 3 important predictors (X_4, X_5, X_6) and includes 1 noise variable (X_7).
- $f_2(x)$ indicates that the model captures all 6 important predictors ($X_1, X_2, X_3, X_4, X_5, X_6$) and includes 1 noise variable (X_7).

For $Y_{I2} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5 + X_6 + X_7$:

- $f_1(x)$ indicates that the model captures all 4 out of 7 important predictors (i.e., X_1, X_2, X_3, X_7), missing 3 important predictors (X_4, X_5, X_6) and includes 0 noise variable.
- $f_2(x)$ indicates that the model captures all 7 important predictors ($X_1, X_2, X_3, X_4, X_5, X_6, X_7$) and hence includes 0 noise variable.

In this chapter, we assume no correlations among attributes ($r = 0$) and there is no missing data. The training size per group in *Stage 1* is 20 for each of the 64 groups, and the testing size in *Stage 2* is 1000. SBLM or RSBML algorithms use the following values for the tuning parameters, unless otherwise specified: learning rate (α) of 0.125, the maximum number of iterations (*epoch*) of 10, penalty coefficient (λ) of 1, and initial similarity scores $S_k^0 = 0.5$ for $k = 1, \dots, K$, where K is the number of predictors included in the model for analysis, i.e., $S^0 = (0.5, \dots, 0.5)$.

4.4.2 Effect of Learning

In this section, we evaluate the impact of learning (*epoch*) of RSBML on the testing errors, given three true outcome functions: Y_3 , Y_6 and Y_{11} . The three scenarios are considered to illustrate the effect of learning in the presence of unmeasured predictors and hopefully to reflect the real-world cases. The results are shown in Table 4.2 and Figure 4.2. When we consider Y_3 and Y_6 , the model $f_1(x)$ indicates that there are one unmeasured predictor and one noise variable, and the model $f_2(x)$ means all of the important predictors are included but three noise variables are included to the detriment of the model. Given Y_{11} , the model $f_1(x)$ fails to capture 2 additional unmeasured predictors.

First, given Y_3 , the testing errors for RSBML without learning (i.e., $epoch = 0$) are 0.9875 for $f_1(x)$ and 1.0098 for $f_2(x)$, which are very close to the testing errors for GLM (1.02 and 0.96 for $f_1(x)$ and $f_2(x)$, respectively). As we increase the maximum number of iterations (*epoch*) to 5, testing errors for RSBML dramatically reduce from 0.9875 to 0.6374 for $f_1(x)$ and from 1.0098 to 0.5585 for $f_2(x)$. Furthermore, RSBML appears very robust and stable throughout iterations of learning (*epoch* from 5 to 20), maintaining the testing errors of 0.64 and 0.56 for $f_1(x)$ and $f_2(x)$, respectively.

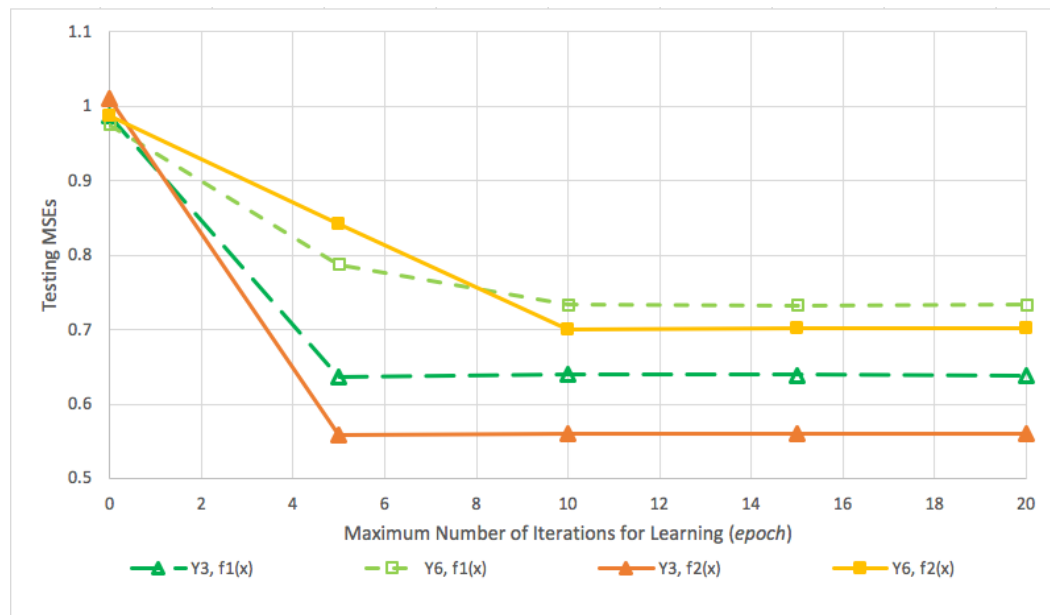
Similar patterns were observed for Y_6 and Y_{11} . As we increase the *epoch* to 5, the testing errors of RSBML reduce roughly from 0.97 to 0.78 for $f_1(x)$ and from 0.98 to 0.8 for $f_2(x)$. RSBML becomes very robust and stable as we further increase the maximum number of iterations (*epoch* from 10 to 20), yielding smaller testing errors around 0.7 for both $f_1(x)$ and $f_2(x)$.

In summary, 1) learning helps reduce the testing errors of RSBML, regardless of the true outcome functions and even in the presence of one or three noise variables and/or unmeasured predictors; and 2) about 10 or 15 iterations of learning seem to produce robust and reliable results. The maximum number of iterations (*epoch*) can be determined by the cross validation.

Table 4.2 Effect of Learning on Testing MSEs for RSBML

<i>epoch</i>	Y_3		Y_6		Y_{11}	
	$f_1(x)$	$f_2(x)$	$f_1(x)$	$f_2(x)$	$f_1(x)$	$f_2(x)$
0	0.9875	1.0098	0.9750	0.9864	0.9774	0.9830
5	0.6374	0.5585	0.7877	0.8415	0.7827	0.8331
10	0.6394	0.5597	0.7331	0.7012	0.7364	0.7246
15	0.6392	0.5602	0.7330	0.7015	0.7366	0.7251
20	0.6391	0.5602	0.7331	0.7018	0.7366	0.7252

Figure 4.2 Effect of Learning on MSEs for RSBML



4.4.3 Effect of Nonlinearity

To understand the impact of true outcome functions (nonlinearity) on the testing errors for RSBML, we compare several true outcome functions (Y) across two models: $f_1(x)$ and $f_2(x)$. The results are summarized in Table 4.3.

Table 4.3 Effect of True Outcome Functions on MSEs

True outcome functions	$f_1(x)$		$f_2(x)$	
	RSBML	GLM	RSBML	GLM
$Y_4 = X_1 + X_2X_3 + 5X_3^2 + X_4$	0.4685	1.0489	0.5463	1.0207
$Y_6 = X_1 + X_2X_3 + 2X_3^2 + 2\sin(X_4)$	0.7331	0.9691	0.7012	0.8312
$Y_8 = X_1 + X_2X_3 + 2X_3^2 + X_7^2$	0.7157	0.9743	0.7019	0.9774
$Y_{11} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5 + X_6$	0.7364	0.9685	0.7246	0.7402
$Y_{13} = X_1 + X_2X_3 + 2X_3^2 + X_4^2$	0.7355	0.9546	0.7299	0.9634

When we use the model $f_1(x)$, the testing errors range from 0.4685 to 0.7364 for RSBML and from 0.9546 to 1.0489 for GLM across various true outcome functions.

When we use the model $f_2(x)$, the testing errors range from 0.5463 to 0.7299 for RSBML and from 0.7402 to 1.0207 for GLM across various true outcome functions. Regardless of the outcome functions (except Y_4) and the models, RSBML seems to achieve robust and consistent testing MSEs around 0.7, which are much smaller compared to GLM.

Importantly, when we have 3 unmeasured predictors such as in Y_{11} with the model $f_1(x)$ instead of $f_2(x)$, RSBML yields consistent performance and lose subtle precision (i.e., MSE slightly increases from 0.7246 to 0.7364), but not as much as GLM (i.e., MSE increases by 0.2 from 0.7402 to 0.9685).

To summarize, RSBML produces stable, robust and much smaller testing errors than GLM, regardless of the true outcome functions (i.e., linear or nonlinear), even in the presence of noise variables or unmeasured predictors.

4.4.4 Effect of Noise Variables

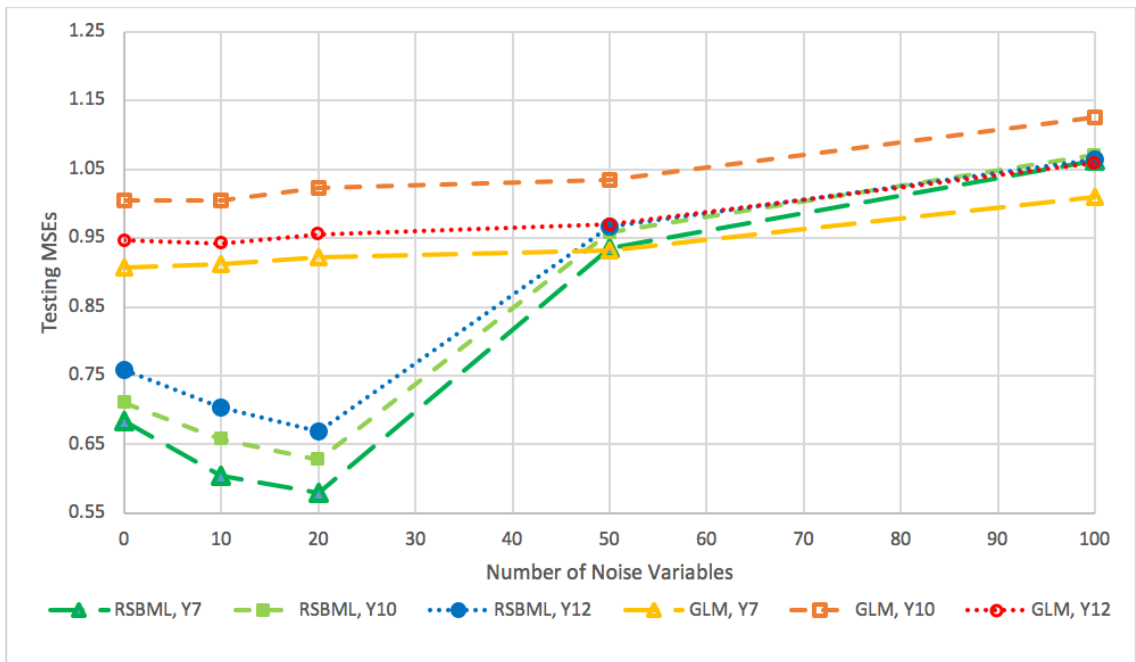
In this section, in order to evaluate the impact of number of noise variables (0, 10, 20, 50, 100), we generate more random numbers from MVN to make up a dataset with additional noise variables (up to 100). For RSBML, *epoch* of 10 or 20 has been used. Table 4.4 and Figure 4.3 shows the prediction accuracy for three outcome functions (Y_7 , Y_{10} , Y_{12}) across two models, $f_1(x)$ and $f_2(x)$, given different number of noise variables. The 1st row of the table represents the base model $f_1(x)$, and the 2nd row represents a model which includes 10 noise variables in addition to the base model $f_1(x)$ for the analysis, and so on.

Given Y_7 and when there is no noise variable included in the model, RSBML yields a testing error of 0.68, whereas GLM yields around 0.9. As we increase the number of noise variables from 0 to 20, the testing error decreases to 0.58 for RSBML, whereas in contrast, it slightly increases to 0.92 for GLM. With 50 or 100 noise variables included in the model, the testing error of RSBML dramatically increase to 0.9 or 1.06. Similar pattern was observed for Y_{10} (which has two additional predictors compared to Y_7) and Y_{12} (which has three additional predictors compared Y_7).

To summarize, when the number of noise variables is smaller or close to the training size (20 per group), RSBML seems to perform well. That is, RSBML seems to outperform GLM, yielding about 0.2-0.3 smaller testing errors, when there are 0, 10 or 20 noise variables included in the model. With 50 or 100 noise variables (i.e., number of noise variables \gg training size), the testing errors for RSBML dramatically increases and get closer to those of GLM. This may be due to *overparameterization* for both RSBML and GLM.

Table 4.4 Effect of Number of Noise Variables on MSEs with RSBML

# Noise Variables	Y_7		Y_{10}		Y_{12}	
	RSBML	GLM	RSBML	GLM	RSBML	GLM
0	0.6835	0.9070	0.7111	1.0040	0.7586	0.9461
10	0.6049	0.9128	0.6577	1.0043	0.7032	0.9418
20	0.5796	0.9214	0.6284	1.0220	0.6691	0.9551
50	0.9344	0.9312	0.9582	1.0336	0.9662	0.9698
100	1.0626	1.0098	1.0705	1.1256	1.0644	1.0589

Figure 4.3 Effect of Number of Noise Variables on MSEs for RSBML

4.4.5 Effect of Unmeasured Predictors

Often times, we fail to capture all of the important variables in the analysis or model specification, but the variables were used in the true outcome function to generate the response Y , and we call these *unmeasured predictors*. In this section, we consider three outcome functions (Y_9, Y_{10}, Y_{11}), given the model $f_l(x)$, and we focus on the effect of unmeasured predictors on the performance of RSBML (Table 4.5).

Table 4.5 Effect of Unmeasured Predictors on MSEs with RSBML

<i>True Outcome Functions</i>	<i># of Unmeasured Predictor(s)</i>	RSBML	GLM
$Y_9 = X_1 + X_2X_3 + 2X_3^2 + X_4$	1	0.7158	0.9726
$Y_{10} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5$	2	0.7221	0.9723
$Y_{11} = X_1 + X_2X_3 + 2X_3^2 + X_4 + X_5 + X_6$	3	0.7364	0.9685

As there are more unmeasured predictors, the testing errors for RSBML slightly increase. When we have only 1 unmeasured predictor (X_4) as in Y_9 using $f_1(x)$, the testing error of RSBML is 0.7158. When there are 2 unmeasured predictors (X_4, X_5) as in Y_{10} using $f_1(x)$, the testing error of RSBML slightly increases to 0.7221. Similarly, when there are 3 unmeasured predictors (X_4, X_5, X_6) as in Y_{11} using $f_1(x)$, the testing error of RSBML slightly increases to 0.7364. However, the difference is very small. To summarize, even we fail to capture important predictors in the analysis, RSBML yields robust, stable and smaller testing errors (around 0.7) compared to GLM (around 0.97), and the impact of unmeasured predictors on prediction accuracy for RSBML is very minimal.

4.4.6 Comparison of RSBML vs. SBML

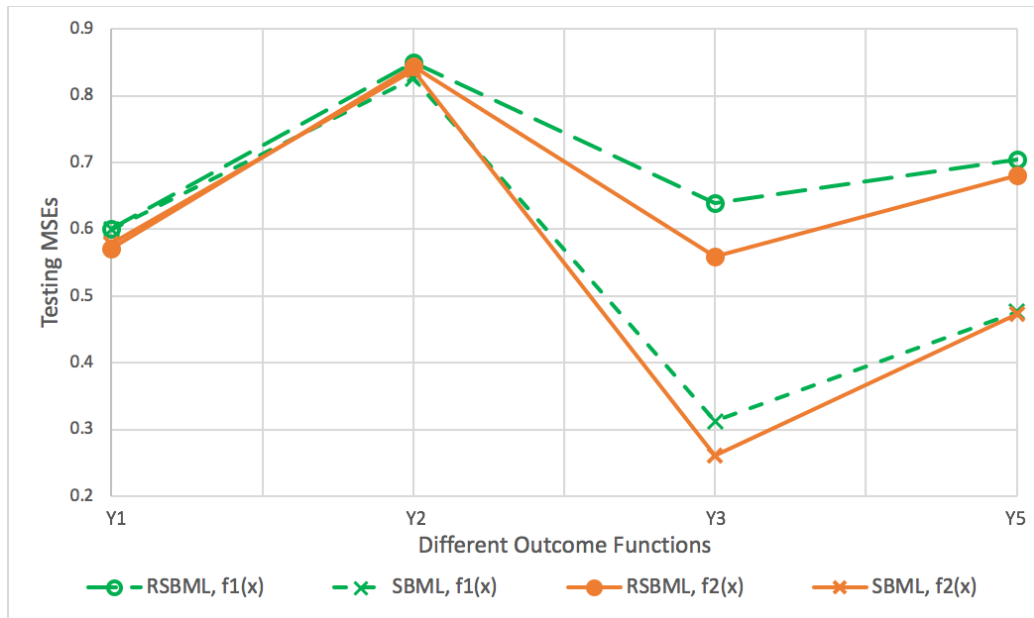
In this section, we compare RSBML against SBML using big data (i.e., combined individual-level data for all groups) in terms of prediction accuracy as well as computation time. We select a few cases in this work, and the performance of both RSBML and SBML with two models, $f_1(x)$ and $f_2(x)$, are summarized in Table 4.6 and Figure 4.4. For some cases, RSBML kept reducing testing errors as we increase *epoch*. However, for simplicity, we only present the results where we use 10 as the maximum

number of iterations for learning – because for the most cases, both RSBML and SBML seem to provide stable results by *epoch* of 10.

Table 4.6 Comparison of RSBML vs. SBML

<i>Outcome Functions</i>	$f_1(x)$		$f_2(x)$		<i>Time Spent (min)</i>	
	RSBML	SBML	RSBML	SBML	RSBML	SBML
Y_1	0.6013	0.5998	0.5708	0.5785	2.36	12.39
Y_2	0.8496	0.8258	0.8443	0.8386	2.44	13.00
Y_3	0.6394	0.3127	0.5597	0.2608	1.84	7.01
Y_5	0.7039	0.4763	0.6807	0.4735	2.15	10.62

Figure 4.4 Comparison of RSBML vs. SBML



First, we consider a linear outcome function Y_1 . Regardless of the model, i.e., $f_1(x)$ or $f_2(x)$, RSBML achieves similar performance as SBML (testing errors around 0.6) at about 6 times faster (i.e., about 2 minutes for RSBML vs. about 12 minutes for SBML). Next, we consider Y_2 . Similar to Y_1 , RSBML provides similar testing errors (around 0.8)

as SBML again at 6 times faster, regardless of the model, i.e., $f_1(x)$ or $f_2(x)$; RSBML takes about 2 minutes, while SBML takes about 13 minutes.

For Y_3 , where we have an additional predictor (X_4), the results are no longer satisfactory. Unlike previous two cases, RSBML cannot achieve similar performance as SBML, and the testing errors of RSBML are almost double of SBML. As we increase *epoch* from 0 to 5, the testing errors of RSBML dramatically reduce from 0.9875 to 0.6374 for $f_1(x)$ and from 1.0098 to 0.5585 for $f_2(x)$; however, as we increase *epoch* to 10 and 15, the testing errors of RSBML slightly increase to 0.6394 then to 0.6392 for $f_1(x)$ and to 0.5597 then to 0.5602 for $f_2(x)$. That is, RSBML seems to reach its local minimum at *epoch* = 5. Similarly, the testing errors of SBML dramatically reduce from 0.9705 to 0.3127 for $f_1(x)$ and from 0.988 to 0.2608 for $f_2(x)$, and SBML seems to reach its local minimum at *epoch* = 5. The testing errors for GLM are 1.0189 for $f_1(x)$ and 0.9591 for $f_2(x)$, which are much larger than RSBML. Hence, with some trade-offs of prediction accuracy, RSBML helps save time (about 4 times faster than SBML) and still achieves better performance than GLM.

Lastly, we consider Y_5 . Unlike Y_1 and Y_2 , and similar to Y_3 , RSBML cannot achieve similar performance as SBML. As we increase *epoch* from 0 to 5, the testing errors of RSBML dramatically reduce from 0.9732 to 0.7623 for $f_1(x)$ and from 1.0004 to 0.8332 for $f_2(x)$. As we increase *epoch* to 10, the testing errors of RSBML further reduce to 0.7039 for $f_1(x)$ and to 0.6807 for $f_2(x)$, then slightly increase to 0.7043 for $f_1(x)$ and to 0.682 for $f_2(x)$. That is, RSBML seems to reach its local minimum at *epoch* = 10. Similarly, the testing errors of SBML dramatically reduce from 0.9568 to 0.4763 for $f_1(x)$

and from 0.9783 to 0.4735 for $f_2(x)$, and SBML seems to reach its local minimum at $epoch = 5$. The testing errors for GLM are 0.9615 for $f_1(x)$ and 0.9177 for $f_2(x)$, which are much larger than RSBML. Hence, with some trade-offs of prediction accuracy, RSBML helps save time (about 5 times faster than SBML) and still achieves better performance than GLM.

In summary, based on the four cases, satisfactory performance of RSBML was reached at much faster computing time, and the performance of both RSBML and SBML stabilized at $epoch$ around 5 or 10. For the first two cases we considered, the computing time of RSBML was about 6 times faster than SBML (about 2 minutes for RSBML vs. about 12-13 minutes for SBML). For the other two cases we considered, where there was an unmeasured predictor (X_4), RSBML performed poorly compared to SBML; however, with some trade-offs of prediction accuracy, RSBML helps save time (about 4-5 times faster than SBML) and still provided smaller testing errors than GLM.

4.5 Summary

In this chapter, we extended SBML algorithm to hierarchical / recursive / meta-analysis frameworks. As in SBML, learning helped reduce the testing errors of RSBML, regardless of the true outcome functions and even in the presence of a few number of noise variables and/or unmeasured predictors. About 10 or 15 iterations of learning seemed to produce robust and reliable results, and the maximum number of iterations ($epoch$) can be determined by the cross-validation.

Regardless of the true outcome functions (i.e., linear or nonlinear), RSBML produced stable, robust and much smaller testing errors than GLM, even in the presence

of a few noise variables and/or unmeasured predictors; RSBML worked well especially when the data is nonlinear.

As we evaluated the effect of noise variables (i.e., high dimensionality), RSBML outperformed GLM and yielded about 0.2-0.3 smaller testing errors, when the number of noise variables was smaller or close to the training size per group in *Stage 1* (N_i). With 50 or 100 noise variables (i.e., number of noise variables \gg training size per group in *Stage 1*), the testing errors for RSBML got larger and closer to GLM testing errors. This may be due to *overparameterization* for both RSBML and GLM.

As the number of unmeasured predictors in the analysis increased, the testing errors of RSBML slightly increased; however, the difference was very small which may be due to random simulation errors. That is, even when we fail to capture important predictors in the analysis, RSBML yielded robust, stable and smaller testing errors (around 0.7) compared to GLM (around 0.97), and the impact of unmeasured predictors on prediction accuracy for RSBML was very minimal.

Most importantly, RSBML was able to achieve satisfactory prediction accuracy as SBML at a much faster running time, and both RSBML and SBML stabilized at *epoch* of 5 or 10. For the half of the cases we considered, the computing time of RSBML was about 6 times faster than SBML. For the other half of the cases we considered, RSBML yielded larger testing MSEs than SBML; however, with some trade-offs of prediction accuracy, RSBML helped save time (about 4-5 times faster than SBML) and still achieved better performance than GLM.

5. CONCLUSION AND DISCUSSION

The main goal in the scientific discovery is often prediction. For drug development, such a prediction can be used in early phase data to predict later phase outcomes and tuning the target population on an aggregated level. Using the outcome from earlier clinical trials, we can also predict the outcomes for future subjects on the individual subject level, also called *precision medicine*.

SBML is based on the *similarity principle*, while other ML methods use *similarity measures* but not necessarily the *similarity principle*. Therefore, SBML is more intuitive and objective. In this work, we 1) proposed a new artificial intelligence (AI) or machine learning (ML) approach, SBML, based on the *similarity principle*, 2) introduced the attribute-scaling factors to reflect the relative importance of each attribute, 3) derived the formulation for partial derivatives for Similarity-Principle-Based Machine Learning (SBML) using the gradient method, and 4) provided the AI or ML algorithms. Through extensive simulations, we evaluated impact of tuning parameters and showed that SBML can achieve a better prediction performance in terms of the testing MSE in most cases studied for both regression and classification problems.

The proposed SBML algorithm has some distinct features as outlined in this chapter. In contrast to the majority of ML methods that require big data, SBML can consistently produce reliable and robust predictions compared to other methods, even with small training data. When data displays complex structures or nonlinearity, SBML shows even better prediction results than other traditional methods and yields similar testing errors to popular ML methods such as support vector machines or random forests.

Unlike most kernel methods, SBML introduces the attribute-scaling factors so that the relative importance of different attributes can be objectively determined in the similarity measures. Therefore, SBML often produces better predictions when we have small training data or when the data is more complex.

In training SBML, we normalize the data for predictors and for continuous outcomes, so the tuning parameters to be determined can be applied to different datasets, regardless of different measurement units. Based on our simulation study, we suggest the following tuning parameters for normalized data: a constant value of 0.5 as the initial similarity scores for all predictors, a learning rate (α) around 0.125, a penalty coefficient (λ) ranging from -2 to 2, and the maximum number of iterations (*epoch*) ranging from 5 to 20. The tuning parameters can be determined via cross-validation.

When the outcome is binary, we can evaluate SBML using several criteria, including the predicted probability of the event (mathematically equivalent to MSE) and misclassification errors (equivalent to the sum of both false positives and false negatives). The conclusions are similar to those for the continuous outcomes.

It is desirable to evaluate SBML using more clinical trial data and other real data as our conclusions are mainly based on the simulation studies. Moreover, it is also desirable to make the tuning parameters automatically determined in the future and improve computationally efficacy using a better learning algorithm. Cross-validation can be also used for method selection between different ML approaches. We believe SBML has a great potential, not only for clinical trials but also in many other fields. Many improvements can be made for this method.

APPENDIX

Application and Demo using Pima Indians Diabetes Dataset

The details of the proposed algorithm in Section 2.2.4 are described and implemented step by step. The proposed algorithm is conducted as an explanation using a partial data of the *PimaIndiansDiabetes* dataset available *mlbench* package in R, which is originally taken from the National Institute of Diabetes and Digestive and Kidney Diseases.

This *Pima Indians Diabetes* dataset consists of all females who are 21 years old and above of the Pima Indian heritage, and it contains 768 observations and 9 variables: 1) Number of pregnancies; 2) Plasma glucose concentration in an oral glucose tolerance test; 3) Diastolic blood pressure (mm Hg); 4) Triceps skin fold thickness (mm); 5) 2-Hour serum insulin (μ U/ml); 6) Body mass index (weight in kg/(height in m)²); 7) Diabetes pedigree function; 8) Age in years; 9) Binary outcome variable (Yes or No for diabetic, diagnosed according to the World Health Organization criteria). The objective of the collected dataset is to predict whether or not a person would develop diabetes within five years given the eight significant risk factors included in the dataset (Smith 1988).

However, for simplicity and visualization purposes, we will only use two independent variables (*pressure* and *age*) as our predictors or attributes and *glucose* as our continuous outcome dependent variable. Furthermore, to simplify the calculation, we will only use the first 5 subjects as a training set and next 3 subjects as a Test set, and assume $\eta = 1$ and $\rho = 2$.

Please note that if the focus is about predicting actual values of the outcome instead of evaluating model or parameters, we can use the raw (non-normalized) outcome values.

```
> head(myTrainPID)
  pressure age glucose
1      72  50     148
2      66  31      85
3      64  32     183
4      66  21      89
5      40  33     137

> head(myTestPID)
  pressure age glucose
6      74  30     116
7      50  26      78
8       0  29     115
```

Step 1) Normalize the Training dataset as described in Section 2.2.2.

```
> xTrain
  pressure      age
1 0.8358876 1.58780808
2 0.3536447 -0.22956261
3 0.1928971 -0.13391152
4 0.3536447 -1.18607350
5 -1.7360742 -0.03826044
```

Step 2) Assign Initial Similarity Scores (S_k^0):

Given (0.5, 0.5) as the initial similarity scores (i.e., $S_k^0 = 0.5$),

```
> initS
[1] 0.5 0.5
```

```
> eta = 1
```

Step 3) Determination / Estimation of Initial Scaling Factors (R_k^0) – Solve for R_k^0 based

on initial similarity scores, e.g., $S_k^0 = p$ -values:

Hand calculations:

For X_1 (i.e., *pressure* variable, $k=1$, 1st variable):

- IQR for $X_1 = 0.3536447 - 0.1928971 = 0.1607476$

- R_1^0 (i.e., attribute-scaling factor for X_1) = $-\log(0.5) / 0.1607476 = 4.312022$

For X_2 (i.e., *age* variable, $k=2$, 2nd variable):

- IQR for $X_2 = -0.03826044 - (-0.2295626) = 0.1913022$

- R_2^0 (i.e., attribute-scaling factor for X_2) = $-\log(0.5) / 0.1913022 = 3.62331$

```
> R0s ## using InitialRs() function - see Appendix
[1] 4.312022 3.623310
```

Step 4) Determination of Scaled Distance (d_{ij}):

For $i=1; j=2$:

```
> xTrain
```

```
      pressure      age
1  0.8358876  1.58780808
2  0.3536447 -0.22956261
```

```
> (4.312022 * abs(0.8358876 - 0.3536447))^2 + (3.623310 * abs(1.58780808 - -
0.22956261))^2
```

```
[1] 47.68495
```

```
> (47.68495)^(1/2)
```

```
[1] 6.905429
```

$d_{12} = d_{21} = 6.90543$

...

For $i=2; j=4$:

```
> xTrain
```

```
      pressure      age
2  0.3536447 -0.22956261
4  0.3536447 -1.18607350
```

```
> ( (4.312022 * abs(0.3536447 - 0.3536447))^2 + (3.623310 * abs(-0.22956261 -
-1.18607350))^2 ) ^ (1/2)
```

```
[1] 3.465735
```

$d_{24} = d_{42} = 3.465735$

...

If we continue the calculations, we get

$$\mathbf{d} = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} \\ d_{41} & d_{42} & d_{43} & d_{44} & d_{45} \\ d_{51} & d_{52} & d_{53} & d_{54} & d_{55} \end{bmatrix}_{5 \times 5}$$

$$= \begin{bmatrix} 0 & 6.90543 & 6.826708 & 10.26349 & 12.55821 \\ 6.90543 & 0 & 0.7749621 & 3.465736 & 9.037534 \\ 6.826708 & 0.7749621 & 0 & 3.874811 & 8.324983 \\ 10.26349 & 3.465736 & 3.874811 & 0 & 9.924357 \\ 12.55821 & 9.037534 & 8.324983 & 9.924357 & 0 \end{bmatrix}_{5 \times 5}$$

We can verify using *Distance()* function – see Appendix.

Step 5) Similarity Functions / Measures (S_{ij}):

By exponentiating the negative values of the scaled distances, we get the following similarity scores. This calculation can be done in R or other software, and the output is the following:

$$S = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} & S_{15} \\ S_{21} & S_{22} & S_{23} & S_{24} & S_{25} \\ S_{31} & S_{32} & S_{33} & S_{34} & S_{35} \\ S_{41} & S_{42} & S_{43} & S_{44} & S_{45} \\ S_{51} & S_{52} & S_{53} & S_{54} & S_{55} \end{bmatrix}_{5 \times 5}$$

$$= \begin{bmatrix} 1 & 0.00100 & 0.00108 & 0.00003 & 3.5e-06 \\ 0.00100 & 1 & 0.46072 & 0.03125 & 0.00011 \\ 0.00108 & 0.46072 & 1 & 0.02075 & 0.00024 \\ 0.00003 & 0.03125 & 0.02075 & 1 & 0.00004 \\ 3.5e-06 & 0.00011 & 0.00024 & 0.00004 & 1 \end{bmatrix}_{5 \times 5}$$

Step 6) Determination of Weight (W_{ij}):

$$W = \begin{bmatrix} 0.99787 & 0.00100 & 0.00108 & 0.00003 & 3.5e-06 \\ 0.00067 & 0.66975 & 0.30856 & 0.02092 & 0.00007 \\ 0.00073 & 0.31070 & 0.67439 & 0.01399 & 0.00016 \\ 0.00003 & 0.02970 & 0.01973 & 0.95048 & 0.00004 \\ 3.5e-06 & 0.00011 & 0.00024 & 0.00004 & 0.99958 \end{bmatrix}_{5 \times 5}$$

Step 7) Prediction of Outcome (\hat{Y}_i):

```
> PredictedY(W, yTrain, yTrain)
$pred_Y
      [,1]
[1,]  0.4721061
[2,] -0.3142942
[3,]  0.5499841
```

```
[4,] -0.9083791
[5,]  0.2075009
```

```
$MSE
[1] 0.225332
```

```
> mySBMLtrain = SBMLtrain(Epoch=EpochN, S0s=initS, Lamda=lamdaVal,
LearningRate=LRval, eta=1, xTrain, yTrain)
```

```
> mySBMLtrain ## <- normalized Xs, raw Ys
$Rs
[1] 4.312022 3.623310
```

```
$Y
      [,1]
[1,] 147.97277
[2,] 115.36986
[3,] 151.20147
[4,]  90.74004
[5,] 137.00266
```

```
$MSE
[1] 387.3007
```

```
> mySBMLtrain ## <- Raw Xs and Ys (i.e., NOT normalized data)
$Rs
[1] 0.3465736 0.3465736
```

```
$Y
      [,1]
[1,] 147.97277
[2,] 115.36986
[3,] 151.20147
[4,]  90.74004
[5,] 137.00266
```

```
$MSE
[1] 387.3007
```

Step 8) Calculation of Errors:

```
> cbind(PredictedY(W, Ytrain, Ytrain)$pred_Y, yTrain)
      yTrain
[1,]  0.4721061  0.4727629
[2,] -0.3142942 -1.0468322
[3,]  0.5499841  1.3169824
[4,] -0.9083791 -0.9503499
[5,]  0.2075009  0.2074368
```

```
> sum((PredictedY(W, Ytrain, Ytrain)$pred_Y - yTrain)^2) / 5
[1] 0.225332
```

```

> cbind(PredictedY(W, Ytrain, Ytrain)$pred_Y, yTrain) #<-normalized Xs, raw Ys
      yTrain
[1,] 147.97277 148
[2,] 115.36986 85
[3,] 151.20147 183
[4,] 90.74004 89
[5,] 137.00266 137

> sum((PredictedY(W, Ytrain, Ytrain)$pred_Y - yTrain)^2) / 5
[1] 387.3007

```

Step 9) is only necessary when we run SBML with learning, therefore, we can skip this step for now and come back later.

Step 10) Prediction of Future Outcomes:

```

> S
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 9.433785e-04 6.116589e-02 2.917710e-02 1.540137e-02 7.287987e-06
[2,] 1.257728e-05 2.998533e-03 5.098282e-03 2.998533e-03 1.454556e-02
[3,] 5.144879e-12 1.151994e-10 2.272285e-10 9.846894e-11 8.899630e-07

> W
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 8.841822e-03 0.573277764 0.2734626275 0.1443494795 6.830671e-05
[2,] 4.902757e-04 0.116886018 0.1987364590 0.1168860180 5.670012e-01
[3,] 5.778107e-06 0.000129378 0.0002551957 0.0001105884 9.994991e-01

> predY
$pred_Y
      [,1]
[1,] -0.3729684
[2,] 0.1461385
[3,] 0.2074312

$MSE
[1] 0.7138606

> predY = PredictedY(W=Weight(Similarity(eta=eta, mySBMLtrain$Rs, xTrain,
xTest)), Ytrain=yTrain, Ytest=yTest)
> predY
$pred_Y
      [,1]
[1,] 112.9373
[2,] 134.4587
[3,] 136.9998

$MSE

```

```
[1] 1226.984
```

Predicted (without learning) vs. Observed Test set:

```
> cbind(PredictedY(W, yTrain, yTest)$pred_Y, yTest)
              yTest
[1,] -0.3729684 -0.2990949
[2,]  0.1461385 -1.2156761
[3,]  0.2074312 -0.3232155

> cbind(PredictedY(W, yTrain, yTest)$pred_Y, yTest) #<-normalized Xs, raw Ys
              yTest
[1,] 112.9373   116
[2,] 134.4587   78
[3,] 136.9998  115
```

Now, let's run SBML with 1 iteration of learning, the learning rate $\alpha = 0.125$ and penalty coefficient $\lambda = 1$. Then the attribute-scaling factors \mathbf{R} will be updated.

Step 9) Iterations of Learning:

```
> mySBMLtrain$Rs
[1] 3.239439 2.719577
```

Note: the previous or initial attribute-scaling factors \mathbf{R} were:

```
> R0s ## using InitialRs() function - see Appendix
[1] 4.312022 3.623310
```

Now that the attribute-scaling factors \mathbf{R} are updated from (4.312, 3.623) to (3.239, 2.719), predictions of future outcomes in Step 10 will be different this time.

Step 10) Prediction of Future Outcomes:

Previously, when we applied SBML without learning, we got the following matrices \mathbf{S} for similarity scores and \mathbf{W} for corresponding weights between the 3 subjects in the Test set and the 5 subjects in the Training set.

```
> S
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 9.433785e-04 6.116589e-02 2.917710e-02 1.540137e-02 7.287987e-06
[2,] 1.257728e-05 2.998533e-03 5.098282e-03 2.998533e-03 1.454556e-02
[3,] 5.144879e-12 1.151994e-10 2.272285e-10 9.846894e-11 8.899630e-07
```

```
> W
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 8.841822e-03 0.573277764 0.2734626275 0.1443494795 6.830671e-05
[2,] 4.902757e-04 0.116886018 0.1987364590 0.1168860180 5.670012e-01
[3,] 5.778107e-06 0.000129378 0.0002551957 0.0001105884 9.994991e-01
```

Now, when we applied SBML with 1 iteration of learning, we got the following updated matrices S for similarity scores and W for corresponding weights between the 3 subjects in the Test set and the 5 subjects in the Training set. As you can see, the testing error (MSE) of SBML reduced from 0.7138606 to 0.6674477 with 1 iteration of learning.

```
> S
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 5.360994e-03 1.225656e-01 7.028963e-02 4.355902e-02 1.382043e-04
[2,] 2.090914e-04 1.272509e-02 1.896384e-02 1.272509e-02 4.170110e-02
[3,] 3.310817e-09 3.416601e-08 5.691583e-08 3.037300e-08 2.847475e-05
```

```
> W
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.0221607974 0.506650564 0.290556982 0.180060360 0.0005712965
[2,] 0.0024221644 0.147410450 0.219681561 0.147410450 0.4830753749
[3,] 0.0001157648 0.001194636 0.001990098 0.001062011 0.9956374907
```

```
> predY
$pred_Y
      [,1]
[1,] -0.30824471
[2,]  0.09626395
[3,]  0.20694763
```

```
$MSE
[1] 0.6674477
```

Predicted (with 1 iteration of learning) vs. Observed Test set:

```
> cbind(PredictedY(W, yTrain, yTest)$pred_Y, yTest)
      yTest
[1,] -0.30824471 -0.2990949
[2,]  0.09626395 -1.2156761
[3,]  0.20694763 -0.3232155
```

R Code for Similarity-Principle-Based Machine Learning

```

### Functions #####

# Calculate initial scaling factors R0s
InitialRs = function(Xtrain, eta, S0s) {
  K = length(S0s); R0s = rep(0, K)
  for (k in 1:K) {
    R0s[k] = min((-log(S0s[k])) ^ (1/eta) / max(IQR(Xtrain[,k]), 0.0000001), 12) }
  return (R0s)
}

Distance = function(Rs, x1, x2) { ## we used rho = 2
  K = length(Rs)
  d2 = 0; for (k in 1:K) { d2 = d2 + (Rs[k] * (x2[k]-x1[k]))^2 }
  return (d2^0.5)
}

SimilarityTrain = function(eta, Rs, Xtrain) {
  N1 = nrow(Xtrain); K = length(Rs);
  S = matrix(0, nrow=N1, ncol=N1)
  for (i in 1:N1) { for (j in i:N1) {
    d2 = 0; for (k in 1:K) { d2 = d2 + (Rs[k]* (Xtrain[i,k]-Xtrain[j,k]))^2 }
    S[i,j] = exp(-d2^0.5)
    S[j,i]=S[i,j] } }
  return (S)
}

Weight = function(S) {
  N1= nrow(S); N2=ncol(S);
  W = matrix(0, nrow=N1, ncol=N2)
  round(head(S),4)
  for (i in 1:N1) {
    sum_S_row = sum(S[i, ])
    for (j in 1:N2) { W[i,j] = S[i,j] / max(sum_S_row,0.0000000001) } }
  return (W)
}

# Calculate Similarity Scores between Training and Test subjects
Similarity = function(eta, Rs, Xtrain, Xtest) {
  N1 = nrow(Xtrain); N2=nrow(Xtest); K = length(Rs)
  S = matrix(0, nrow=N2, ncol=N1)
  for (i in 1:N2) { for (j in 1:N1) {
    d2 = 0; for (k in 1:K) { d2 = d2 + (Rs[k]* (Xtest[i,k]-Xtrain[j,k]))^2 }
    S[i,j] = exp(-d2^0.5) } }
  return (S)
}

# Calculate predicted outcome and Error
PredictedY = function (W, Ytrain, Ytest) {
  OutObj = list()
  OutObj$pred_Y = W %*% Ytrain # For binary outcome, pred_y = prob of being 1.
  OutObj$MSE = mean((OutObj$pred_Y - Ytest)^2)
  return (OutObj)
}

```

```

DerivativeE = function(eta, pred_Y, Rs, X, S, O) {
  N = nrow(X); K =length(Rs)
  der_S = matrix(0, nrow=K*N, ncol=N); der_W = matrix(0, nrow=K*N, ncol=N)
  dist = matrix(0, nrow=N, ncol=N); der_E = rep(0, K)

  for (i in 1:N) { for (j in i:N) {
    d2 = 0; for (k in 1:K) { d2 = d2 + (Rs[k]* (X[i,k]-X[j,k]))^2 }
    dist[i,j] =max(d2^0.5, 0.0000001)
    dist[j,i]=dist[i,j]  }}

  for (m in 1:K) { for (i in 1:N) { for (j in i:N) {
    der_d = (Rs[m]/dist[i,j]) * (X[i,m]-X[j,m]) ^2
    der_S[(m-1)*N+i, j] = -1 * S[i,j] * eta * (dist[i,j])^(eta-1) * der_d
    der_S[(m-1)*N+j, i] =der_S[(m-1)*N+i, j]  } } }

  # Weight Derivative
  for (m in 1:K) { for (i in 1:N) {
    sum_der_S = sum(der_S[(m-1)*N+i, ]); sumSi = sum(S[i, ])
    for (j in 1:N) {
      der_W[(m-1)*N+i, j] = der_S[(m-1)*N+i, j] / sumSi - S[i,j] * sum_der_S /
sumSi^2  } } }

  # Derivatives of E
  for (m in 1:K) { for (i in 1:N) {
    err = (pred_Y[i] - O[i])
    for (j in 1:N) { der_E[m] = der_E[m] + 2/N * err * O[j] * der_W[(m-1)*N+i,
j] }  } }
  return (der_E)
}

Learning = function (LearningRate, Lamda, Rs, der_E) {
  K=length(Rs)
  der_lossFun = der_E+2*Lamda*Rs
  Rs = Rs - LearningRate * der_lossFun
  for (m in 1:length(Rs)) { Rs[m] = min(max(0,Rs[m]),25) }
  return (Rs)
}

SBMLtrain = function(Epoch, S0s, Lamda, LearningRate, eta, Xtrain, Ytrain) {
  TrainObj=list(); OutObj0= list(); OutObj= list()
  R0 = InitialRs(Xtrain, eta, S0s);
  S = SimilarityTrain(eta, R0, Xtrain);
  OutObj0 = PredictedY(Weight(S), Ytrain, Ytrain)
  Rs=R0; OutObj =OutObj0 ;
  TrainMSE0 = OutObj0$MSE ;
  preLoss = OutObj0$MSE+Lamda*sum(Rs^2) ;
  iter=0;
  while (iter<Epoch) {
    preRs=Rs
    Rs = Learning(LearningRate, Lamda, Rs, DerivativeE(eta, OutObj0$pred_Y, Rs, Xtrain,
S, Ytrain))
    OutObj = PredictedY (Weight(SimilarityTrain (eta, Rs, Xtrain)), Ytrain, Ytrain)
    iter=iter+1
    Loss = OutObj$MSE+Lamda*sum(Rs^2)
    if (Loss>preLoss) {Rs=preRs; iter=Epoch+1}
    preLoss=Loss
  }
}

```

```

TrainObj$Rs = Rs; TrainObj$Y = OutObj$pred_Y; TrainObj$MSE = OutObj$MSE
return( TrainObj)
}

getY = function(yfunc, xData, yData, isXvector) {
  if (isXvector == FALSE) {
    x1 = xData[,1]; x2 = xData[,2]; x3 = xData[,3]; x4 = xData[,4]; x5 = xData[,5]; x6 =
xData[,6]; x7 = xData[,7]; }
  if (isXvector == TRUE) {
    x1 = xData[1]; x2 = xData[2]; x3 = xData[3]; x4 = xData[4]; x5 = xData[5]; x6 =
xData[6]; x7 = xData[7]; }
  if (yfunc == "yf7") { yData = x1+x2*x3+2*x3^2 +x7 }
  return(yData)
}

##### Preparation of Pima Indians Diabetes Dataset for Demo of SBML #####

library(mlbench)
data(PimaIndiansDiabetes)
ntrain = 5; ntest = 3; lamdaVal = 1; EpochN = 0;
myXvars = c("pressure", "age")
myYvars = c("glucose")
myvars = c(myXvars, myYvars)

### Trainset - Pima Indians Diabetes (PID)
myTrainPID = PimaIndiansDiabetes[1:ntrain, myvars]
xTrain = myTrainPID[, myXvars]
yTrain = myTrainPID[, myYvars]

### Testset - Pima Indians Diabetes (PID)
myTestPID = PimaIndiansDiabetes[(ntrain+1):(ntrain+ntest), myvars]
xTest = myTestPID[, myXvars]
yTest = myTestPID[, myYvars]

```

R Code for Recursive Similarity-Principle-Based Machine Learning

The following R code is example of RSBML used for simulated multivariate data, assuming we have 7 independent variables, 64 groups (dividing a variable into 4 categories, and based on first 3 variables, so $4^3 = 64$), 50 subjects for the *Training set in Stage 1* in each group, 1000 new subjects for the *Test set in Stage 2 and 3*, learning rate $\alpha = 0.125$ and penalty coefficient $\lambda = 1$ for SBML with iterations of learning, and initial similarity scores = (0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5) for 7 variables.

```
### Simple SBML (SSBML-sv) - sv (selected variables)
pt_start_RSBML = proc.time()
mySBMLtrain_sv = SBMLtrain(Epoch=EpochN, S0=initS_sv, Lamda=lamdaVal,
LearningRate=LRval, eta=1, xTrain_sv, yTrain_sl)
predY_ig_sv = PredictedY(W=Weight(Similarity(eta=eta, mySBMLtrain_sv$Rs, xTrain_sv,
xTest_sv)), Ytrain=yTrain_sl, Ytest=yTest_sl)
predYs_sv[ig, ] = predY_ig_sv$pred_Y

### Recursive Testing
xTest2 = rmvnorm(n=ntest, mean=mu, sigma=sigma) ;
yTest2 = getY(yfunc = yfunc, xData = xTest2, yData = yTest2, isXvector = FALSE)
data_test_all = cbind(xTest2, yTest2)

## Group-level Rs -- sv: selective variables
mySBMLtrain2_sv = SBMLtrain(Epoch=EpochN, S0=initS2_sv, Lamda=lamdaVal,
LearningRate=LRval, eta=1, xTrain2[,sLevelVars], yTrain2)
predY2_sv = PredictedY(W=Weight(Similarity(eta=eta, mySBMLtrain2_sv$Rs,
xTrain2[,sLevelVars], xTest2[,sLevelVars])), Ytrain=yTrain2, Ytest=yTest2)

pt_end_RSBML = proc.time()
time_min = round( (pt_end_RSBML - pt_start_RSBML)/60, 2)
timespent_RSBML = paste("timespent_RSBML:", time_min[3], "min =");

### Stage 3 - Simple SBML Comparison - sv
pt_start3 = proc.time()
xTrain3_sv = data_train_all[, c(sLevelVars)]
xTest2_sv = xTest2[, c(sLevelVars)]
yTrain3 = data_train_all[, 9] ;

mySBMLtrain3_sv = SBMLtrain(Epoch=EpochN, S0=initS2_sv, Lamda=lamdaVal,
LearningRate=LRval, eta=1, xTrain3_sv, yTrain3)

predY3_sv = PredictedY(W=Weight(Similarity(eta=eta, mySBMLtrain3_sv$Rs, xTrain3_sv,
xTest2_sv)), Ytrain=yTrain3, Ytest=yTest2)

pt_end3 = proc.time()
time_min3 = round( (pt_end3 - pt_start3)/60, 2)
timespent3 = paste("Mega Subj-level timespent3", time_min3[3], "min = ");
```

BIBLIOGRAPHY

- Allison, P D (1999). *Logistic Regression Using the SAS System: Theory and Application*. SAS Institute Inc.
- Altman, N S (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*. 46 (3): 175–185.
- Beale, E M L (1970). Note on procedures for variable selection in multiple regression, *Technometrics*, 12: 909-914.
- Benjamini, Y, Hochberg, Y (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*. 57 (1): 289–300.
- Breiman, L (2001). *Statistical Modeling: The Two Cultures*. *Statistical Science*, Vol. 16, No. 3 (Aug., 2001), 199-215.
- Breiman, L, Friedman, J, Olshen, R and Stone, C (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.
- Carneiro, G, Nascimento, J C, Freitas, A (2012). The segmentation of the left ventricle of the heart from ultrasound data using deep learning architectures and derivative-based search methods. *IEEE Transactions on Image Processing* 21(3): 968-982.
- Chang, M (2012). *Paradoxes in scientific inferences*. Boca Raton, FL: CRC Press/Taylor Francis.
- Chang, M (2014). *Principles of scientific methods*. Boca Raton, FL: CRC Press/Taylor Francis.
- Chang, M (2016). What Constitutes Science and Scientific Evidence: Roles of Null Hypothesis Testing, *Educational and Psychological Measurement* 2017, 77(3) 475–488.
- Davies, J C, Wainwright, C E, Canny, G J, et al. (2013 Jun). VX08-770-103 (ENVISION) Study Group. Efficacy and safety of ivacaftor in patients aged 6 to 11 years with cystic fibrosis with a G551D mutation. *American Journal of Respiratory and Critical Care Medicine*, 187(11):1219-25.
- Efron, B (2020). Prediction, Estimation, and Attribution, *Journal of the American Statistical Association*, 115:530, 636-655.
- Efroymson, M A (1960). Multiple regression analysis, *Mathematical Methods for Digital Computers*. Eds. A. Ralston & H.S.Wilf, New York: John Wiley & Sons, Inc.

- Fisher, R A (1992). *Statistical Methods for Research Workers*. Springer, New York, NY
- Flume, P A, Liou, T G, Borowitz, D S, et al. (2012 Sep). VX 08-770-104 Study Group. Ivacaftor in subjects with cystic fibrosis who are homozygous for the F508del-CFTR mutation. *Chest*. 142(3):718-724.
- François-Lavet, V, Henderson, P, Islam, R, et al. (2018). An Introduction to Deep Reinforcement Learning, *Foundations and Trends in Machine Learning*: Vol. 11, No. 3-4.
- Hastie, T, Tibshirani, R, Friedman, J (2001, 2nd edn, 2009). *The Elements of Statistical Learning*. Springer, London.
- Hayes, T, Usami, S, Jacobucci, R, McArdle, J (2015 Dec). Using Classification and Regression Trees (CART) and Random Forests to Analyze Attrition: Results From Two Simulations. *Psychology and Aging*. 30(4): 911–929.
- Hoerl, A and Kennard, R (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1): 55-67.
- Hofmann, T, Scholkopf, B, Smola, A J (2008). *Kernel Methods in Machine Learning*.
- Hughes, T B, et al. (2015). Modeling epoxidation of drug-like molecules with a deep machine learning network. *ACS Central Science*, 1, 168-180.
- Kalita, J, Valentina E B, Samarjeet B, et al. (2018). *Recent Developments in Machine Learning and Data Analytics: IC3*.
- Kim, J, Kang, U, Lee, Y (2017). Statistics and Deep Belief Network- Based Cardiovascular Risk Prediction, *Healthcare Informatics Research*, 2017 July;23(3):169-175.
- Le Cun, Y and Bengio, T (1995). Convolutional networks for images, speech, and time series. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks*. Cambridge, MA: MIT Press.
- Lehmann, E L (1992). *Introduction to Neyman and Pearson (1933) On the Problem of the Most Efficient Tests of Statistical Hypotheses*. Springer, New York, NY.
- Lehmann, E L, Romano, J P (2005). *Testing Statistical Hypotheses*. Springer, New York, NY.
- Li, X, Wu, X (2014). Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. 2014-10-15. arXiv:1410.4281.

- Lu, L, Zhang, Y, et al (2017). Deep Learning and Convolutional Neural Networks for Medical Image Computing. Springer, Switzerland, 2017.
- MacQueen, J B (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1. University of California Press. pp. 281–297.
- Moss, R B, Flume, P A, Elborn, J S, et al. (2015 Jul). VX11-770-110 (KONDUCT) Study Group. Efficacy and safety of ivacaftor in patients with cystic fibrosis who have an Arg117His-CFTR mutation: a double-blind, randomised controlled trial. The Lancet. Respiratory Medicine, (7):524-33.
- Ngo, T A, Lu, Z, Carneiro, G (2016). Combining deep learning and level set for the automated segmentation of the left ventricle of the heart from cardiac cine magnetic resonance. Medical Image Analysis, 35:159-171.
- Pardilla-Delgado, E & Payne, J D (2017). The Deese-Roediger-McDermott (DRM) Task: A Simple Cognitive Paradigm to Investigate False Memories in the Laboratory. Journal of Visualized Experiments, 1–10, 10.3791/54793.
- Pearson, K (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. Philosophical Magazine. 2 (11): 559–572.
- Quinlan, J R (1986). Induction of decision trees. Machine Learning. 1: 81–106.
- Ramasubramanian, K, Singh, A (2017). Machine Learning Using R - A Comprehensive Guide to Machine Learning, Springer.
- Ramsey, B W, Davies, J, McElvaney, N G, et al. (2011 Nov). VX08-770-102 Study Group. A CFTR potentiator in patients with cystic fibrosis and the G551D mutation. New England Journal of Medicine, 365(18):1663-72.
- Ratjen, F, Hug, C, Marigowda, G, et al. (2017 Jul). VX14-809-109 investigator group. Efficacy and safety of lumacaftor and ivacaftor in patients aged 6-11 years with cystic fibrosis homozygous for F508del-CFTR: a randomised, placebo-controlled phase 3 trial. The Lancet. Respiratory Medicine, (7):557-567.
- Rokach, L, Maimon, O (2008). Data mining with decision trees: theory and applications. World Scientific Pub Co Inc.
- Schölkopf, B, Tsuda, K, Vert, J (2004). Kernel methods in computational biology, The MIT Press.
- Shalev-Shwartz, S, Ben-David, S (2014). Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press.

- Smith, J W, Everhart, J E, Dickson, W C, et al. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261–265). IEEE Computer Society Press.
- Smola, A, Vishwanathan, S V N (2008). Introduction to Machine Learning, Cambridge University Press.
- Stadler, M A, Roediger, H L, McDermott, K B (1999). Norms for word lists that create false memories. *Memory and Cognition*. 27 (3), 494–500.
- Sutton, R S, Barto, A G (1998). Reinforcement Learning, The MIT Press.
- Tibshirani, R (1996). Regression Shrinkage and Selection via the Lasso, *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 58, No. 1 pp. 267-288.
- Vapnik, V (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.
- Vapnik, V (1998). *Statistical Learning Theory*. John Wiley and Sons, Inc., New York.
- Wald, A (1950). *Statistical Decision Functions*. John Wiley and Sons, New York; Chapman and Hall, London.
- Williams, R J, Hinton, G E, Rumelhart, D E (1986 Oct). Learning representations by back-propagating errors. *Nature*. 323 (6088): 533–536.

CURRICULUM VITAE

