A Bayesian Rule Generation Framework for 'Omic' Biomedical Data Analysis

by

Jonathan Llyle Lustgarten

B.S. Computational Biology with Chemistry Minor, Carnegie Mellon University, 2004

M.S. Bioinformatics, University of Pittsburgh, 2006

Submitted to the Graduate Faculty of

School of Medicine in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2009

UNIVERSITY OF PITTSBURGH

School of Medicine

This dissertation was presented

by

Jonathan Llyle Lustgarten

It was defended on

April 9, 2009

and approved by

Shyam Visweswaran M.D., Ph.D.

Assistant Professor, Department of Biomedical Informatics, University of Pittsburgh

William R. Hogan M.D., M.S.

Associate Professor, Department of Biomedical Informatics, University of Pittsburgh

Robert P. Bowser Ph.D.

Associate Professor, Department of Pathology, University of Pittsburgh

Dissertation Advisor: Vanathi Gopalakrishnan M.S., Ph.D.

Assistant Professor, Department of Biomedical Informatics, University of Pittsburgh

A Bayesian Rule Generation Framework for 'Omic' Biomedical Data Analysis

Jonathan Llyle Lustgarten, M.S., Ph.D.

University of Pittsburgh, 2009

A Bayesian Rule Generation Framework for 'Omic' Biomedical Data Analysis

Jonathan Llyle Lustgarten, M.S.

University of Pittsburgh 2009

High-dimensional biomedical 'omic' datasets are accumulating rapidly from studies aimed at early detection and better management of human disease. These datasets pose tremendous challenges for analysis due to their large number of variables that represent measurements of biochemical molecules, such as proteins and mRNA, from bodily fluids or tissues extracted from a rather small cohort of samples. Machine learning methods have been applied to modeling these datasets including rule learning methods, which have been successful in generating models that are easily interpretable by the scientists. Rule learning methods have typically relied on a frequentist measure of certainty within IF-THEN (propositional) rules. In this dissertation, a Bayesian Rule Generation Framework (BRGF) is developed and tested that can produce rules with probabilities, thereby enabling a mathematically rigorous representation of uncertainty in rule models.

The BRGF includes a novel Bayesian Discretization method combined with one or more search strategies for building constrained Bayesian Networks from data and converting them into probabilistic rules. Both global and local structures are built using different Bayesian Network generation algorithms and the rule models generated from the network are tested on public and private 'omic' datasets. We show that using a specific type of structure (Bayesian decision graphs) in tandem with a specific type of search method (parallel greedy) allows us to achieve statistically significant higher overall performance over current state of the art rule learning methods. Not only does using the BRGF boost performance on average on 'omic' biomedical

data to a statistically significant point, but also provides the ability to incorporate prior information in a mathematically rigorous fashion for modeling purposes.

TABLE OF CONTENTS

LIST OF TABLES ..... **XI**

LIST OF FIGURES ..... **XVII**

GLOSSARY ..... **XX**

ACKNOWLEDGEMENTS ..... **XXIII**

1.0 INTRODUCTION **1**

1.1 PROBLEM DESCRIPTION **2**

1.2 THE APPROACH **4**

1.2.1 Thesis **6**

1.3 SIGNIFICANCE **7**

1.4 DISSERTATION OVERVIEW **8**

2.0 BACKGROUND **9**

2.1 HIGH-DIMENSIONAL 'OMIC' BIOMEDICAL DATA **9**

2.1.1 Genomic Data Generation **11**

2.1.2 Whole-Sample Proteomic Mass Spectra **14**

2.2 MODELING OF BIOMEDICAL DATA **19**

2.2.1 Machine Learning for Rule Model Generation **22**

2.2.1.1 Variable Selection and Transformation via Discretization **22**

2.2.1.2 Model Generation **24**

2.2.1.3 Performance and Heuristic Measures 24

2.2.2 Discretization 25

2.2.3 Rule Model Generation and Classification 27

2.2.3.1 Decision Tree for Rule Model Generation 28

2.2.3.2 Rule Learner (RL) 30

2.2.4 Bayesian networks 31

2.2.4.1 Components of a Bayesian network 32

2.2.4.2 D-Separation 34

2.2.4.3 Markov Blanket Model 35

2.2.4.4 Bayesian Local Structure 36

2.2.4.5 Model Generation using the K2 algorithm 40

2.2.4.6 Inference using Bayesian networks 43

2.2.5 Hybrid Rule Learning 43

2.2.5.1 Hybrid Decision Tree/Genetic Algorithm 45

2.2.5.2 Markov Blanket Bayesian network Hybrid Rule Learner 46

3.0 BAYESIAN RULE GENERATION FRAMEWORK 49

3.1 DESCRIPTION OF BRGF 50

3.1.1 Discretization 51

3.1.1.1 Efficient Bayesian Discretization (EBD) 51

3.1.2 Bayesian Global Structure 55

3.1.3 Bayesian Local Structure 56

3.1.3.1 Bayesian Local Structure - Decision Tree (BLS-DT) 56

3.1.3.2 Bayesian Local Structure – Decision Graph (BLS-DG) 58

3.1.4 Search Paradigms 60

3.1.4.1 Beam Search 61

3.1.4.2 Parallel Greedy Search 63

3.2 COMPUTATIONAL COMPLEXITY FOR THE ALGORITHMS 68

3.2.1 Greedy Search 68

3.2.1.1 BGRL 68

3.2.1.2 BLSRL_DT and BLSRL_DG 68

3.2.2 Beam Search 69

3.2.2.1 BGRL 69

3.2.3 Parallel Greedy Search 69

3.2.3.1 BGRL 69

3.2.3.2 BLSRL-DT and BLSRL-DG 70

3.3 METHODOLOGY FOR EVALUATION OF BRGF 70

3.3.1 Simulated Data 71

3.3.2 Biological Data 74

3.3.3 Experimental Design 76

3.3.4 Performance Measures 77

3.3.4.1 Accuracy 77

3.3.4.2 Balanced Accuracy 78

3.3.4.3 Relative Classifier Information 78

3.3.4.4 Running Time 80

3.3.4.5 Model Complexity (Number of Variables and Rules) 81

3.3.5 Statistical Analysis 81

4.0 SUFFICIENCY OF THE BAYESIAN RULE GENERATION FRAMEWORK 83

4.1 BAYESIAN DISCRETIZATION 83

4.2 BAYESIAN RULE GENERATION 88

5.0 EVALUATION OF THE BAYESIAN RULE GENERATION FRAMEWORK 91

5.1 BAYESIAN LOCAL STRUCTURE COMPARISON DECISION TREE VS. DECISION GRAPH 92

5.1.1 MDLPC 92

5.1.1.1 Results 92

5.1.1.2 Discussion 94

5.1.2 EBD 100

5.1.2.1 Results 100

5.1.2.2 Discussion 102

5.1.3 Discretization Comparison 108

5.1.3.1 Results 108

5.1.3.2 Discussion 111

5.1.4 Summary of Bayesian Local Structure Experiments 120

5.2 BGRL VS BLSRL-DG 121

5.2.1 Results 121

5.2.2 Discussion 123

5.3 COMPARISON WITH OTHER RULE LEARNERS 128

5.3.1 Results 128

5.3.2 Discussion 130

5.4 RESULTS ON UNPUBLISHED PROTEOMIC SETS 136

5.4.1 Results 137

5.4.2 Discussion 138

6.0 CONCLUSIONS AND FUTURE WORK 141

6.1 SPECIFIC FINDINGS 141

6.2 FUTURE WORK 144

6.2.1 Bayesian discretization methods 145

6.2.2 Beam Search for Local Structure 145

6.2.3 Heuristics for stopping specialization for Bayesian Decision Trees and Graphs 146

6.2.4 Bayesian Model Averaging 146

6.2.5 Heterogeneous Data Sources for analysis 147

6.2.6 Incorporation of Prior Knowledge 147

6.2.6.1 Prior Biological Knowledge derived from Literature 147

APPENDIX A 149

APPENDIX B 157

APPENDIX C 172

REFERENCES 176

LIST OF TABLES

Table 2-1. An example of a conditional probability table of the *Fatigue* node from Figure 2-9. *LC* is *Lung Cancer* and *CB* is *Chronic Bronchitis*, and *F* is *Fatigue* 36

Table 2-2. The parameters of the conditional probability table associated with the local structure for Figure 2-9, Figure 2-10, and Figure 2-11..... 38

Table 3-1. Nomenclature for labeling Greedy, Beam, and Parallel Greedy Searches..... 61

Table 3-2. The counts from the model in Figure 3-12 (1) which is the Bayesian Global Structure. Some of the parameters are 0.5 since in the training data, there is no training case that exemplifies that specific parent state 72

Table 3-3. The counts from the model in Figure 3-12 (2) which is the Bayesian Local Structure Decision Tree 72

Table 3-4. The counts from the model in Figure 3-12 (3) which is the Bayesian Local Structure Decision Graph 72

Table 3-5. The performance results of application of Bayesian Rule Generation using various structures 74

Table 3-6. Publicly Available Biological Datasets that will be used in the evaluation of the BRGF Framework..... 75

Table 3-7. Additional proteomic datasets that are not publicly available..... 75

Table 3-8. A brief description of the performance measures used 77

Table 4-1. The datasets used for the discretization comparison between MDLPC and EBD to prove sufficiency..... 84

Table 4-2. The accuracies of EBD, MDLPC, and No Discretization using a Naïve Bayes classifier 85

Table 4-3. The average RCI for EBD, MDLPC and No Discretization using a Naïve Bayes classifier 86

Table 4-4. Statisical comparison of EBD and MDLPC across accuracy and RCI. 87

Table 5-1. The percent accuracy averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers. 96

Table 5-2. The percent balanced accuracy averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers..... 97

Table 5-3. The percent RCI averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers. 98

Table 5-4. The comparison of accuracies of the different search strategies and local structure type using MDLPC discretization..... 99

Table 5-5. The balanced accuracy comparison of the different search strategies and the local structure type using MDLPC discretization..... 99

Table 5-6. The Statistical comparison of RCI of the different search strategies and local structure type using MDLPC discretization..... 99

Table 5-7. The percent accuracy averaged over 10x10 fold cross-validation for EBD discretization and Bayesian local structure classifiers.. 104

Table 5-8. The percent BACC averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers.. 105
Table 5-9. The percent RCI averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers. 106
Table 5-10. The comparison of accuracies of the different search strategies and local structure type using EBD. 107
Table 5-11. The comparison of BACC of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison..... 107
Table 5-12. The comparison of RCI of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison..... 107
Table 5-13. Comparison of percent accuracy of the Discretization Methods across Bayesian Local Structure Rule Learning (BLSRL) algorithms..... 116
Table 5-14. Comparison of percent balanced accuracy of the Discretization Methods across Bayesian Local Structure Rule Learning (BLSRL) algorithms..... 117
Table 5-15. Comparison of percent Relative Classifier Information (RCI) of the Discretization Methods across Bayesian Local Structure Rule Learning (BLSRL) algorithms..... 118
Table 5-16. Accuracy Comparison of EBD to MDLPC Across Search Strategies and Local Structure. 119
Table 5-17. BACC Comparison EBD to MDLPC across Search Strategies and Local Structure. 119
Table 5-18. RCI Comparison to MDLPC across Search Strategies and Local Structure..... 120
Table 5-19. The percent accuracies of the different search strategies for Bayesian global structure and the local structure BLSRL_DG_PG using EBD 124

Table 5-20. The percent balanced accuracies of the different search strategies for Bayesian global structure and the local structure BLSRL_DG_PG using EBD 125
Table 5-21. The Relative Classifier Information (RCI) of the different search strategies for Bayesian global structure and the local structure BLSRL_DG_PG using EBD 126
Table 5-22. The comparison of Accuracies of the different search strategies and local structure type using EBD 127
Table 5-23. The comparison of BACC of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison..... 127
Table 5-24. The comparison of RCI of the different search strategies and local structure type using EBD 127
Table 5-25. The percent accuracies of the different rule learning methods paired with the discretization method that gives it the highest overall accuracy..... 132
Table 5-26. The percent BACC of the different rule learning methods paired with the discretization method that gives it the highest overall accuracy..... 133
Table 5-27. The percent RCI of the different rule learning methods paired with the discretization method that gives it the highest overall accuracy 134
Table 5-28. The comparison of accuracies of the different rule learning methods using the discretization method which gives them the greatest accuracy 135
Table 5-29. The comparison of BACC of the different rule learning methods using the discretization method which gives them the greatest accuracy 135
Table 5-30. The comparison of RCI of the different rule learning methods using the discretization method which gives them the greatest accuracy 136

Table 5-31. A summary of the different rule learning algorithms in the BRGF strengths and weaknesses as seen from the results in Chapters 5.0 139

Table 5-32. The accuracy of four different rule learning algorithms on four unpublished proteomic datasets..... 140

Table 5-33. The BACC of four different rule learning algorithms on four unpublished proteomic datasets..... 140

Table 5-34. The RCI of four different rule learning algorithms on four unpublished proteomic datasets..... 140

Table 6-1. The percent accuracies of various rule learning methods using MDLPC. BGRL_G is the Bayesian Global Rule Learning with Greedy Search. 157

Table 6-2. . The percent BACC of various rule learning methods using MDLPC. BGRL_G is the Bayesian Global Rule Learning with Greedy Search 158

Table 6-3. The percent RCI of various rule learning methods using MDLPC. BGRL_G is the Bayesian Global Rule Learning with Greedy Search. 159

Table 6-4. The percent accuracies of various rule learning methods using EBD. BGRL_G is the Bayesian Global Rule Learning with Greedy Search. 160

Table 6-5. The percent BACC of various rule learning methods using EBD. BGRL_G is the Bayesian Global Rule Learning with Greedy Search. 161

Table 6-6. The percent RCI of various rule learning methods using EBD. BGRL_G is the Bayesian Global Rule Learning with Greedy Search. 162

Table 6-7. The number of rules generated using MDLPC across the four Bayesian local structure algorithms with a maximum of 8 possible variables per a path within the *local structure*. 163

Table 6-8. The number of variables used in the model over the different methods using EBD with a maximum of 8 possible variables per a path within the *local structure.* 164

Table 6-9. The time it takes to run one fold of the classifier with EBD with a maximum of 8 possible variables per a path within the *local structure*..... 165

Table 6-10. The average number of rules over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using MDLPC..... 166

Table 6-11. The average number of rules over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using EBD. 167

Table 6-12. The average number of variables over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using MDLPC..... 168

Table 6-13. The average number of variables over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using EBD. 169

Table 6-14. The run time(in minutes) for a single fold averaged over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using MDLPC..... 170

Table 6-15. The run time(in minutes) for a single fold averaged over all folds for the Bayesian global rule learning algorithms, C4.5, and RL using EBD with a maximum of 8 possible parents in minutes..... 171

LIST OF FIGURES

Figure 2-1. Generic workflow for the generation of biomedical 'Omic' data. 10

Figure 2-2. An example of the fluorescence of a genome microarray experiment result adapted from [88]. 13

Figure 2-3. Two sample mass spectrometry equipment for SELDI-TOF and MALDI-TOF MS. 16

Figure 2-4. Sample Spectra of both SELDI and MALDI-TOF techniques prior to baseline correction. 18

Figure 2-5. A sample methodology of how inference and refinement proceed for Expert derived models. 21

Figure 2-6. Decision tree and it's rule representation. 29

Figure 2-7. A Hypothetical Bayesian Network [63, 124, 132] 32

Figure 2-8. Some of the different graphs possible between three nodes *X*, *Y*, and *Z*. 34

Figure 2-9. A Markov-Blanket with target node *Fatigue* derived from Figure 2-7's Bayesian Network 36

Figure 2-10. Global and Local Structure Representation of part of a Bayesian Network 37

Figure 2-11. Two Bayesian Local Structures. Examples of a Decision Tree (Left) and a Decision Graph (Right) 38

Figure 2-12. A general characterization of a rule revision algorithm (top) and a hybrid rule algorithm (bottom).. 44

Figure 2-13. Two different DT/GA implementations with GA-SMALL being on the left and GA-large-SN on the right..... 45
Figure 2-14. Conversion of a Bayesian Network Model (left) to a Rule Model (right) where the certainty factor is expressed as the likelihood ratio of the conditional probability of the target given the variable value. 47
Figure 3-1. The Bayesian Rule Generation Framework. 50
Figure 3-2. Pseudocode for the EBD algorithm..... 54
Figure 3-3. A pseudocode representation of Greedy Constrained Bayesian Global Structure search. 55
Figure 3-4. The pseudocode for a Bayesian Local Structure Decision Tree. 57
Figure 3-5. The pseudocode for Constrained Greedy Bayesian Local Structure Decision Graph algorithm. 59
Figure 3-6. How Breadth First Marker Propagation (BFMP) allows for sample tracking in both decision trees and decision graphs..... 60
Figure 3-7. Beam Search for Constrained Bayesian Global Networks for rule generation..... 62
Figure 3-8. The pseudocode for Parallel Greedy Constrained Global Bayesian Network algorithm. 64
Figure 3-9. The pseudocode for the Parallel Greedy Bayesian Local Structure Decision Tree Algorithm. 66
Figure 3-10. The pseudocode for the Parallel Greedy Bayesian Local Structure Decision Graph algorithm. 67
Figure 3-11. The simulated data used to verify the Bayesian Structure Search algorithms. Originally developed by Dr. Visweswaran in [124]. 71

Figure 3-12. Examples of the structures generated from the simulated data. 73

Figure 3-13. The pseudocode for the Experimenter used in testing the different algorithmic and discretization method combinations. 76

Figure 4-1. The Bayesian global structure learned, the parameterization, and the rule generated by the BGRF when combined with EBD discretization 89

Figure 4-2. The Bayesian local structure decision graph learned, the parameterization, and the rule generated by the BGRF when combined with EBD discretization 90

Figure 5-1. The Accuracy with standard errors for all datasets using MDLPC or EBD as the discretization techniques and the four different Bayesian local structure algorithms. 113

Figure 5-2. The Balanced Accuracy with standard errors for all datasets using MDLPC or EBD as the discretization techniques and the four different Bayesian local structure algorithms. 114

Figure 5-3. The Relative Classifier Information with standard errors for all datasets using MDLPC and EBD as the discretization technique and the four different Bayesian local structure algorithms. 115

GLOSSARY

Bayesian Network– A Directed Acyclic Graph that contains a set of random variables that are connected by directed links (arcs or arrows) between nodes such that there is no cycle, and each node has a conditional probability distribution associated with it that is defined by those nodes with arrows going into it (parents) with the following equation: P(X|*parents(*X)) where this represents the probability of the states within node X given its parents.

Bayesian Network Structure Search – A method for creating an optimal Bayesian Network by searching over possible arcs (connection between two variables that is directed). Optimality is determined by calculating the probability of the structure and the data and selecting the structure which maximizes this probability. There are many methods for searching over possible arcs which include greedy search, best-first search, and parallel greedy search all of which seek to increase the probability of the model and the data (the joint probability).

Best-First Search – Expands the model that is closest to the goal (highest scoring), but contains the ability to backtrack if the expansion produces a worse mode. It is similar to depth-first in that you are not guaranteed to reach a global optimum.

Constrained Bayesian Network Structure – A specific type of Bayesian network that represents the semantics involved in a propositional logic rule. Particularly, the only relationships allowed in the network are from observed variables (as parents) to a target class (as child)

Data preprocessing – Steps performed for a specific data type, prior to learning models, to help remove known variance (mechanical or biological) from the data that occurs during acquisition. e.g., log 2 transformations, spectral alignment, etc.

Directed Acyclic Graph (DAG) – A graph containing nodes and edges that have a direction associated with every edge and there are no cycles (loops) within the graph.

Discretization – The process of transforming a continuous variable into a discrete one through the division of a number line into discrete intervals (e.g., (-∞ - 0), [0 - ∞) where there is a cut-point or split point at the value of 0)

D-separation – A set of rules for deriving conditional independence among random variables from a directed acyclic graph (DAG) where nodes in the DAG correspond to variables.

Greedy Search – A method that always chooses the best model for expansion without the ability to back-track. Once a model cannot be improved (generate a higher score), it terminates. It is known to be an effective search method due to its speed, but it is prone to reaching local maxima.

Inference – The application of a model (e.g., rule model) to a set of samples (e.g., patients) for the purpose of assigning a target value (the consequent) to each sample (prediction).

Local Bayesian Structure – A Bayesian network which, given a set of variables, can have a reduced number of parameters within the network by considering local contextual independencies (e.g., variable-value independencies).

Markov Blanket – A set of variables, that given the target variable, and that set of variables, all other variables in the network are d-separated from the target. Those variables included within the Markov Blanket of a target node are: The parents of the target, the children of the target, and the parents of the children of the target.

Markov Condition - A node is conditionally independent from its non-descendents given its parents within a Directed Acyclic Graph (DAG).

'Omic' biomedical data – While there are many other types of 'Omic' data, for the purposes of this thesis, when we reference 'Omic' data, we are specifically referring to proteomic and genomic data, which will be comprised of Whole Sample Mass Spectrometry and Gene Expression Data respectively. All samples within each dataset are from human subjects and have, as their target class (to be predicted), either a type of disease or normal.

Parallel Greedy Search – Uses greedy search to efficiently search a larger number of possible structures. By forcing multiple seed structures that are each unique (as suppose to only the highest scoring one of greedy search), it explores a larger set of possible structures than greedy search while utilizing the speed of greedy search for each seed model.

Rule – A propositional rule of the structure, IF <*Condition*> THEN <*Assertion*>. The condition contains variables with their respective values joined together using a conjunction (AND, OR, NOR, etc.). The assertion consists of a single variable (the target) and a specific value.

Rule Model – A set of rules grouped (OR'd) together to be used for prediction on a set of samples.

Sufficient – The ability to satisfy a requirement. This also means it does not have to be the only method possible, just one that can fulfill the requirement.

Variable – A feature within a dataset corresponding to a specific measurement. For 'Omic' data, this can be either a mass-to-charge (*M/Z*) for proteomic data, or a specific gene for Genomic data.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Vanathi Gopalakrishnan Ph.D. for her advice, support, and guidance during this research. Her patience, intuition, and commitment to supporting me has made this Ph.D. possible. I thank my committee members, Shyam Visweswaran M.D. Ph.D., William R. Hogan M.D. M.S., and Robert P. Bowser Ph.D. for their support and guidance. I additionally would like to thank Greg Cooper M.D. Ph.D. for his support.

I thank Toni Porterfield whose encouragement, conversations, and assistance has made my time at the Department of Biomedical Informatics wonderful. I would also like to thank her for all the logistical help that she has given me during my time here.

I thank Charles Dizzard and Yolanda Dibucci for helping make my stay at the Department of Biomedical Informatics enjoyable and assisting me throughout my years here.

I acknowledge the financial support of the National Library of Medicine, the NIH, and NIGMS, without which, I would not have been able to complete my doctoral degree.

Finally, I cannot thank enough my family and friends for supporting my endeavor in becoming an informatics researcher. Their support and confidence enabled me to pursue my dreams and overcome any barrier that I might have encountered during the course of my degree.

1.0 INTRODUCTION

The human body has some of the most complicated mechanisms for homeostasis and disease, among all flora and fauna. Doctors and scientists routinely analyze various components of the body to diagnose and/or give a prognosis based on clinical observations (variables). Within the past decade, genomics — the study of gene activation, and proteomics — the study of protein expression, have been utilized extensively to discover common patterns between people who have the same disease. This has been difficult due to the large number of variables and the orders-of-magnitude smaller number of samples being used in analysis [1, 2]. However, even with these challenges, the discoveries made have advanced our understanding of protein structure, disease progression and pathways, and have provided newer non-traditional diagnostic tests [2-10]. These advances have developed into potent applications like screening tests for prostate cancer, understanding the genetic differences between breast cancer types, and the tracing of hereditary genes influencing cancer [10-13].

These complex genomic and proteomic data, which are often referred to as 'Omic' biomedical data, present many unique challenges toward building predictive models for disease that are not only robust in terms of classification performance such as accuracy, but also understandable to their users, the biomedical scientist and the clinician. The latter, understandability of a model, can be easily accomplished by the use of rule-based models, which are a collection of conjunctive statements of the form IF <antecedent>-THEN <consequent>

with appropriate measures of confidence attached [14-16]. Each conjunctive rule has a set of variable-value pairs conjoined using an AND, as the antecedent, and a single target value as the consequent. Such rule models have been used previously in the analysis of proteomic data from Amyotrophic Lateral Sclerosis, a degenerative neurological disease [2, 6], as well as in knowledge discovery efforts in the domains of protein crystallization and blood pathogen prediction [17-20].

The crucial notion of "understandability" of rule-based models has promoted extensive research in the field over the past decades, leading to development of sophisticated rule-generation methods, such as inductive Decision Trees [21] and Rule Learners [2, 22, 23]. These computational methods have not only increased the accuracy of the learned models, but also have also helped detect significant variables that are indicative of disease states [2, 18, 23-25].

1.1 PROBLEM DESCRIPTION

The development of robust and at the same time understandable classification models is non-trivial. It can be highly rewarding since knowledge extraction is easier when using understandable models like IF-THEN rules as compared to more complex models like Support Vector Machines which have been previously translated into conjunctive rules for interpretability [14, 26]. Researchers have focused on many different aspects of rule learning, including efficiency of rule generation [27], increasing performance via addition of different preprocessing techniques [28], and encapsulating knowledge by the creation of knowledge bases which store rule models [29-35]. However, in all these efforts, the use of a non-probabilistic measure of uncertainty complicates inference [2, 14, 18, 25, 36-40].

For inference, a rule model, a collection of rules grouped together for a classification task, assigns a prediction (e.g., a disease state) to a set of descriptors for a sample (patient) by matching these to the proposition within an antecedent of a rule and assigning the consequent. However, this can result in a conflicting prediction (e.g., two rules with different consequents may apply to the same sample), and must be resolved. Since the measures of uncertainty for the rules are non-probabilistic, conflict resolution (deciding which of the conflicting rule(s) are to be applied) is often heuristic-based [2, 14, 38].

Rule learning from data also requires discrete variables. Specifically, both the antecedent and the consequent utilize discrete variable-value pairs. However, the observed variables within 'Omic' biomedical data are continuous. A method is required to transform the continuous variables into discrete variables. This process, known as discretization, is a requirement for any method that attempts to perform rule learning on 'Omic' biomedical data. Often, discretization is treated as a preprocessing step and thus ignored by rule learning algorithms; however, it has been shown that discretization interacts with classification algorithms, and thus is important to consider when developing a rule learning method [28, 41-49].

The goal of producing a robust classification rule model (models for predicting the class, such as disease state, of samples or patients) from 'Omic' biomedical data involves considering the following four rule characteristics which will be referred to as the 4 R's:

1) a **R**ule's discrete value within the antecedent. This means that the rule learning method must be able to convert continuous values of variables into discrete values, either using a preprocessing method, or within the rule generation method itself.

2) **R**ule conflict resolution mechanism, which needs to be robust; rules might conflict in their prediction and the resolution of these conflicts must be a well defined mechanism using proper measures of confidence in the correctness of a rule (certainty factor) [14, 16, 18, 19, 24, 37, 38, 40, 50].

3) **R**ule over-fitting; given a specific set of samples, rule generation might focus on a specific class of samples due to an imbalance and thus generate rules with high coverage of one class of samples, while not covering (matching the condition) for the other classes [1, 17, 19, 51].

4) **R**ule sparseness, which is the variable to sample size imbalance; this can cause difficulty in learning a generalizable model [16, 52-54].

Given the 4 Rs mentioned above, the central question is: can we develop a framework for rule learning that addresses each of the Rs when analyzing 'Omic' biomedical data?

1.2 THE APPROACH

To develop a framework that addresses the 4 Rs, two components must be developed in tandem: discretization and rule learning. Researchers have approached the latter component by combining rule learning with other classification algorithms through either the expansion and revision of rule models [16, 55], or an alternative rule generation mechanism [56]. The former has been developed as a separate preprocessing step from the classification algorithm [43, 46-49, 57-61].

For the rule learning component of the framework, researchers have utilized expansion and revision of the rule models to improve classification performance; however, it still suffers from the need for a seed or starting rule model [16, 40, 54, 55].

The development of an alternative rule generation method either utilizes an additional method that assists the standard rule learning method or changes the methodology for rule generation completely [16, 56]. These two approaches to alternative rule generation are defined as hybridization.

The overall goal of this dissertation is to create a hybrid method that addresses some characteristics of rule learning for 'Omic' biomedical data, which include: the sparseness of 'Omic' biomedical data, over-fitting caused by the sample to variable imbalance, and the use of probabilities as certainty factors for rule inference and conflict resolution [56, 62-64]. Though hybrid rule-learning algorithms have been previously developed, there remain significant difficulties in translating the alternative method into rules, which limits their application. Specifically:

a) does the new classification model directly translate to the semantics of a rule?

b) does the uncertainty of a rule (certainty factor) reflect the model from which the rule was generated?

This research describes a framework called the Bayesian Rule Generation Framework (BRGF), which attempts to combine a probabilistic method, specifically Bayesian networks (a probabilistic representation of variables as a directed acyclic graph where the directed arcs represent probabilistic dependencies between variables), with both discretization and rule generation. The outcome of this combination will be an ability to generate a model that is probabilistically optimized for rules and the ability to address the 4 Rs listed above. For the development of a Bayesian network, we will focus on two specific types of structures: a constrained *global structure*, which is a model that only allows observed variables to be parents of the target (directed arcs connecting a variable to the target) and combining the variable-values of the parents combinatorially, and *local structure*, which is a general form of the Global structure since it allows for contextual (variable-value) independencies in the parameterization though allowing variables to be only parents of the target [56, 63, 65-70].

This work is organized in two parts: first, we determine if using the BRGF is sufficient for generating probabilistic rules, and second, we evaluate the performance of the BRGF and compare it to the performance of other non-hybrid rule learning methods.

1.2.1 Thesis

First, it is my hypothesis that using the BRGF is sufficient to generate probabilistic rules.

Second, it is my hypothesis that using the BRGF leads to increased classification performance over standard rule learning methods.

Based upon the above theses, four specific claims are made, all of which are strong.

1. Bayesian Discretization and Bayesian Rule Generation is sufficient to generate probabilistic rules.
2. Under controlled conditions (constrained Bayesian networks), Bayesian local structure increases classification performance (on average) over Bayesian global structure for rule generation.
3. Under controlled conditions (constrained Bayesian networks) when combined with Bayesian rule generation methods, Bayesian discretization produces equivalent or greater performance (on average) than Fayyad and Irani [49] discretization (the de facto standard for discretization).
4. Under controlled conditions (constrained Bayesian networks), the BRGF leads to increased (on average) classifier performance over traditional rule learning methods.

1.3 SIGNIFICANCE

Although development of rule-base systems has led to biomarker discovery in many diseases such as lung cancer, hepatocellular carcinoma, and amyotrophic lateral sclerosis, the use of non-probabilistic rules (rules developed deterministically with non-probabilistic certainty factors) have hindered the ability to address the 4 Rs. In this work, I demonstrate that it is possible to use the BRGF to generate probabilistic rules that address the 4 Rs.

Using the BRGF, it is now possible to develop a specific type of Bayesian network that is optimized for the purposes of rule generation. It also affords additional advantages in terms of increased performance over standard rule generation techniques while using fewer overall variables, thus providing a more parsimonious list for further investigation by the scientist.

The incorporation of a probabilistic model for rule generation allows for evidence chaining and hence, generalizes the applicability of BRGF for modeling data from any domain of interest. Furthermore, it facilitates the incorporation of priors into the modeling framework. For example, using a probabilistic model, it is possible to use expert opinion to give priors of genes associated with a target disease and learn a model that accounts for these priors.

Probabilistic rule learning is a general technique that has the advantages of a probabilistic model with the understandability of rules. This allows for the application of probabilistic rule learning to other fields such as economics and protein crystallization, which also have issues with complex data and where prior information is available but under-utilized for modeling purposes.

1.4 DISSERTATION OVERVIEW

Chapter 2 provides background information on discretization, machine learning methods used in this work, and relevant prior work on the use of discretization and machine learning on biomedical data. Chapter 3 describes the Bayesian Rule Generation Framework including the different discretization methods, Bayesian network structures, and search strategies that are available, and how the BRGF is going to be evaluated. It also gives an example of how the framework can be applied to a biomedical dataset. Chapter 4 shows that the BRGF is capable of producing probabilistic rules (claim1). Chapter 5 shows the performance of the BRGF on biomedical genomic and proteomic datasets, comparing it not only to different algorithms within the BRGF, but also to two state-of-the-art rule learning algorithms (claims 2, 3, and 4). Chapter 6 presents conclusions with discussions on further development of the methods presented in this thesis.

2.0 BACKGROUND

This chapter provides the background and concepts required for generating probabilistic rules from Bayesian networks learned from biomedical data. Section 2.1 explains the types of biomedical data that will be used and some of the intricacies involved in acquisition and analysis. Section 2.2 reviews many of the critical machine learning concepts that were used to formulate the hypotheses tested in this research.

2.1 HIGH-DIMENSIONAL 'OMIC' BIOMEDICAL DATA

High-dimensional 'Omic' biomedical data offer the opportunity to investigate many aspects of biology concurrently. This allows for exploratory studies to query multiple types of variables (e.g., genes or proteins) in a single experiment allowing for a large increase in speed of knowledge discovery. They also offer the ability to investigate many variables in a cost-sensitive manner.

Both types of 'Omic' biomedical data used within this thesis share three similar challenges: the sparseness of the data, the ratio of variable size to sample size (which is several times greater than one), and the variability of measurements across samples caused by both experimental and biological variation. Given these challenges, however, both genomics and proteomics offer the ability to screen many samples for multiple biological indicators (e.g., in

genomics, gene activation level and in proteomics, relative protein expression level) in a high-throughput fashion producing a wealth of data that can be used for prospective or exploratory analysis. The commonalities between genomics and proteomics continue beyond challenges and advantages; there are also similarities in the processing and acquisition of data [29, 30, 71-76].

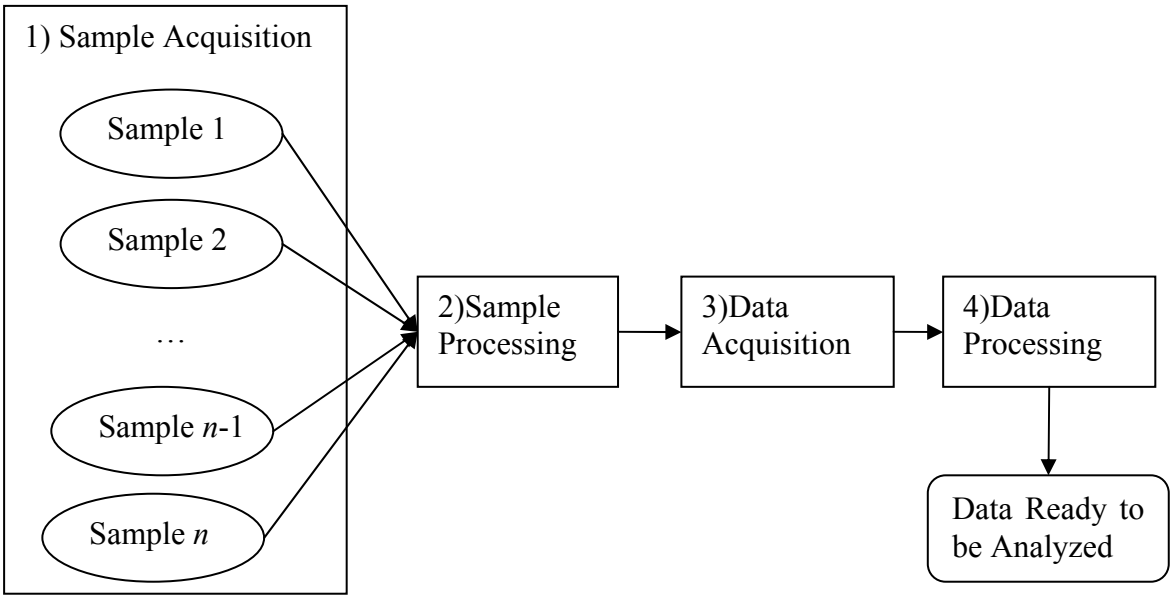


Figure 2-1. Generic workflow for the generation of biomedical 'Omic' data.

As seen in Figure 2-1 there are four areas that, while common in terms of the overall process, differ depending upon whether the goal is to produce data from genomic or proteomic platforms. The next two sub-sections (Section 2.1.1 and Section 2.1.2) discuss some of these steps within each area of the workflow with respect to genomic and proteomic data generation.

2.1.1 Genomic Data Generation

Within genomics, there are multiple types of data that can be generated, including "knock-out" data (silencing of a specific gene then measuring outcome) and gene microarray expression data [77]. For the study of disease, particularly to discover those genes that are correlated with the expression of disease, generation of gene microarray expression data is the preferred method due to its ability to measure many variables in parallel in a high throughput fashion [33, 74, 76, 78-88]. Focusing the analysis on this highly used data-type can be advantageous; however, there are many different methodologies for data acquisition including serial analysis of gene expression (SAGE), oligonucleotide arrays, and cDNA arrays, all of which provide the opportunity to survey many different genes in a single experiment [29, 30, 71, 73, 74]. Depending on the methodology used to acquire the data, certain processing procedures are required before analysis (e.g., logarithmic transformation to account for fold change within cDNA arrays). We will discuss cDNA fluorescence microscopy in the following paragraphs as an exemplary technique for genomic data generation.

The sample acquisition component of a genomic experiment usually involves the biopsy of a patient's tissue (e.g., breast, liver, brain, etc.), which is then processed for the mRNA it contains. Location in the body and time at which the biopsy is taken play a critical role in the results obtained from downstream analysis. Depending on the disease, e.g., cancer, the area of biopsy, whether it is near the margin (change point between cancerous and non-cancerous cells) or in the center can affect which genes are active. In general, two biopsies are taken from a patient—one diseased and one non-diseased—and both are used in later steps [8, 9, 73, 74, 84, 86, 87, 89].

After the sample has been acquired properly (e.g., the cancer biopsy acquired is actually from cancer tissue, and the normal biopsy is from normal tissue), the next stage is sample processing, where the mRNA is extracted, isolated, purified, and then reverse transcribed to create cDNA with fluorescent tags. For DNA microarray analysis, specifically those that use comparative fluorescent microscopy, the diseased biopsy sample mRNA usually has one color fluorescent tag (e.g., red) associated, while the other mRNA from the non-diseased biopsy sample has a different color associated with the cDNA (e.g., green) [7, 29, 30, 71-75, 79, 81, 87, 88, 90-94]. Once the respective samples are tagged with fluorescent markers, which in this case is green for healthy and red for diseased, the samples are ready for data acquisition.

This process is repeated for many patients and thus often requires automation; however, data acquisition is not complex in the sense of the mechanical process. Robotic machines can spot thousands of samples at a single time; however, one component of data acquisition is critical to the success of future analysis: the choice of which genes to probe for activity. The choice of the probes (selected genes) for the experiment represents a significant strength of high-dimensional analysis [30, 71-73, 75, 90]. In Figure 2-2, we see the result of the data generation step. Within Figure 2-2, we see the relative intensities (red lower expression, green higher expression than baseline) and the differentiation between the low and the high. The next step is data processing.

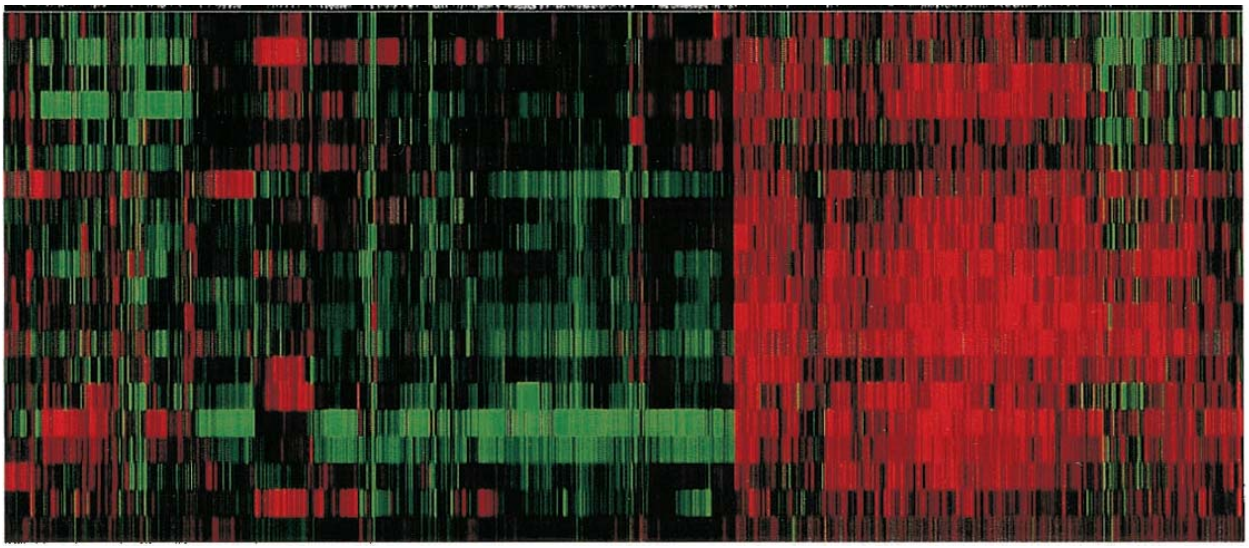


Figure 2-2. An example of the fluorescence of a genome microarray experiment result adapted from [88].

Data processing for cDNA fluorescence microscopy involves two main steps: normalization of intensities and the validation of the data. The latter refers to the process of comparing genes between the biopsies that are non-correlated with the disease (e.g., house-keeping genes). Since these genes should be similar across diseased and non-diseased tissues, they should be very close in fluorescence intensity, and they can be used to measure the variation within the experimental procedures. If there is a detected difference within the validation genes, the data is discarded, and the data generation process starts from the sample acquisition step again. Given that the samples have been validated, a correction is necessary for the fold-change which can occur during the comparison of intensities. If we have Sample A with an intensity of 4 units while Sample B has an intensity of 2 units, the fold change is either 2 or ½ depending upon which direction is measuring. This fold change can be difficult to handle especially when comparing many different samples. One method that is commonly employed is to take the $\log_2$ of ratios that way the numbers of the ratio would be 1 or -1 respectively thus normalizing the scale and allowing simpler comparisons [74].

While what has been listed in the previous paragraphs is representative of the methodology used, there are many different variations that can be employed. For the purposes of this thesis, we will assume that all genomic data is from cDNA microarrays and have been processed through to step 5 in Figure 2-1.

2.1.2 Whole-Sample Proteomic Mass Spectra

While genomics offers insight into which genes are activated within a cell, often it is difficult to obtain samples due to the invasiveness of the biopsy procedures (e.g., brain tissue). The final product of the genes (proteins), will also not necessarily be known given that most proteins have post-translational modifications that affect the structure beyond what can be predicted from the gene sequence [4]. Ultimately, proteomics studies the form (structure), function, and expression of proteins [95]. It allows a scientist to understand the interaction of a gene on the mechanistic level and further understand the complexities that can be associated with disease [10]. There are many different forms of proteomic analysis, e.g., Whole-sample, Fractionated Sample, and tandem Mass Spectrometry (MS/MS). They all follow the same high-level steps in Figure 2-1. For the purpose of this thesis, we will focus on whole-sample mass spectrometry, which is a very useful technique for survey because it can be performed on easily acquired body fluids (blood serum, nipple aspirate fluid, cerebrospinal fluid, etc.) [12, 96].

Although whole-sample mass spectrometry offers many advantages, including higher throughput (faster sample acquisition and preprocessing for data acquisition [12, 96]), it has other challenges that occur throughout the steps in Figure 2-1 including sample preparation and preprocessing that can make the overall analysis more difficult [1, 6, 10, 97-99]. These challenges include an increase in noise due to location in the body from where the sample is

taken (Step 1, e.g., disease is in organ yet samples are from blood serum), complex processing of whole sample (Step 2, e.g., desalting and removal of most abundant protein, etc), complex molecular interactions when acquiring spectra (Step 3, e.g., matrix thickness, protein/protein interaction, partial ionization, more than one peak one protein relationship), and complex post-processing (Step 4, e.g., spectra alignment, total ion current normalization, baseline subtraction, etc.) [1, 97, 100].

For sample acquisition in whole-sample mass spectrometry, one can use a surrogate fluid (CSF, blood serum) instead of a tissue biopsy; however, it must first be proven through analyses (Western blotting, immuochromatography, etc.) that surrogates such as these contain markers for the diseases being investigated, as done in [4, 6, 9, 101-104]. Once the surrogate fluid has been extracted for analysis, the next step (Step 2 in Figure 2-1) requires processing of these samples.

Biological samples often contain impurities (e.g., salt) and non-disease specific proteins (e.g., hemoglobin), as well as having massive concentrations of other proteins like albumin. Procedures such as desalting, and removal of high-concentration peptides/proteins is essential for detecting the less-concentrated proteins. Depending on the method used for producing the mass spectra, Surface Enhanced Laser/Desorption Ionization Time-of-Flight Mass Spectrometry (SELDI-TOF MS) and Matrix Assisted Laser/Desorption Ionization Time of Flight Mass Spectrometry (MALDI-TOF MS), certain preprocessing steps are more important than others. MALDI-TOF requires more sample preprocessing including desalting and removal of the most concentrated proteins [105], while SELDI-TOF allows for less sample processing [96, 102, 106, 107]. For the purposes of this thesis, we will analyze both SELDI- and MALDI-TOF data, and we will assume that all of the aforementioned procedures (desalting, removal of the most concentrated protein) that are relevant to the specific platforms have already been performed.

Data acquisition, which is the next step (Step 3 in Figure 2-1), involves preparing the sample for the mass spectrometer as well as optimization of the mass spectrometer experimental settings; it has multiple requirements. The experimenter must choose not only the optimal substrate to mix with the protein in order for transferring the laser power to the molecule for ionization (matrix), which can be very different depending on the method (e.g., SELDI or MALDI), but also the type of machine (e.g., Ciphergen Chip Reader, MALDI-TOF mass spectrometer). Within these machines, there is also the choice of laser power, which is critical in the ionization step. If the laser power is too high, the proteins will be shattered apart by the energy in the laser, but if it is too low, the amount of energy transferred upon impact of the laser would not be enough to ionize a large portion of the proteins. For the purposes of this thesis, we will assume that all mass spectrometry data was generated, which consists of a set of mass-to-charge (*m/z*) ratios along with the intensity of that peak, from an optimized protocol for the data

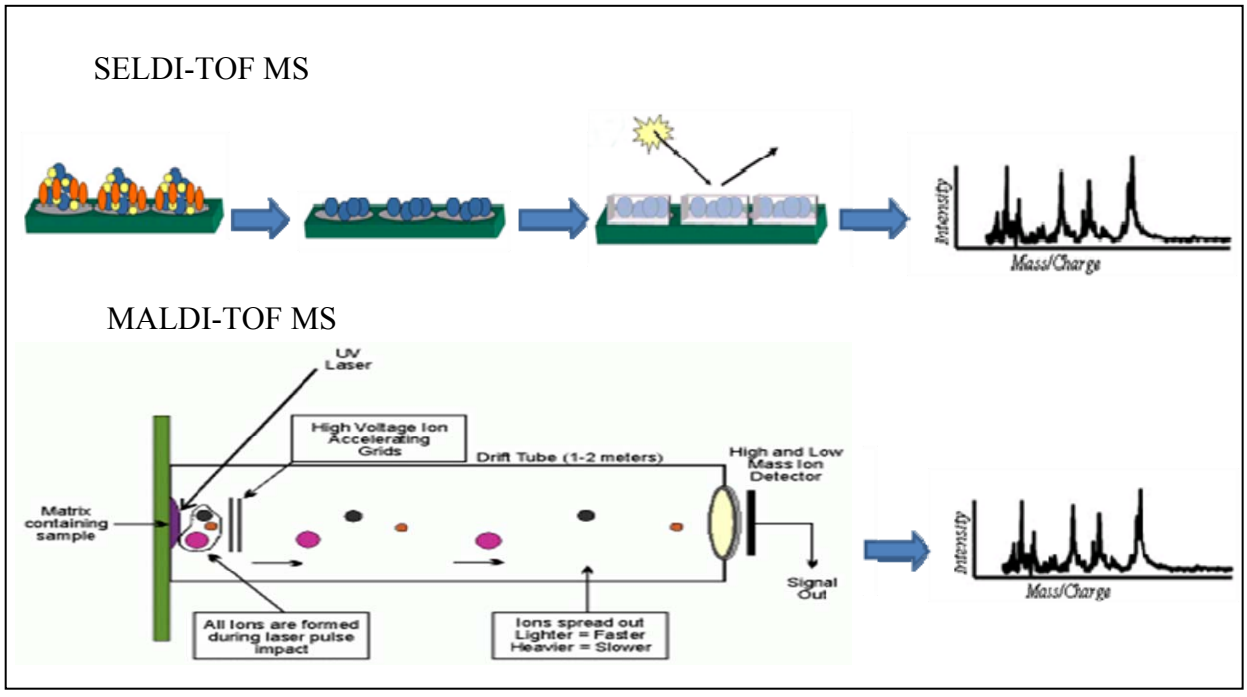


Figure 2-3. Two sample mass spectrometry equipment for SELDI-TOF and MALDI-TOF MS.

acquisition including experimental choices of matrix type, laser power, etc.

An important aspect of mass-to-charge ratios is the many-to-many relationship between *m/z* and protein (one protein can produce many *m/z*s and one *m/z* can represent many proteins). This is due to the ionization of a protein by the laser. When ionized, it is possible that a protein can take multiple charges (1, 2, 3) of which depending on the charge, it causes the ionized protein to behave differently when acquiring the spectra. A protein that has a molecular weight of 10,000 daltons with a double charge can appear at 5,000 *m/z* while if it has a single charge, it would appear at the 10,000 *m/z* peak. From this behavior, it is possible to have a peak represent multiple proteins by having two proteins, one with a molecular weight of 10,000 daltons and carrying a double charge while another with a molecular weight of 5000 daltons and carrying single charge. Both proteins would appear at the *m/z* peak of 5,000.

Given that we have acquired the spectra through an optimal protocol (Step 3), we proceed to data preprocessing (Step 4) which is essential for further analysis. This step can include various mathematical techniques such as total ion current normalization, baseline subtraction, spectra alignment, peak height correction via square root transformation [4, 105]. Baseline correction and spectra alignment are critical to the analysis of this data and are generally a requirement. Baseline correction is necessary since the base concentration of protein can be different per sample, and makes comparison across samples, specifically the relative intensities, error-prone unless made roughly equivalent. Spectra alignment allows for the correction of the random drift that occurs in all time-of-flight spectroscopy experiments [108, 109]. If there were no random drift, technical replicates would look the same across spectra (the intensities might vary, but each *m/z* would align). The goal of alignment is to correct for this drift, such that the *m/z* values measured across different samples correlate [108-110]. A program that conveniently

allows for all of these data processing techniques, SpecAlign©, was used to process all of the spectra. We performed baseline correction and spectra alignment using SpecAlign©, with the default parameters for baseline subtraction, and the default parameters for the Recursive Alignment via Fast-Fourier Transform (RAFFT). Proteomics offers a significant opportunity: if one can analyze these spectra for significant peaks (i.e., *m/z*s), it can help direct further investigation (eventual sequencing if necessary [111]) and speed up the proteomic biomarker discovery process.

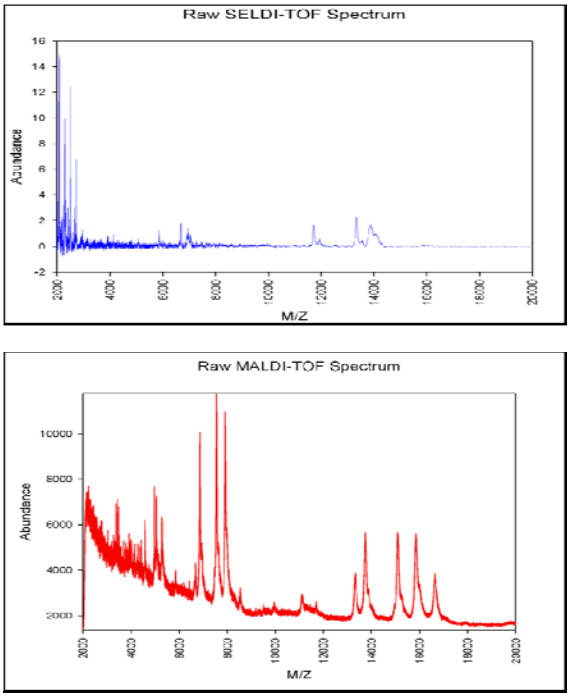


Figure 2-4. Sample Spectra of both SELDI and MALDI-TOF techniques prior to baseline correction.

In this thesis, all data will either be Genomic Microarray Expression data or whole sample MALDI-/SELDI-TOF Proteomic Mass Spectra, which having been processed through steps 1-4 in Figure 2-1, are ready for analysis. There are many methodologies one can employ in the analysis of these data, including machine learning techniques that aim to model the data. If one can model 'Omic' data effectively, new diagnostics, molecular understanding, and disease

differentiation are possible. The next section, Section 2.2, reviews the pertinent techniques used to generate models from these 'Omic' biomedical datasets within this thesis.

2.2 MODELING OF BIOMEDICAL DATA

Some of the earliest modeling systems applied to biomedical data used propositional rules, IF *antecedent* THEN *consequent*, derived directly from expert knowledge. The antecedent consists of one or more variable-value pairs (e.g., variable A >= 10) joined via some conjunction (e.g., AND, NOR, XOR, etc), while the consequent usually has a single variable-value pair denoting the target value (e.g., CLASS = SICK). These early rule-based systems related different experts' opinions (a rule/hypothesis) along with some measure of how sure the expert was of the knowledge (certainty factor)[112, 113]. As more knowledge/expert opinions (additional evidence on existing rules) are added to the system, the certainty factors are updated. Some of the earliest implementations of these rule base systems used a certainty factor that had a range from -1 to 1 where -1 related to the belief of the expert that the rule proposed is certainly false, while 1 represented the expert's belief that the rule is certainly true [1-3].

A system developed at Stanford University, called DENDRAL, often acknowledged as the progenitor of many rule-base systems including MYCIN and Prospector (discussed later), has two main parts, Heuristic Dendral [114] and Meta-Dendral [112]. Heuristic Dendral uses a knowledge base of chemistry and data from a system that measures chemicals (e.g., NMR, IR, Liquid Chromatography) to propose possible chemical structures that explain the data. Meta-Dendral takes possible chemical structures and the data, and tries to generate hypotheses that represent the relation between the chemical structure and the data (e.g., the structure causes the

data to exhibit a specific pattern) [112, 114]. Both of these systems work together to refine the knowledge base after every hypothesis generation, producing better possible chemical structures in future analyses. This is just one example system that utilized rule generation and rule inference for knowledge acquisition.

The system MYCIN [115] performs diagnoses of bacterial infection and recommends the antibiotic along with the dosage. The expert uses a simple interface answering yes/no questions through text prompts to receive the treatment recommendations. If an expert feels the diagnosis is wrong, they can modify the knowledge base by modifying the stored certainty factor. There is also the capability to add rules to the knowledge base. One of the main components of MYCIN is the chaining of rules that allows for inference on the final target given the input using certainty factor combination, a complex process that has a series of equations depending on the relationship between the rules and the antecedents in each rule. Research has shown that the MYCIN certainty factor implementation violates some of the tenets of probability including some independence assumptions, and Heckerman [38] showed that a transformation into a more probabilistic measure (e.g., likelihood ratio), allows for more flexible updating of certainty factor as well as combination of multiple rules without violation of the laws of probability.

Within a few years of the deployment of MYCIN, a probabilistic rule-based system was developed at SRI International called PROSPECTOR [116] to act as a consultation system to assist geologists working in mineral exploration [117]. It created semantic networks (where each connection between concepts and resultant conclusions had a specific meaning) using the probabilities assigned by the expert. It was mainly used in prospecting for ore deposits by comparing the current data with the stored knowledge base. It had the ability to ask for more information if needed and suggested which model is best suited and where the best drilling sites

might be within the geological data given to the system. These systems, along with other expert based systems such as CASNET, had difficulty keeping the model coherent since these expert-derived concepts were entered individually by experts [118].

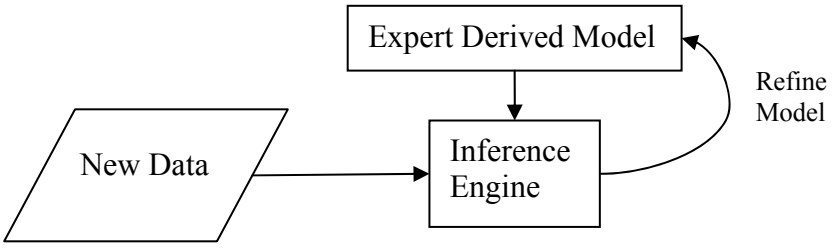


Figure 2-5. A sample methodology of how inference and refinement proceed for Expert derived models.

In all of the previously mentioned systems, a critical component is the inference engine that applies the rules in the model to a new unseen data to make predictions. In the case of rule-based systems, this requires the matching of the variables and their values to the rule's antecedents. If any variable in the rule's antecedent is not in the new data, then the rule would not be applied. Given that a set of rules applies to a given test case, a conflict resolution mechanism is employed to "choose" which assertion from the set of assertions contained within the rule set would be appropriate to apply. For example, let us assume that a test case has two rules that apply to it, one where the consequent is Class=Sick, while the other rule has the consequent Class=Healthy. The rules' certainty factors (CF) are 0.51 and 0.49 respectively. Using a majority conflict resolver, it would select the consequent Class=Sick as the correct one for that particular sample. There are also other ways to perform inference wherein a weighted voting mechanism can be employed for each prediction of a test case. The rules whose antecedent matches the test case have their consequent weights calculated combining the CFs of those rules whose consequent match. Then majority vote is utilized for class prediction.

While some of these systems that are still in use today have been shown to be very effective, they were acknowledged to be limited by the requirement of expert input in the development of the knowledge base [113, 119]. Therefore, the use of machine learning techniques to generate the model for classification has become more popular, particularly in the case of exploratory studies, where it is incredibly difficult to extract expert opinions about the thousands of variables that can be measured. The next section, Section 2.2.1, discusses some of the common components of machine learning for model generation.

2.2.1 Machine Learning for Rule Model Generation

Some important common components of machine learning for rule model generation are as follows:

1) Variable Selection and Transformation
2) Model (Hypothesis) Generation
3) Application of Model (Inference)

The next sections (Sections 2.2.1- 2.2.5) describe these components with respect to the rule models discussed previously.

2.2.1.1 Variable Selection and Transformation via Discretization

Although many possible methods exist for the selection and transformation of variables, one transformation that is required for the learning of propositional rules is the conversion of continuous variables into discrete variables, which can be performed by discretization. The process of discretization transforms a continuous-valued variable into a discrete one by creating a set of contiguous intervals (or equivalently a set of cut-points) that together span the range of the variable's values. The set of intervals produced by a discretization algorithm is called a

discretization model or simply, a discretization. Discretization algorithms can be broadly categorized into two types: unsupervised methods, which do not use any information about the target variable (class values of samples), and supervised methods, which do. Examples of unsupervised methods include equal-width discretization which divides the range of values of a variable into a user specified number of intervals, and an equal-frequency method that takes as an input a user-specified number of intervals and divides the samples up such that there are an equal number of samples in each interval. Compared to unsupervised methods, supervised methods tend to be more sophisticated and usually yield better discretization from the data automatically. Most supervised discretization methods utilize a score to measure the goodness of a set of intervals (where goodness means how well the discretized variable predicts the target variable).

Unlike unsupervised discretization, supervised discretization has the capability of determining that a variable has no relation to the target. This transforms a continuous-valued variable into a discrete one by creating a single interval from ($-\infty$ to ∞). These variables, which do not receive even a single cut point, are eliminated since all information is "smoothed" out by representing it as a single interval. This enables irrelevant variables, and hence noise in the data, as determined by the scoring metric, to be removed [46, 59]. This is especially critical in genomics and proteomics where many of the variables (e.g., genes and mass-to-charge ratios) are irrelevant to the classification problem. Therefore, the application of discretization to biomedical data can have multiple benefits (transformation and selection), and choosing a specific discretization method impacts the development of a classification model directly.

A second way to categorize discretization methods is as univariate and multivariate methods. Univariate methods discretize a continuous-valued variable independently of all other

predictor variables (observed variables that are not the target) in the dataset, while multivariate methods take into consideration the possible interactions of the variable being discretized with respect to the remaining predictor variables in the dataset. Univariate techniques have a distinct advantage of running time when compared to multivariate techniques (one variable at a time vs. multivariate combinatorics), and given that the 'Omic' biomedical datasets we are using have large numbers of predictor variables (> 5000), we have restricted this work to only univariate discretization methods. The net effect of these considerations is that, for the purposes of this thesis, we consider only supervised, univariate discretization algorithms.

2.2.1.2 Model Generation

When generating a rule model for classification, many permutations and combinations of variable-values are possible (e.g., if selecting a subset of variables, many different subsets can be possible). In order to build a classifier (a model used for classification), often an intelligent search algorithm (a basic definition of artificial intelligence [120]), which looks for the "best" model (according to some score or heuristic), is necessary to find an optimal model given the conditions specified by the algorithm (called the algorithm's bias [121]). This search for a single best model is called Model Selection. For the experiments listed in this thesis, we have used only algorithms that perform model selection.

2.2.1.3 Performance and Heuristic Measures

The methodology of choosing an "optimal" model typically involves some measurement of "fitness," which tries to encapsulate how well the model fits the underlying distributions of the data. This concept can be captured in a myriad of ways. A common and practical method is to choose the parameters of model such that the model predicts well on test data. These

performance measures, such as Accuracy, Balanced Accuracy, Relative Classifier Information [122] all require inference to be performed. This means that the learned model has to be applied to a test set of data to gather the unbiased estimates of the performance measures mentioned above.

While the previously mentioned measures require inference on a test set (a set of samples generally different from those that the model was learned on selected for testing the accuracy of the model) it is possible to utilize a measure that calculates how well the model fits the data without performing classification. A very popular method called K2 [70] calculates a score based on Bayesian principles that allows the user to estimate how well the model explains the data. K2 makes several assumptions to compute the score efficiently (which I shall describe in later sections), but it allows one to select models with minimal computational cost and without the need to split datasets into training and test subsets. The K2 score has been used by many researchers for measuring the goodness of a model [43, 63, 67, 68, 70, 119, 123-126].

In this thesis, we will use both the performance measures and the K2 score, for model selection. Further details are given in Sections 2.2.4 and 3.3.4.

2.2.2 Discretization

There has been significant research into discretization methodologies leading to the development of such algorithms as Fayyad and Irani's Supervised Minimum Description Length Principle Criterion (MDLPC) [49], which has become the de facto method [46, 127]. Although there have been subsequent developments, MDLPC remains the standard method. The data for a predictor variable (observed variable) X and target variable Z consist of a list of instances (samples) described by pairs of values: the value of the continuous predictor variable and the corresponding

value of the target variable. For example, suppose the target variable takes discrete values of either T or F as values, and the predictor variable is continuous. Then the data might be the following: ((2.3, T), (4.6, F), (1.5, T)). If we sort the list by the first term of each pair, we obtain a sorted list which, for this particular example, is the following: ((1.5, T), (2.3, T), (4.6, F)). The sequence *S* of the target values in the sorted list is "TTF". The discretization algorithms typically partition *S* into a list of mutually exclusive and exhaustive subsequences of *S*. Thus, these algorithms consider only the order of the continuous variable's values and not the distance between the values.

MDLPC is a supervised discretization algorithm that uses the entropy minimization heuristic for recursively selecting cut-points in the range of a continuous-valued variable. For a variable *X*, given a sequence *S* and a cut-point *C*, the entropy *E*(C; *S*) of the partition induced by *C* is given by:

$$E(C;S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2) \tag{1}$$

where, S_1 and S_2 are the partitions of *S* induced by *C*, $|S|$ is the cardinality of *S* and *Ent*(*S*) is the entropy of *S*. The MDLPC algorithm selects a cut-point *C* from all possible cutpoints that minimizes *E*(*C*; *S*) and then recursively selects a cut-point in each newly created interval in a similar fashion [49].

A criterion based on the Minimal Description Length Principle (MDL) is applied to terminate the recursion that creates the intervals. Recursive partitioning of a sequence *S* terminates if and only if:

$$Gain(C;S) > \frac{\log_2(n-1)}{n} + \frac{\Delta(C;S)}{n} \qquad (2)$$

where,

$$Gain(C;S) = Ent(s) - Ent(S_1) - Ent(S_2)$$
$$\Delta(C;S) = \log_2(3^k - 2) - [k \cdot Ent(S) - k_1 \cdot Ent(S_1) - k_2 \cdot Ent(S_2)]$$

Gain(*C*; *S*) is the information gain of partitioning *S* at the cut-point *C* and k_i is the number of distinct values present in subsequence S_i. Thus, in Equation 2, if the *Gain* on the left hand side is greater than the MDL value on the right hand side, cut-point *C* is accepted; otherwise discretization terminates for the subsequence under consideration. The MDLPC algorithm is one of the most widely used discretization algorithms in machine learning.

There are also other discretization methods available including three unsupervised techniques (Gaussian, Equal Width, and Equal Frequency), however this thesis focuses on comparison of the three supervised techniques (those that utilize class information) since they have been proven to perform better especially on high-dimensional datasets [28, 46, 47, 49].

2.2.3 Rule Model Generation and Classification

To discuss rule model generation and classification, first we introduce some terminology. A target variable (sometimes called the target) within the dataset is the variable that the researcher wishes to predict, e.g., the diagnosis of cancer would be considered the target variable. Rules are **IF** *Antecedent* **THEN** *Consequent* structures (IF-THEN) where the antecedent consists of variables and their associated values and the consequent is the target with a specific *variable-*

value. A knowledge base is a set of rules (or a set of models) that have been validated or agreed upon and are used to further assist inference. A rule has an associated confidence measure called a certainty factor that represents how much support exists for that specific rule. The coverage of a rule is the number of samples from a dataset that match the variable values that occur within the antecedent of the rule. The inductive strength of a rule is the number of samples from a dataset that have not previously been covered by any other rule. A True Positive (TP) of a rule is a sample that matches both the antecedent and the consequent, while a false positive matches only the antecedent. A False Negative (FN) is a sample that matches the consequent but not the antecedent, while a True Negative (TN) matches neither the antecedent nor the consequent.

Given these definitions, we will discuss some pertinent classification algorithms that have been developed from the need to analyze datasets that were either too large in terms of the observed variable size to acquire expert opinion, or exploratory, such that the expert may not have an opinion on what rules should exist. The following sections (2.2.3.1 and 2.2.3.2) describe two different rule learning algorithms, one that subsets the samples (patients) without replacement, and another that does sampling with replacement. Sampling without replacement corresponds to a decreasing sample space (fewer samples) after every subset generation. Sampling with replacement allows a sample to be in multiple subsets creating an overlap in terms of samples within the subsets (coverage).

2.2.3.1 Decision Tree for Rule Model Generation

Decision tree algorithms use a sub-setting technique that splits the data by variable-values that optimize the tree according to different measures that are characteristic of the particular methods implemented (see Figure 2-6). Commonly, the entropy measure (Equation 3) is used such that every split (differing values of a variable) within a decision tree decreases the overall entropy of

the dataset with the goal of creating a tree that has the lowest entropy (smallest number of unique classes) [21, 118, 128, 129] given by the following equation:

$$Entropy(D) = \sum_{t} p_t \log_2 p_t \tag{3}$$

where D is the data, t are the values of the target within the data and p_t is the probability of the target having the value t.

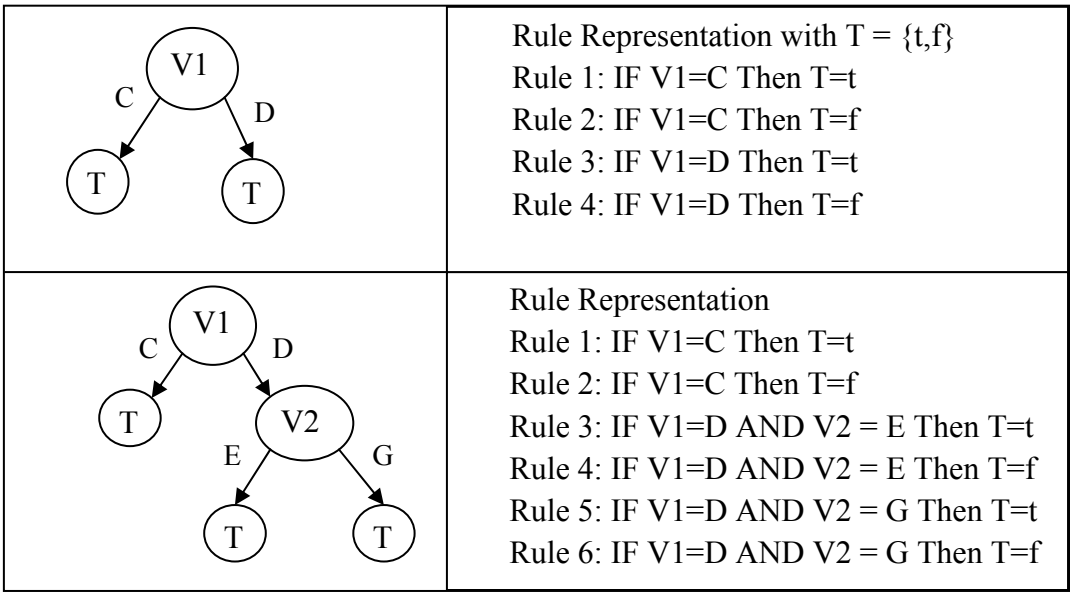


Figure 2-6. Decision tree and it's rule representation.

In Figure 2-6, we show two different decision tress and their rule representation with variables *V1* that has values *C* and *D*, *V2* with values *E* and *G* and target variable *T* which takes variable-values of t and f.

When building a decision tree, a histogram (or distribution) is created at each leaf node. This histogram is populated using the samples that match the variable-values contained in the path (from root to parent of the leaf) of the decision tree. It is then used to assign a probability

(or uncertainty) that a sample has a particular target value given the variables and their values on the path.

A characteristic of decision tree algorithms is the ability to easily transform the decision tree model into a rule model. This is accomplished by traveling every path of the decision tree model going from root to leaf (where leaf is the target node) as seen in Figure 2-6. Most methods for transforming a decision tree into a rule model will choose one rule out of the many that have the same antecedent but different consequents for simplification of the rule model, e.g., either Rule 3 or Rule 4 in Figure 2-6 depending on which rule has more instances correctly predicted which is reflected in the histogram [128].

2.2.3.2 Rule Learner (RL)

Sampling without replacement allows for sample space reduction, and for sparse datasets this can reduce the sample size to a point that any rules discovered might not have enough samples to have confidence in the statistics calculated [16]. One solution to this problem is sampling with replacement, where instead of removing the sample from the dataset once covered by a generated rule, one can mark a flag on the sample to denote that it has been covered. When calculating the number of samples that support a rule (inductive strength), if a sample that matches the antecedent of the rule has been previously covered, it is not included in the calculation, but it is used in the other statistics, such as positive predictive value, which can increase the statistic since previously covered samples are included in the cohort for calculation. Algorithms like MetaDendral [112, 113], and RISE [19] used this technique as well as some more complex learners such as Genetic Algorithms [54]. The MetaDendral algorithm and its descendant rule learning algorithm RL4 [20] have been used for knowledge discovery because of its applicability to both large datasets and small datasets [23].

RL4, and its successor, RL [2, 6], which implements a method for decreasing database look-up for matching rules called breadth first marker propagation [27], have been implemented into a system called Heuristic Autonomous Model Building (HAMB) [17, 130, 131]. HAMB generates and test hypotheses automatically for knowledge acquisition and discovery. It was shown that this system's use of rules allowed for ease of understandability by experts [17, 18, 20, 23, 130, 131].

RL has a specific advantage with respect to rule learning due to its use of the Breadth First Marker Propagation (BFMP) algorithm [27]. This algorithm stores the data in a way that allows for fast and efficient look-up reducing the overall search time when adding conjuncts to the antecedent. To achieve this end, BFMP links a sample to its respective variable-values, and the *variable-value* to those samples that have it. This bi-directional look-up allows a linear increase in processing time as the number of samples increase as detailed by Aronis et. al. [27].

2.2.4 Bayesian networks

Bayesian networks are Directed Acyclic Graphs that capture probabilistic relationships among variables, which can be used to perform probabilistic inference. The expression of uncertainty within these relationships has been proven beneficial in resisting over-fitting to the dataset (increasing generalizability) as well as uncertainty propagation [62, 63, 70, 132, 133]. In the next subsections we discuss how different Bayesian network models are represented and a principle used for calculating the probability of a particular model. We discuss Global Structure, Markov, and Local Structure (a generalization of the Global Structure) models (Sections 2.2.4.1-0). Then we discuss how a Bayesian network model is generated (Section 2.2.4.5). Finally we discuss how inference is performed with a Bayesian network once it has been created (Section 2.2.4.6).

2.2.4.1 Components of a Bayesian network

A Bayesian network uses a directed acyclic graph (DAG) and a parameter representation that represents the joint distribution over a set of random variables [63, 134]. A formal definition of a Bayesian network is a DAG constituting nodes and edges that when combined with parameters, represents the joint distribution over all the variables contained within the network. Specifically, Model B (M_b - a Bayesian network) contains a pair (M_b^G , M_b^Θ) that is a DAG,

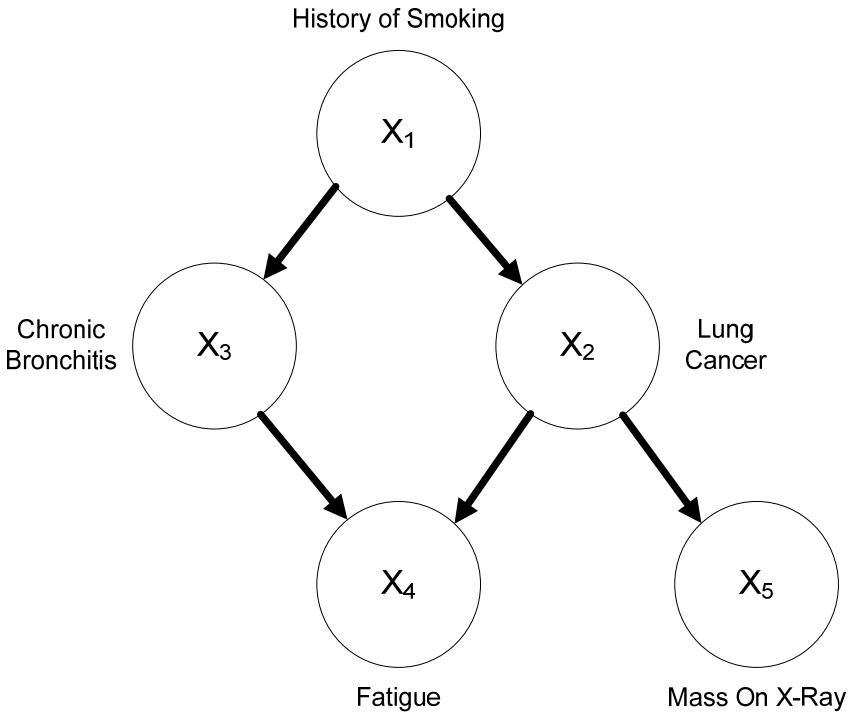


Figure 2-7. A Hypothetical Bayesian Network [63, 124, 132].

which represents all the variables and edges within **X** and a set of parameters that represent the

relationship between connected nodes. There is a node for every variable within **X** and there are edges (arcs) connecting them [63, 70, 124].

Terminology associated with M_b is defined as follows: any originating node that has one directed edge leading into another node is known as a parent ($X_i \rightarrow X_j$, X_i is the parent of node X_j), while any node that is at the terminating end of an edge is known as a child (X_j is the child of X_i); all nodes disseminating from X_i (local – one edge, and remote – more than one) are also known as the successors or descendants of X_i.

As seen in Figure 2-7, Bayesian network (BN), the variable *History of Smoking* is the parent of *Chronic Bronchitis* (*CB*) and *Lung Cancer* (*LC*), while *Fatigue* is the child of both *CB* and *LC*. In terms of descendants, *Fatigue* is a descendant of *History of Smoking* due to a direct path that exists between them (through *CB* or *LC*). A formal definition is: X_j is a descendant of X_i if, using the directionality of the edge, there is a direct path from X_i to X_j.

The BN in Figure 2-7 utilizes independencies and dependencies between nodes to create a DAG. The graph, M_b^G utilizes the Markov Condition, which allows for the reduction of the complex joint distribution into components otherwise known as factorization.

$$P(X_1, X_2, X_3, X_4, X_5) = P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1)P(X_4 \mid X_2, X_3)P(X_5 \mid X_2) \qquad (4)$$

Equation 4 represents a factorization of the example BN in Figure 2-3. A more general form exists as, for any model with variables $\boldsymbol{Y} = \{Y_1, Y_2, Y_3, \ldots, Y_n\}$, the joint probability distribution can be factorized as [133-135]:

$$P(Y_1, Y_2, Y_3, \ldots, Y_n) = \prod_{i=1}^{n} P(Y_i \mid parents(Y_i)) \quad (5)$$

In Equation 5, the function parents(Y_i) refers to the set of nodes which are the parents of Y_i (arcs terminating at the Y_i node). If Y_i has no parents, i.e., the set of parents is null (Ø) and P(Y_i | parents(Y_i)) = P(Y_i). A conditional probability table (CPT) is defined from each specific product in Equation 5, where for each variable Y_i, each row would be a value of Y_i and each column would be each unique state of parents(Y_i) as seen in Table 2-1 for the variable Fatigue from Figure 2-7.

2.2.4.2 D-Separation

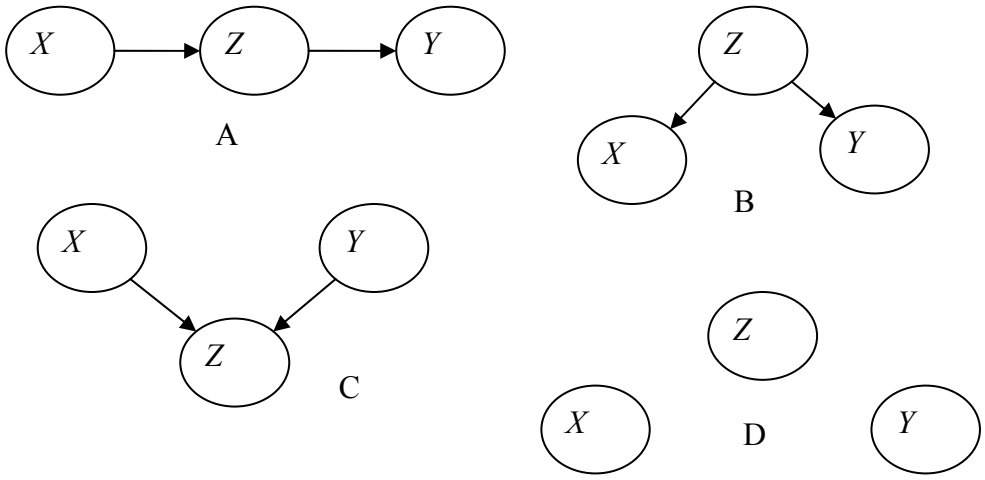


Figure 2-8. Some of the different graphs possible between three nodes *X*, *Y*, and *Z*.

Depending on how the BN is used, sometimes it is necessary to know which nodes (variables) are independent from each other. D-separation helps determine which nodes are dependent and which are independent. It can be defined as a set of rules that are used to determine conditional independence using three structures in a DAG: a collider (Figure 2-8C) or V structure, a non-

collider (Figure 2-8A) or linear structure, a Descendant (Figure 2-8B) where X and Y are descendents of Z, and a unconnected DAG where *Z, X*, and *Y* have no connection and can thus be said to be independent (Figure 2-8D). We can also define D-separation using the Markov condition, which states that a node is conditionally independent from its non-descendents given its parents within a DAG. We will use Figure 2-8 to illustrate some principles of D-separation. Using Figure 2-8A-C, the rules of D-separation are as follows: If we consider all paths from *X* to *Y*, *X* is D-separated from *Y* given *Z* (*X* and *Y* are conditionally independent given *Z*): if and only if (iff) the paths that connect *X* to *Y* meet head to tail at *Z* (Figure 2-8A), iff all paths are blocked by *Z* if the edges meet tail to tail at Z (Figure 2-8B), or iff *Z* and all of *Z*'s descendents are not in any path between *X* and *Y* where edges are head to head at Z (Figure 2-8C). Figure 2-8D shows one case of how *X* and *Y* are always conditionally independent regardless of what path is taken and what nodes are on that path since there are no edges [63]. Use of D-separation allows for the formation of a parsimonious graph representing all the dependencies derived from either experts or from data using the least amount of directed edges.

2.2.4.3 Markov Blanket Model

Using D-separation a Markov blanket is described as follows: given ***X***, where ***X*** is a list of nodes {1,…,n}, all nodes not in ***X*** are D-separated from the target node and therefore do not affect the distribution of the target. Three basic rules can be used to define the Markov blanket for a target node: (1) all parents with edges connecting to the target node, (2) all children with edges deriving from the target node, and (3) all parents with edges connect to children of the target node. After construction of a global BN, isolating the Markov blanket of any node is easily accomplished using these rules. For example, in the BN in Figure 2-7, using the above three rules, the Markov blanket of Fatigue would consist of Chronic Bronchitis and Lung Cancer with

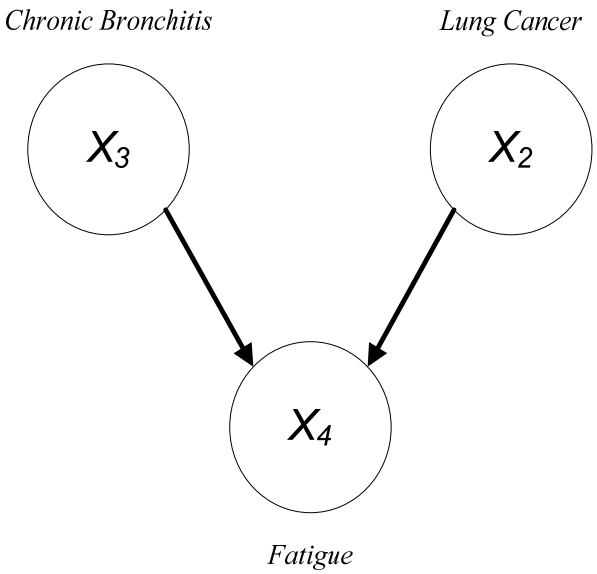


Figure 2-9. A Markov-Blanket with target node *Fatigue* derived from Figure 2-7's Bayesian Network.

their edges connecting to *Fatigue* (the parents of the target) with no other edges or nodes being a part of the blanket as shown in Figure 2-9.

Table 2-1. An example of a conditional probability table of the *Fatigue* node from Figure 2-9. *LC* is *Lung Cancer* and *CB* is *Chronic Bronchitis*, and *F* is *Fatigue*

	LC = Yes *CB* = Yes	*LC* = No *CB* = Yes	*LC* = Yes *CB* = No	*LC* = No *CB* = No
F = Yes	P(*F*=Yes|*LC*=Yes&*CB*=Yes)	P(*F*=Yes|*LC*=No&*CB*=Yes)	P(*F*=Yes|*LC*=Yes&*CB*=No)	P(*F*=Yes|*LC*=No&*CB*=No)
F = No	P(*F*=No|*LC*=Yes&*CB*=Yes)	P(*F*=No|*LC*=No&*CB*=Yes)	P(*F*=No|*LC*=Yes&*CB*=No)	P(*F*=No|*LC*=No&*CB*=No)

2.2.4.4 Bayesian Local Structure

When constructing a BN, most algorithms focus on the *global structure* of the network, which focuses on all combinatoric values when building the conditional probability table. Due to the requirement of global structures to parameterize all combinatoric values of all the parents for the CPT, dependencies/independencies between the variable-values of the parents of a node and the child node are not considered. A group of algorithms model this *local structure* capturing the relationships of the values of X_i in relation to the values of the parents(X_i). Local structure can capture additional independencies within the variable-values reducing the number of parameters required by M_b^{Θ} to properly calculate the joint distribution. Since local structure allows for

context-specific (variable-value) independencies, it can be viewed as a more general form of the global structure [65, 66].

If we use the example BN above in Figure 2-7 and specifically focus on the parents of *Fatigue*, which are *CB* and *LC*, we can demonstrate the previously mentioned characteristics. First, we have to define the possible values for each parent variable of *Fatigue*. Let us assume *CB* and *LC* are binary variables with possible values of *yes* or *no*. Let us also assume that only some combinations of values of *CB* and *LC* (e.g., *CB* = yes and *LC* = no) affect the distribution of *Fatigue,* while if *LC* = yes, the distribution of Fatigue is the same regardless of the values of *CB*.

This example shows how utilizing the independencies of the variable values one can reduce the parameterization (decrease the complexity) from four parameters to three parameters due to the relaxation of the requirement of all combinatoric variable-values be explicitly defined within the conditional probability table (Figure 2-10, Figure 2-11 and Table 2-2).

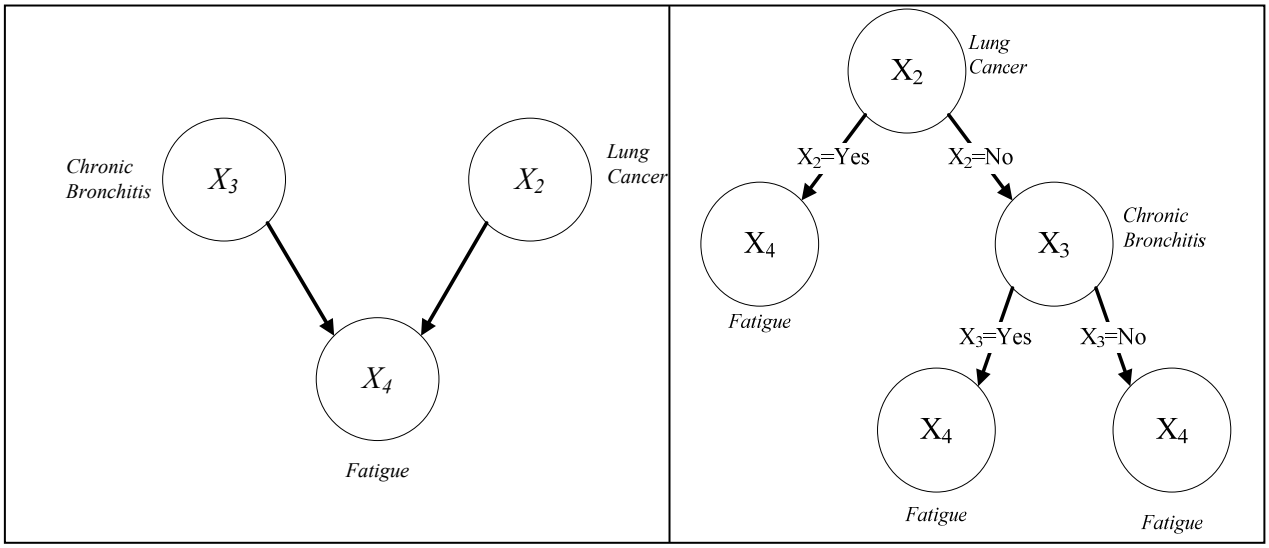


Figure 2-10. Global and Local Structure Representation of part of a Bayesian Network. This Figure represents a global structure of *Fatigue* and its parents(left) and one possible local structure (right) of *Fatigue*. This representation shows how once *Lung Cancer* = yes, then the distribution remains the same for all values of *Chronic Bronchitis*.

Table 2-2. The parameters of the conditional probability table associated with the local structure for Figure 2-9, Figure 2-10, and Figure 2-11.

	Lung Cancer = no		*Lung Cancer* = yes	
	Chronic Bronchitis = no	*Chronic Bronchitis* = yes	*Chronic Bronchitis* = no	*Chronic Bronchitis* = yes
Fatigue = yes	θ_1	θ_2	θ_3	θ_4
Fatigue = no	$1 - \theta_1$	$1 - \theta_2$	$1 - \theta_3$	$1 - \theta_4$

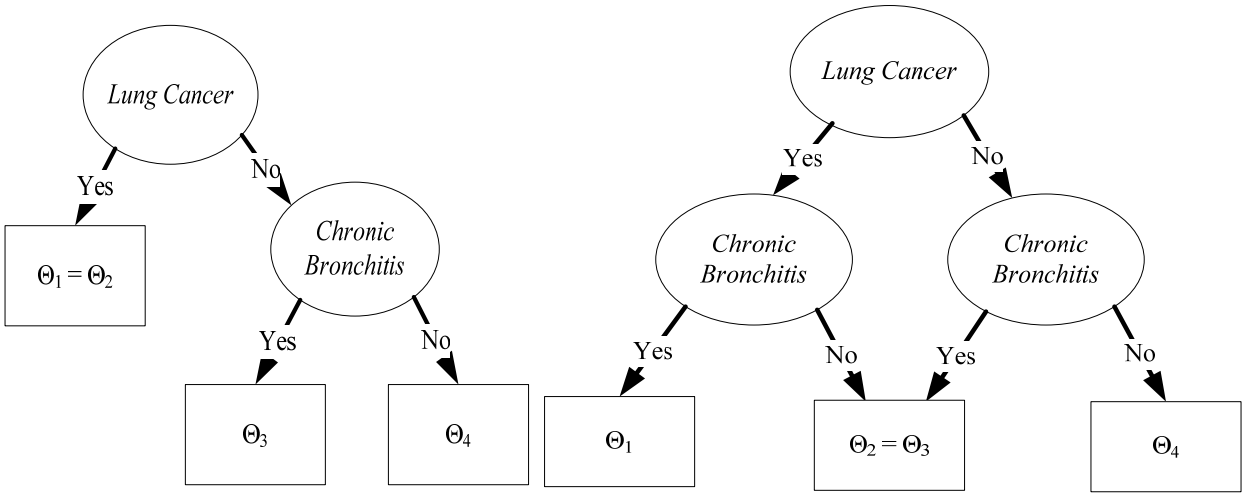


Figure 2-11. Two Bayesian Local Structures. Examples of a Decision Tree (Left) and a Decision Graph (Right). Here both the Decision Tree and Decision Graph can represent the same constraints. As shown above, Decision Tree cannot always represent the same constraints as the Decision Graph

While a local structure algorithm defines the interactions between the parents of a node and the node's distribution, one can generate a global structure directly from the local structure of each node. That is, if a variable appears in the local structure, then in the global structure that variable is a parent and has an edge leading into the node while maintaining the distribution within the CPT . In Figure 2-10 on the Right, we show one single representation of the local structure of the BN in Figure 2-9.

There are multiple algorithms for generating this local structure. Two popular algorithms are the Bayesian Decision Tree algorithm [68] and the Bayesian Decision Graph algorithm [67]. In the Bayesian Decision Tree algorithm, each branch denotes variable-values of each parent and each leaf is the distribution of the predictor variable given all the values of the parents on the

path from root to leaf in Figure 2-11 (Left), while for the Bayesian Decision Graph algorithm, a node, except for the root node, can have multiple parents as seen in Figure 2-11 (Right) where each circle denotes a variable, each arc denotes a specific variable-value and each square denotes the final distribution of the target node along that path. There are some advantages that a decision graph has that will be explained later. From Figure 2-10, if we were looking at the rightmost path, the distribution for X_4 would be the distribution of that variable given X_2 = no and X_3 = no, P(X_4 | X_2 = no, X_3 = no). In Figure 2-11 , we show a representation of a possible decision tree and decision graph with their respective parameterization in Table 2-2.

All equalities (constraints) represented by a Decision Tree can be represented by a Decision Graph, but the converse is not true. In Figure 2-11 (Right), we show an example of how a Decision Graph can be used to represent the previously mentioned equality that a Decision Tree is incapable of representing. So using the equality stated before: P(*Fatigue* = yes | LC= yes, CB = no) = P(*Fatigue* = yes | LC = no, CB = yes). The advantage of a larger space of possible local structures can make Decision Graphs an attractive method for modeling Bayesian local structure as expressed in Chickering, et al[67].

As described above, the number of possible models (model space) is much richer when utilizing Bayesian local structure with the added advantage of possibly fewer parameters needed. This richer space involves how a variable is "split" or branched when building the local structure as seen in Figure 2-10 and Figure 2-11. There are two methods of splitting that are used, binary and complete for variable-values. For a binary split, given a variable V has three or more values depicted as V = {A,B,C}, two branches are created such as one branch for V=A and a second branch for V=Not A = {B,C}. This allows the grouping of variable-values, in this case B and C, which individually might not have large discriminative power compared to that of value A. A

complete split, in the case of the variable V, has three branches with each branch representing a unique variable-value. For global BNs, every variable is added using a complete split. This reduction in complexity (number of parameters) within the conditional probability table makes calculation of the parameters easier and has in fact been shown to increase the performance and accuracy of the network from simulated data over traditional global structure [68]. Recently Visweswaran et al. [136] utilized Bayesian local structure to improve prediction using local structure vs. global structure for a patient-specific model generation algorithm.

Uncertainty in BNs, whether in *global* or *local structures*, is encapsulated by the parameterization of the model, M_b^{Θ}. Multiple methods exist for learning the parameterization including K2 [70] and BDeu [62], which can learn the structure while learning the parameters. Each method, K2 and BDeu, uses different priors to build and parameterize the model M_b to maximize the join probability P(M_b, Data). The result from either method is a BN that has conditional probability tables with each cell's θ calculated by counting the number of samples that have a specific variable value and adding it to the prior counts as reflected by the Dirichlet priors. Often when calculating the posterior probability of a target value given a set of observed variable-values, P(Y=t|X_1=a,X_2=b,...), the probability of the model and the data are ignored by assuming that their probability is uniform. This can often result in a calibration issue, (e.g., probabilities over target values do not sum to 1) which is often resolved by normalizing the probabilities such that the target value probabilities sum to 1 [137].

2.2.4.5 Model Generation using the K2 algorithm

Many BN generation algorithms, both global and local, try to maximize the probability of the model in a greedy fashion. Many structure learning algorithms, such as K2, which combines both

structure and parameter learning together, maximize the probability of the model and the parameters concurrently using a closed form solution for the join of the model and the data or P(B_s,D) where the model $B_s = (M_b^G, M_b^\Theta)$ and the data is D. K2 Scoring is a greedy search algorithm that maximizes the likelihood by adding directed edges between nodes [70]. It has the following assumptions: 1) all variables in dataset D are discrete, 2) given a Belief Model (Bayesian network), each case or instance in D occurs independently, 3) there are no missing values in dataset D, and 4) the density function for each distribution in B_s is uniformly distributed, that is parameterization of the distribution within each node prior to the addition of data are all equally likely apriori. Cooper and Herskovits then showed how to relax the 4th assumption by using Dirichlet priors in lieu of the uniform priors on the parameterization within the CPT for the nodes within B_s. P(B_s, D) is used as an approximation to the posterior probability given that all models are uniformly probable: P(B_i) = P(B_j) for all $i \neq j$. Therefore P(B_s, D) = P(D |B_s)*P(B_s) ≈ P(D|B_s). The equation used to calculate the likelihood is seen in:

$$P(Bs, D) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

$$\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}, \; N_{ij} = \sum_{k=1}^{r_i} N_{ijk} \qquad (6)$$

where $\Gamma(\bullet)$ is the gamma function, n represents all the nodes in the BN and i is a specific node within the network, q_i is the number of possible parent states of node n_i and j is a specific state of the set of possible parent states, r_i is the number of states or values of node n_i and k is a specific value of node n_i, and N_{ijk} is the number of instances that match state j of the parents of node i and k is the specific node value of node i. The α_{ijk} are the hyperparameters of the Dirichlet which

define the prior probability. Equation 7 below represents the reduced from of Equation 6 when the α_{ijk} is set the uninformative value of 1.

$$P(B_s, D) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{7}$$

The running time for this algorithm is $O(mu^2n^2r)$ where r is the maximum number of states any variable I can take on, u is the maximum number of parents per a variable, m is the number of instances within dataset D, and n is the number of variables in dataset d.

The algorithm for building a BN using K2 algorithm uses the node order as a guide and only allows nodes that preceded it to be the parents of that node. That is, the parents of node i can only be a subset of nodes 1 and i-1 in the order specified. This algorithm is a greedy hill climbing search, maximizing the likelihood $P(D|B_s)$, as well as using a user specified parameter u, which represents the maximum number of parents. Thus, if the node i has u parents, then no more nodes are considered to be added as a parent of node i. Ultimately, what is generated is a list of nodes with each node having a list of parents. This can easily be transformed into the representation seen in Figure 2-7. If $u = n$, where n is the total number of variables, the time complexity is $O(mn^4r)$.

While the original implementation of K2 requires an variable order to help guide the structure building phase, many algorithms utilize different heuristics such as randomization and restarts to help alleviate the order dependencies [63, 138]. The next section describes inference using a model derived from a BN structure search algorithm such as K2.

2.2.4.6 Inference using Bayesian networks

For inference using BNs, the result is a distribution over the target variable given a set of observed variable-values. Inference using BNs can be performed in multiple ways, but since performing exact inference in a DAG is NP-Hard [139], the algorithms mentioned will all be approximate. The Message-Passing Algorithm (MPA) uses local independencies to propagate uncertainty when performing approximate inference for singly connected (each node has only one parent) DAGs [134], and it has been extended through the use of clustering based on d-separation to multiply connected DAGs (where nodes can have more than one parent). An alternative to MPA is the Junction Tree algorithm which also passes messages, but through derived junction trees instead of the original DAG [140]. Another approximate solution is the use of symbolic probabilistic inference, which uses factoring (reduction of the joint to reflect the conditional independencies within the DAG) [141]. Once the posterior distribution is calculated for the target variable, a threshold is generally used for deciding what to target value to assign given that the prediction is actually a probability and not discrete. For binary targets, the threshold for deciding the assigned target value given a probability without an Receiver Operator Curve (ROC) analysis [142] is usually set to 0.5, which is similar to majority voting. For a multi-valued target variable, a majority vote is usually the preferred method since calculation of an ROC curve can be difficult especially in multi-class problems.

2.2.5 Hybrid Rule Learning

As mentioned earlier, the usefulness of rules is apparent in the understandability by experts. This has led researchers to try to generate rules from hybrid algorithms that combine two or more different algorithms together, thus combining the benefits of understandability and modularity of

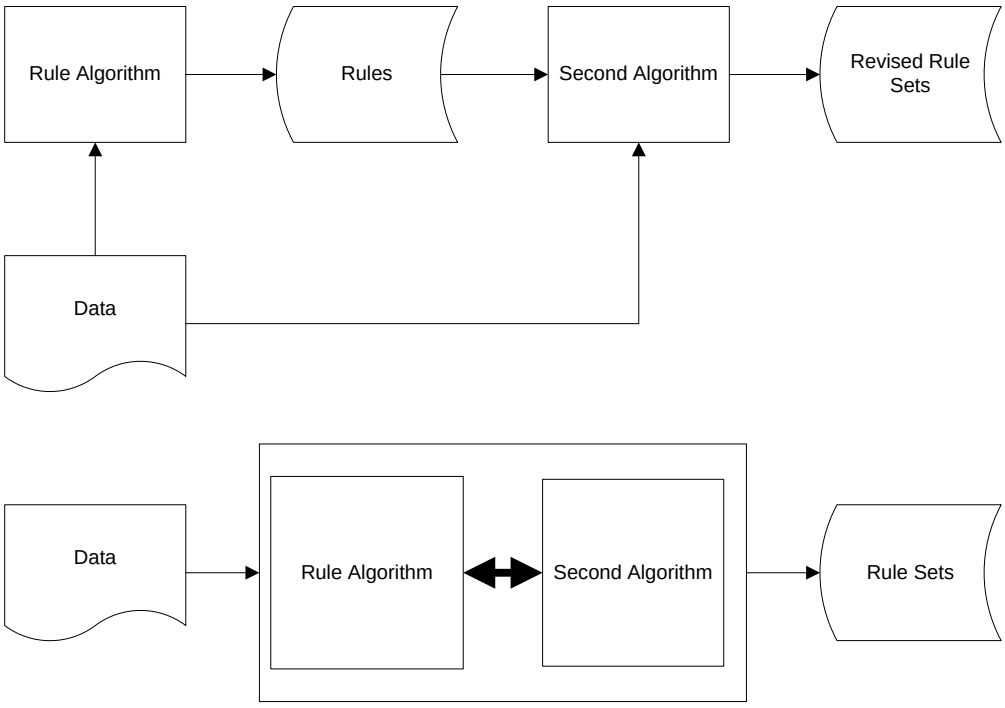


Figure 2-12. A general characterization of a rule revision algorithm (top) and a hybrid rule algorithm (bottom). On the top, there is a re-evaluation of the data by the second algorithm because this second algorithm takes both generated rules and data as arguments.

rules with the benefits of other algorithms by using them to address the 4Rs (Section 1.1) [16, 26, 56, 143].

Knowledge Based Artificial Neural Networks (KBANN) [40, 144] and RAPTURE [64] are both examples of algorithms use an additional algorithm to revise a prior rule set (Figure 2-12 top). This methodology however requires a prior rule set and thus is still constrained by the algorithm that was used to generate the seed rule model. An alternative method of rule generation uses hybrid rule learners that try to combine the strengths of multiple algorithms to generate rules (Figure 2-12 bottom). As we see, the primary algorithm and the secondary algorithm interact to produce a rule set. Two recent examples described below utilize Genetic Algorithms and BNs to generate the rule sets.

2.2.5.1 Hybrid Decision Tree/Genetic Algorithm

One research group has combined Genetic Algorithms and Decision Trees to produce a rule set that performs better on high-dimensional datasets [16]. They utilize the strength of decision trees to define rules (as seen in Figure 2-6) allowing decision tree growth, as explained in Section 2.2.3.1, until the coverage of the leaf node is below a threshold representing the minimum number of samples (e.g., in Figure 2-13 the threshold is 16 samples). Once this threshold is reached, those instances along the path that fell below the threshold are condensed into a new dataset. This dataset is then analyzed by a Genetic Algorithm, which treats variable-values as genes on a chromosome (the set of variable-values on a single chromosome can be considered a single antecedent) and then simulates cross-over (switching one variable-value for another) and

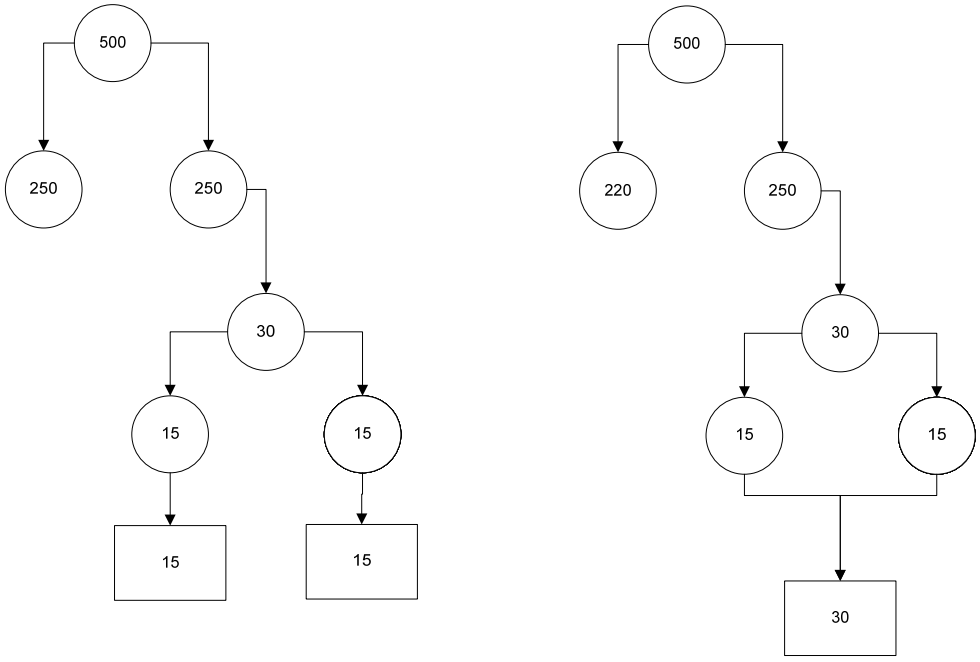


Figure 2-13. Two different DT/GA implementations with GA-SMALL being on the left and GA-large-SN on the right. Circles represent a decision node and squares represent a run of the genetic algorithm. The rules generated from the Decision Tree and the Genetic Algorithms are combined to create a new rule set. The numbers in the circles represent the number of samples that are covered by that specific path, and the numbers in the square represents the dataset size used for the GA.

measures how fit the new chromosome is where fitness is a statistical measure and can be something as simple as accuracy. It is usually run for a pre-defined number of simulations. A rule model is then returned as those chromosomes are converted into rules. The rules from the decision tree are then simply merged with the rules generated by the Genetic Algorithm to create a single rule model. Carvalho, et al. showed that the rule set derived from both the decision tree and genetic algorithms was more optimal (had a higher performance) than rules derived solely from a decision tree algorithm. While their results are impressive, it has been shown that conquering without separating (sampling with replacement described in Section 2.2.3.2) produces a superior rule set [19, 121]. Another recent hybrid rule learner discussed next combines BNs with Rule Generation to produce hybrid rules that have the certainty measured by the BN.

2.2.5.2 Markov Blanket Bayesian network Hybrid Rule Learner

Hruschka, et al. used the K2 to generate the Markov blanket model (see Section 2.2.4.5), restricting the model space to connections that are parents, children, or parents of the children of the target variable [56]. This provides a definitive speed up over standard BNs due to the consideration of a small subset of network structures. Once learned through Model Selection, this model is converted into rules by transforming the blanket into another BN where all of the variables within the blanket are made into the parents of the target; this is done regardless of whether the variable was a parent, child, or parent of the child of the target variable.

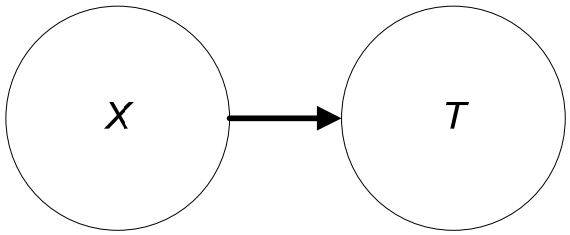
Bayesian Model: M_b	Rule Model: M_r
 P(*X*=x1) = .7 P(*X*=x2) = .3 P(*T* = t | *X* = x1) = .8 P(*T* = f | *X* = x1) = .2 P(*T* = t | *X* = x2) = .4 P(*T* = f | *X* = x2) = .6	Rule 1: **IF** *X* = x1 **THEN** *T* = t CF = 4, P(*X* = x1) = .7 Rule 2: **IF** *X* = x1 **THEN** *T* = f CF = .25, P(*X*=x1) = .7 Rule 3: **IF** *X* = x2 **THEN** *T* = t CF = .667, P(*X*=x2) = .3 Rule 4: **IF** *X* = x2 **THEN** *T* = f CF = 1.5, P(*X*=x2) = .3

Figure 2-14. Conversion of a Bayesian Network Model (left) to a Rule Model (right) where the certainty factor is expressed as the likelihood ratio of the conditional probability of the target given the variable value. As seen in Rule 1, the CF is the conditional probability of P(*T*=t | *X*=x1)/(1- P(*T*=t | *X*=x1)) = .8/.2 = 4. Within the rule there is also the probability of observing that variable value, e.g., P(*X* = x1) = .7.

A simple example is shown in Figure 2-14. If we were to introduce additional variables as parents of variable *T*, the number of rules would grow exponentially. To calculate the certainty factor for a rule, Hruschka, et al. set the Markov blanket to the variable-values within the antecedent and then calculated the resulting distribution on the target variable. They showed that their method for generating rules, in all but a few cases, generated a substantially smaller subset of rules that had equal to or greater predictive power than those generated by simple rule learners or Decision Trees.

Using the Markov blanket and transforming the network into one where all the variables in the Markov blanket are parents of the target results in a structure that is not optimized for a single CPT where all the variables are the parents of the target, which is what is used for rule generation. While they showed that you could extract rules using their network transformation

method, the certainty factor value is incorrect due to the distribution on the target variable. Since there is no such transformation from a Markov blanket to a model that only has parents of the target that maintains the independencies within a collision structure (e.g., the target, a child of the target, and a parent of the child), one cannot generate an equivalent probability BN with all of the nodes as parents of the target node. Thus the structure used to generate rules is fundamentally different than the Markov blanket structure. Hruschka, et al. even acknowledged that there might also be a semantic component to rules which is not captured by their method of rule generation due to the use of a Markov blanket.

3.0 BAYESIAN RULE GENERATION FRAMEWORK

This chapter describes the Bayesian Rule Generation Framework (BRGF) and the different methods currently implemented for generating probabilistic rules from BNs learned from biomedical data. In Figure 3-1, we show the BRGF (Figure 3-1 Right) next to the current non-probabilistic rule generation method (Figure 3-1 Left). Specifically we show how with the BRGF, the user selects a specific type of Bayesian Structure and then within the framework, they can select specific search strategies to generate probabilistic rules. Section 3.1 explains the various aspects of the BRGF including all the algorithm pseudo code, while Section 3.2 analyzes the computational complexity of the algorithms within the BRGF. The methods for evaluating the BRGF including the testing of the BRGF on simulated data as well as the performance measures used are detailed in Section 3.3.

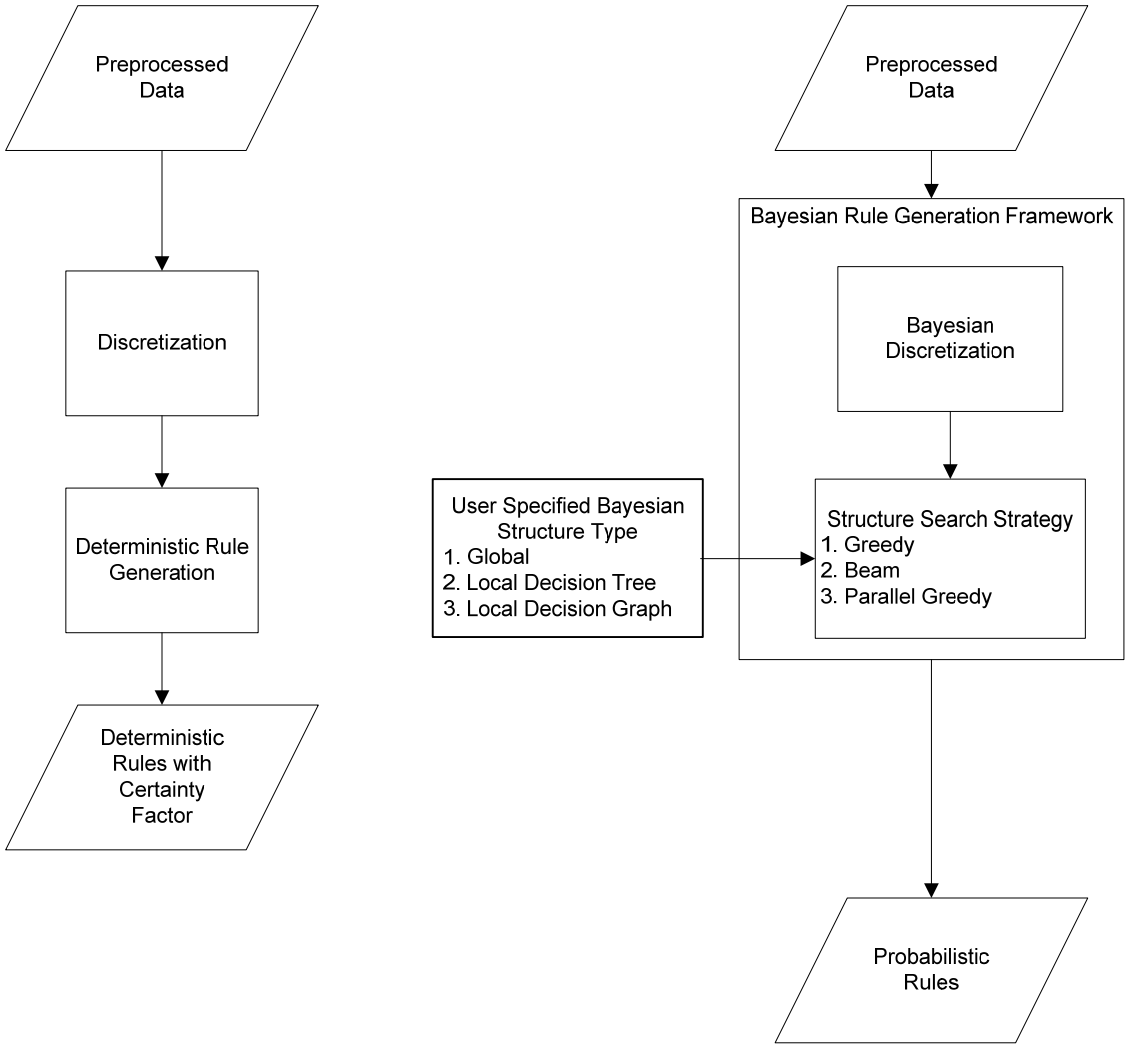


Figure 3-1. The Bayesian Rule Generation Framework. On the left is the traditional model of rule generation and on the right is the new framework.

3.1 DESCRIPTION OF BRGF

The BRGF is a framework established for ease of interchangeability of rule generation methods, combining both probabilistic and deterministic search strategies with probabilistic discretization methods. In Figure 3-1, we see a linear framework that allows modularity as well as ease of interpretation and addition for future search techniques/rule generation methods. There are three

important components to the BRGF: a) the choice of Bayesian discretization, b) the choice of Bayesian structure, and c) the choice of the search strategy. These will be explained in Sections 3.1.1 through 3.1.4.

3.1.1 Discretization

Since all the algorithms that I have used to test my hypotheses require discrete data, it is imperative that I test different discretization methods in an attempt to identify which technique produces the highest performance. I have implemented three discretization techniques in Java: a) Fayyad and Irani's MDLPC [49] as described in Section 2.2.2 and b) a new Bayesian technique called Efficient Bayesian Discretization (EBD). All have been integrated to be used with the Naïve Bayes implementation within the Waikato Environment for Knowledge Acquisition (WEKA) version 3.5.6 [127].

3.1.1.1 Efficient Bayesian Discretization (EBD)

The Efficient Bayesian Discretization (EBD) algorithm uses a Bayesian score to evaluate a discretization policy. Assume that we have a dataset of n instances that form a vector of values for a continuous variable X and target variable Z in the form of (X,Z) pairs, the data might be as follows: <(1.2,T), (3.2,F), (2.3,T), (4.3,F)> for an n of 4. Given this dataset of n instances the string S can be constructed from the target values where S = "TTFF" when the vector is sorted for continuous variable X. EBD scores all possible discretizations up to I intervals (all combinations of splitting up string S into I substrings or intervals) and selects the one with the highest Bayesian score. We now derive the Bayesian score used by EBD. The posterior probability $P(M \mid S)$ of a discretization M is given by Bayes rule as follows:

$$P(M \mid S) = \frac{P(M) \cdot P(S \mid M)}{P(S)} \tag{8}$$

where $P(M)$ is the prior probability of discretization M, $P(S \mid M)$ is the marginal likelihood of the data S given discretization M, and $P(S)$ is the probability of the data. Since $P(S)$ is the same for all discretizations, the Bayesian score evaluates only the numerator on the right hand side of Equation 8, as follows:

$$score_{EBD} = P(M) \cdot P(S \mid M) \tag{9}$$

The marginal likelihood in Equation 9 can be derived using the following equation:

$$P(S \mid M) = \int P(S \mid M, \theta_M) \cdot P(\theta_M \mid M) d\theta_M \tag{10}$$

where θ_M are the parameters of the conditional distribution of the target variable Z given the continuous variable X, namely, $P(Z \mid X)$. Equation 9 has a closed-form solution under the following assumptions: (1) the values of the target variable were generated according to iid sampling from $P(Z \mid X)$, which is modeled as a multinomial distribution, (2) prior belief about the distribution $P(Z \mid X = x_i)$ is independent of prior belief about the distribution $P(Z \mid X = x_j)$ for all values x_i and x_j of X, such that $i \neq j$, and (3) for all values x_j of X, prior belief about the distribution $P(Z \mid X = x)$ is modeled using a Dirichlet distribution with hyperparameters α_i and α_{ij}. The closed-form solution to the conditional marginal likelihood is given by simplifying equation 10 to the following expression [69, 70]:

$$P(S \mid M) = \prod_{i=1}^{I} \left(\frac{\Gamma(\alpha_i)}{\Gamma(n_i + \alpha_i)} \prod_{j=1}^{J} \frac{\Gamma(n_{ij} + \alpha_{ij})}{\Gamma(\alpha_{ij})} \right) \tag{11}$$

where, $\Gamma(\bullet)$ is the gamma function, n_i is the number of instances in the interval i, n_{ij} is the number of instances in the interval i that have target-value j, α_{ij} are the hyperparameters in a

Dirichlet distribution which define the prior probability over the θ_M parameters, and $\alpha_i = \sum_j \alpha_{ij}$. I refers to all the cardinality of the unique parent states for the target variable, and J refers to all possible states of the target (child) variable. The hyperparameters can be viewed as prior counts, as for example from a previous (or a hypothetical) dataset of instances in the interval *i* that belong to the value *j*.

The prior probability *P*(*M*) is modeled as a product of two terms:

$$P(M) = \frac{1}{n} \cdot \left[\binom{n+I-1}{I-1} \right]^{-1} \tag{12}$$

where the first term is the prior probability of observing *I* intervals under the assumption that *I* is uniformly distributed between 1 and *n* where n is the number of samples currently seen. The second term is the prior probability of observing a specific discretization given that it contains *I* intervals and under the assumption that all possible *I*-interval discretizations are equiprobable. Substituting Equation 11 and 12 into Equation 9, the EBD score is defined as:

$$score_{EBD} = \frac{1}{n} \cdot \left[\binom{n+I-1}{I-1} \right]^{-1} \cdot \prod_{i=1}^{I} \left[\frac{\Gamma(\alpha_i)}{\Gamma(\alpha_i + n_i)} \prod_{j=1}^{J} \frac{\Gamma(\alpha_{ij} + n_{ij})}{\Gamma(\alpha_{ij})} \right] \tag{13}$$

where the symbols have the same meaning as in Equation 6. When α_{ij} and n_{ij} are positive integers, the gamma function can be expressed in factorial form and Equation 10 can be written as:

$$score_{EBD} = \frac{1}{n} \cdot \left[\binom{n+I-1}{I-1} \right]^{-1} \cdot \prod_{i=1}^{I} \left[\frac{(\alpha_i - 1)!}{(\alpha_i + n_i - 1)!} \prod_{j=1}^{J} \frac{(\alpha_{ij} + n_{ij} - 1)!}{(\alpha_{ij} - 1)!} \right] \tag{14}$$

We maximize the score of Equation 14 by maximizing the conditional marginal likelihood since for efficiency. The meaning of setting all the hyperparameters α_{ij} to 1 is that we

believe *a priori* that every distribution $P(Z \mid X = x)$ is equally likely; thus, this prior is sometimes said to be non-informative.

Let $S^{i,j}$ be the subsequence of S consisting of the instances from i to j. $S^{1,n} = S$.
Let $Disc(S^{i,j})$ be the discretization of $S^{i,j}$ with up to $length(S^{i,j})$ intervals.

$Disc(S^{1,1}) = \{S^{1,1}\}$
For $1 \le j \le n$
  $Disc(S^{1,j})$:=
    For $1 \le i < j$ return the best scoring discretization from
      ARGMAX($Disc(S^{1,i}) \cup \{S^{i+1,j}\}, \{S^{1,j}\}$)

Figure 3-2. Pseudocode for the EBD algorithm.

The pseudocode for the EBD algorithm is given in Figure 3-2 and the recurrence relation used by EBD to compute the Bayesian score at any given stage is shown in Equation 14.

This algorithm has a time complexity of $O(n^2)$ as implied by the two nested *for loops*. EBD eliminates the consideration of empty intervals as well as using sub-problem solutions instead of considering all possible discretizations. For example, if EBD has solved the sub problem that samples 1 – 4 should be in a single interval, this means that SCORE($S_{1\text{-}4}$) > than any combination of up to 4 intervals splitting string $S_{1\text{-}4}$ into smaller substrings. Any future discretizations that contain the interval of $S_{1\text{-}4}$, e.g., $S_{1\text{-}4}$, $S_{5\text{-}8}$ which represents a discretization policy that splits variable X between samples 4 and 5, will never consider any subintervals of string $S_{1\text{-}4}$. This is because EBD has already considered those subintervals and that their score will always be lower than the score of string $S_{1\text{-}4}$.

3.1.2 Bayesian Global Structure

As described in Section 2.2.4, the algorithm depicted in Figure 3-3, maximizes the conditional marginal likelihood which is calculated using the K2 score by assuming *i* is always equal to the target variable:

$$P(B_s, D) = \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

$$= \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!, \quad \alpha_{ijk} = 1 \tag{15}$$

where the variables are the same as in Equation 11. This differs from the algorithm developed in [70] in that there is no explicit variable order required and that it tries every possible variable as an additional parent of the target node (Line 4). Also, we only accept the addition if the score

Input: A list of variables V = {v_1, v_2, v_3, v_m} where v_i is variable i
Variable T which is the target variable
Output: A Bayesian Global Structure M
O = Set of operators Add(M,v_i) which adds vi as a parent of T in model M
Var(M) = All vi that are in model M

1 M'={T}
2 DO
3 M = M'
4 ΔScore = $\max_{v_i \in V - Var(M)} Score(Add(M, v_i))$ - Score(M)
5 IF ΔScore > 0
6 M' = $\arg\max_{v_i \in V - Var(M)} Score(Add(M, v_i))$
7 END IF
8 WHILE (ΔScore > 0)
9 Return M'

Figure 3-3. A pseudocode representation of Greedy Constrained Bayesian Global Structure search.

change is greater than 0 (Line 5 and 6 of Figure 3-3).

3.1.3 Bayesian Local Structure

Bayesian Local structure can be an efficient representation of the conditional probability table allowing a generalization over global structure which requires a combinatorial combination of all variable-values. However, construction of an optimal local structure requires exhaustive search, which is nearly intractable due to the size of the search space involved. Greedy Search, as was used by [67, 124, 145], is a tractable search method over possible local structures, but the gathering of statistics to parameterize the structure is a time consuming procedure. To speed up this process, the BRGF utilizes the Breadth First Marker Propagation (BFMP) methodology [27] to reduce look up time for the parameterization (see Figure 3-5). This critical component allowed the different search paradigms to be explored that are in Section 3.1.4 and the description of how it was used are provided in Sections 3.1.3.1 and 3.1.3.2.

3.1.3.1 Bayesian Local Structure - Decision Tree (BLS-DT)

We adapted the method developed by Friedman et al. which can be used for developing an entire global network based on local structure [68]. We constrained our model, as seen in Figure 2-10, to only those models with variables being a direct parent of the target variable so some of the algorithmic complexities associated with global model construction using local structure can be removed. Using breadth-first marker propagation for this algorithm also provides significant speed up since database look-up is an expensive operation [27].

Input: A list of variables V = {v_1, v_2, v_3, v_m} where v_i is variable i
Variable T which is the target variable
Output: A Bayesian Local Structure M
L = All leaves of Model M where L_j is leaf j
O = Set of operators BinarySplit(M, L_j,v_i) and CompleteSplit(M, L_j,v_i)
Var(L_j) = All v_i that are on path from L_j to root

1 M'={T}
2 DO
3 M = M'
4 ΔScore = $\max_{l \in L, v_i \in V - Var(l),\, o \in O} Score(o(M, l, v_i))$ - Score(M)
5 IF ΔScore > 0
6 M' = $\arg\max_{l \in L, v_i \in V - Var(l),\, o \in O} Score(o(M, l, v_i))$
7 END IF
8 WHILE (ΔScore > 0)
9 Return M'

Figure 3-4. The pseudocode for a Bayesian Local Structure Decision Tree.

Figure 3-4 is the algorithm used for greedy search when building a Bayesian Local Structure Decision Tree. As is detailed in the algorithm, the search method is only concerned with finding those parents of the target variable (Line 4), which should result in a large speedup as compared to the general model detailed in [68, 124] since the algorithm is only searching over a constrained global structure. Similar to the pseudocode in Figure 3-3, we accept an additional node into the model if and only if the score increases (Lines 5 and 6). This eliminates equivalent models, which is acceptable given the greedy nature of our search strategy. We also used the heuristic of maximum number of parents to prevent overspecialization as well as to reduce the running time (default was set to 10 variables per path). An advantage of using BFMP is that with each split the samples that match that variable-value are the only ones that are tracked. Figure 3-5 (A-C) shows how the samples are tracked as the nodes are split. Once a node is split, only those samples that are on the path need to be considered when calculating the score.

The Bayesian local structure decision tree is a generalized model of the global structure previously discussed in Section 3.1.2 since it relaxes the constraint of combinatorial combination of all variable-values of the parents of the target node. It is possible, using the decision tree local structure, to achieve the same model (same number of parameters and parameter values within the CPT of the target variable) as those achieved by the original global structure algorithm by using the complete split operator.

3.1.3.2 Bayesian Local Structure – Decision Graph (BLS-DG)

The decision tree, while beneficial as seen in [68], does not allow for the correction of specialization (splitting on a particular variable). Specifically, as seen in [67, 124] and in Figure 2-11 (Right), a decision graph allows combination of leaves joining together distributions that are similar.

On line 3 of Figure 3-5, the function finds the maximum scoring model by either adding a variable to a leaf (using complete split or binary split), or merging any two leaves. This involves multiple internal iterations over leaves, variables, and operations and thus is computationally complex.

Comparing Figure 3-4 to Figure 3-6, we see the only real modification is at line 3 and line 6 where we add the function of combining leaves (columns in the CPT). This is a place where marker propagation can have a significant speed impact [27]. When merging two separate paths, the samples have to be tracked from both paths. Using marker propagation, the samples that are both leaves can be combined for tracking, thus streamlining counting and further specialization. The next figure, Figure 3-5, shows how when combining leaf nodes (Figures C and D in Figure 3-6), the samples are added to the same nodes, Future specialization can now only impact those samples that have variable-values that match either of the paths.

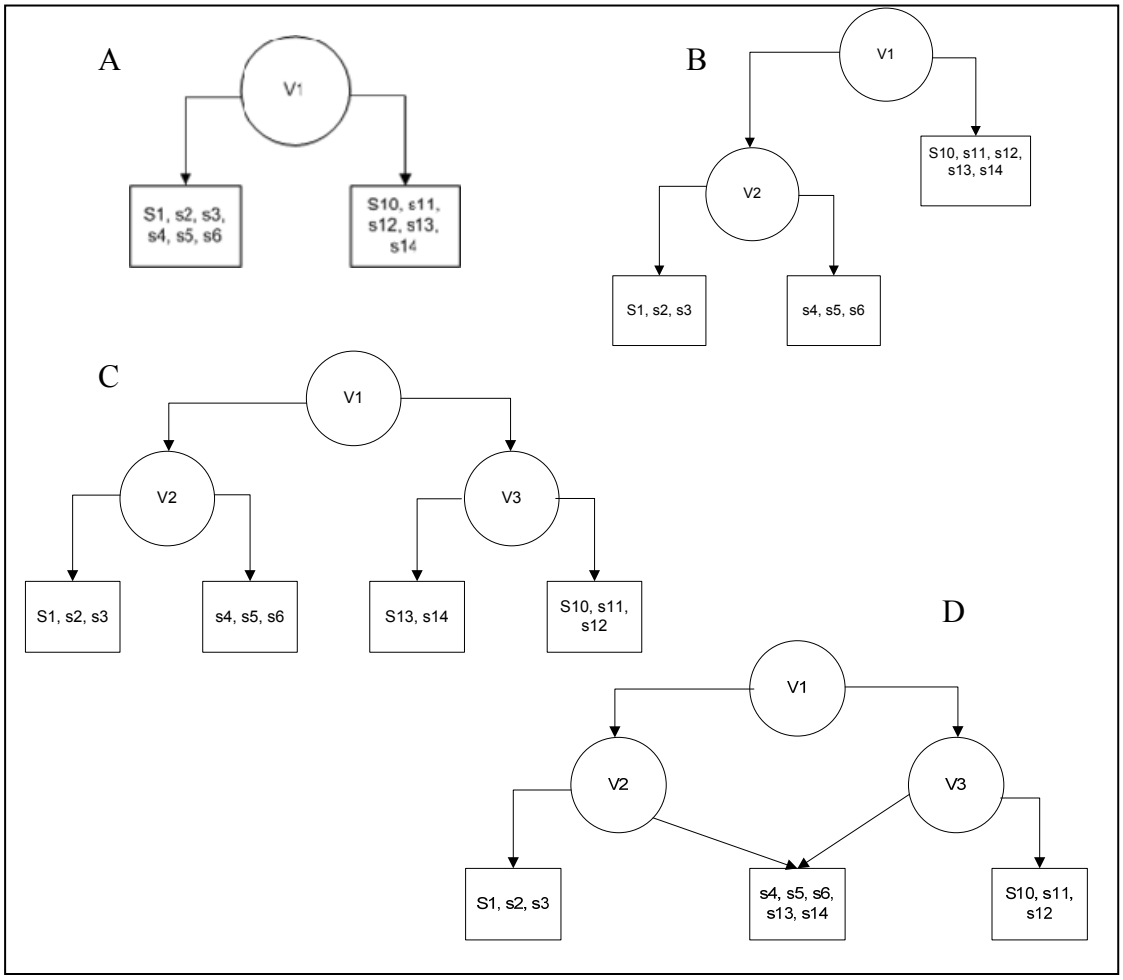


Figure 3-5. How Breadth First Marker Propagation (BFMP) allows for sample tracking in both decision trees and decision graphs. A-D represent the different stages of BFMP combined with Decision Graphs.

Input: A list of variables V = {v_1, v_2, v_3, v_m} where v_i is variable i
Variable T which is the target variable
Output: A Bayesian Local Structure M
L = All leaves of Model M where L_j is leaf j
O = Set of operators BinarySplit(M, L_j,v_i) and CompleteSplit(M, L_j,v_i)
Var(L_j) = All v_i that are on path from L_j to root
Merge(M,L_j,$L_{j'}$) = Merges Leaves L_j and $L_{j'}$ in model M

1 M'={T}
2 DO
2 M = M'
3 ΔScore =
$\max(\max_{l\in L, v_i \in V-Var(l), o\in O} Score(o(M,l,v_i)), \max_{l_i,l_j \in L, l_i \neq l_j} Score(Merge(M,l_i,l_j))) -$
Score(M)
5 IF ΔScore > 0
6 M' =
$\arg\max($
$\max(\max_{l\in L, v_i \in V-Var(l), o\in O} Score(o(M,l,v_i)), \max_{l_i,l_j \in L, l_i \neq l_j} Score(Merge(M,l_i,l_j))))$
7 END IF
8 WHILE (ΔScore > 0)
9 Return M'

Figure 3-6. The pseudocode for Constrained Greedy Bayesian Local Structure Decision Graph algorithm.

3.1.4 Search Paradigms

Using the global or local structure as the rule-generating model requires an investigation into the search strategy for generating models. While Greedy search is effective [68, 124], there are many other possible search strategies that one can employ when building a Bayesian model. We will explore two alternative methods besides greedy search, which include beam and parallel greedy search. The nomenclature that we used to represent the learning is as follows:

Table 3-1. Nomenclature for labeling Greedy, Beam, and Parallel Greedy Searches.

Structure Type	Type of Local	Greedy	Beam	Parallel Greedy
Global	N/A	BGRL_G	BGRL_B	BGRL_PG
Local	Decision Tree	BLSRL_DT_G	BLSRL_DT_B	BLSRL_DT_PG
	Decision Graph	BLSRL_DG_G	BLSRL_DG_B	BLSRL_DG_PG

3.1.4.1 Beam Search

Beam search is a heuristic search algorithm that attempts to constrain best-first search to reduce its memory requirements. It reduces this by constraining the number of best prior models kept by ranking them and only keeping a specific number of them. The heuristic component is how the models are ranked in order to select the most profitable model for exploration. These heuristics, for example positive predictive value, signal-to-noise, F-measure, K2 [70], BDEu [126], try to estimate how well the model captures the information in the data. Definitions of a model also vary depending on the problem. For example, in [20] each rule is considered a model with PPV as the heuristic used. For models such as those developed in [67, 68, 124], the score called K2 is used for determining how well a Bayesian model fits the data. Irrespective of the choice of ranking or model class, beam search allows one to store the best B models, where B is the size of the beam. The hope is that one of the models on the beam will allow you to avoid local optima.

While beam search can be an effective method of search, it has similarities with breadth-first and depth first in that it could theoretically search *ad infinitum*. Stopping criteria are often required in order for beam search to be effective, which limits the optima that can be reached. For BNs, often the maximum number of parents is an example of a heuristic that is used as stopping condition.

Figure 3-7 describes the standard beam search algorithm as was implemented in [1, 20], to which we also added Line 13, which prevents the repetition of models as explained by the following scenario. If adding variable *B* to the model $A \rightarrow T$ (where *A* is a parent of *T*) increases the score, then we wish to remove the possibility of revisiting that structure by considering adding variable *A* to the Model $B \rightarrow T$ (where *B* is a parent of *T*).

Unfortunately, for Bayesian Local Structure, even with this stopping condition of max number of variables on a path, one cannot utilize the added heuristic of line 13 in Figure 3-7. Just because you have seen the variable along one path, does not eliminate the possibility of seeing

B = Priority Queue of Size m that sorts Bayesian Networks by their K2 Score
V = $\{v_1, v_2, \ldots v_d, C\}$ All the attributes within the dataset D and Target attribute C
V_{Bi} = All the variables in in Bayesian Network i
I = $\{s_1, s_2, \ldots, s_d\}$ All instances within the dataset D
FB = Final Beam of Bayesian Network Models of size M
Init: Create all single Bayesian Network Models $v_i \rightarrow C$ and add them to B

1) While Bm is not Ø
2) Bi ← First Model on B
3) B ←B-B_i
4) NBi ←Bi
5) For v_i in V - V∩V_{Bi}
6) NBi ←Add V_i as parent of C in Bi
7) IF NBi K2 score is greater than Bi
8) B ← B U NBi
9) ENDIF
10) ENDFOR
11) IF No v_i not in Bi improves Bi
12) FB = FB U Bi
13) V = V - V_{Bi}
14) ENDIF
15) ENDWHILE
16) Return First Model on FB

Figure 3-7. Beam Search for Constrained Bayesian Global Networks for rule generation.

that same variable along an additional path. The search space is too large making beam search intractable for local search. This richness in possible structures, which could all be equivalent makes beam search almost intractable in practice, and therefore I excluded it as a search strategy for Bayesian local structure. Therefore, in the results' section, I don't provide performance statistics for BLSRL_DT_B or BLSRL_DG_B.

3.1.4.2 Parallel Greedy Search

Parallel Algorithms utilize seed models to generate many different models in parallel and then compare the final results. This is particularly effective in greedy search since each seed model search is independent of the other seed models [80, 146]. I use this same idea of multiple seed points to help greedy search avoid as many local maxima as possible both in the global and local structure search. However, given the large dimensionality, 10,000 or greater variables per a dataset, I used a fixed number (1000) of top univariate models ($V \rightarrow T$) by K2 score as seed models for either local or global structure greedy searches. To search for the top 1000 models, I used the beam search algorithm that was implemented for Bayesian Global Beam Search (Section 3.1.4.1).

The search algorithm differs in implementation for both *global* and *local structure* changes. For Bayesian global network search, instead of starting with a simple model of just the Target variable (Line 1 in Figure 3-8), we utilize the initialized single variable models as seen in Line 3. This forces the algorithm to explore an expanded set of initial parents for the target variable. Since we are performing multiple greedy searches, the time complexity and space complexity of the algorithm is only extended by a constant, yet, as we will see in the results, there are some interesting effects that occur when one forces the algorithm to have a parent *a*

priori to structure learning. This is similar to setting the prior weight of ($Vi \rightarrow T$) to be 1 if structure priors were to be incorporated explicitly into this algorithm.

Input: A list of variables V = $\{v_1, v_2, v_3, \ldots, v_m\}$ where v_i is variable i
Variable T which is the target variable
Output: A Bayesian Local Structure M
O = Set of operators Add(M,v_i) which adds vi as a parent of T in model M
Var(M) = All vi that are in model M
B = Priority Queue of Size m that sorts Bayesian Networks by their K2 Score
Init: Create all single Bayesian Network Models $v_i \rightarrow$ C and add them to B

1 M_F = M'={T}
2 WHILE (B != Ø)
3 M'= First Model on B
4 B = B – M'
5 DO
6 M = M'
7 ΔScore = $\max_{v_i \in V - Var(M)} Score(Add(M, v_i))$ - Score(M)
8 IF ΔScore > 0
9 M' = $\arg\max_{v_i \in V - Var(M)} Score(Add(M, v_i))$
10 END IF
11 WHILE (ΔScore > 0)
12 IF Score(M') > *Score*(M_F)
13 M_F = M'
14 ENDWHILE
15 Return M_F

Figure 3-8. The pseudocode for Parallel Greedy Constrained Global Bayesian Network algorithm.

The Bayesian local structure parallel algorithms differ from global parallel algorithms significantly. Lines 1-8 in Figure 3-9 and Figure 3-10 show how we initialize all the possible start states. First, we have the beam of size B that restricts our starting space. Then we utilize the operators available, which in this case are the same for both the decision tree and the decision graph. This algorithm can be transformed into greedy search by changing the beam size to one.

As with the Parallel Genetic algorithms, this multi-seed approach is used in the hope of avoiding local maxima. Since I don't do exhaustive seeding, the use of beam B in Lines 1-12 in Figure 3-9 and Figure 3-10, increases the computational complexity by only a constant. Parallel greedy search paradigms also present the opportunity to parallelize the search thus decreasing overall run time. Each greedy search run (Lines 13-19) is logically separated from all the other runs and therefore can be sent to different processers.

Input: A list of variables V = {v_1, v_2, v_3, v_m} where v_i is variable i
Variable T which is the target variable
Output: A Bayesian Local Structure M
O = Set of operators BinarySplit(M, l,v_i) and CompleteSplit(M, l,v_i)
L = All leaves of Model M
Var(l) = All v_i that are on path from l to root
B = Priority Queue of Size m that sorts Bayesian Networks by their K2 Score

1 B = {}
2 For all v_i in V
3 For all *o* in O
4 For all l in L({t})
5 B = B U *o* ({T},l,v_i)
6 ENDFOR
7 ENDFOR
8 ENDFOR
9 M_F = M'={T}
10 WHILE (B != Ø)
11 M'= First Model on B
12 B = B – M'
13 DO
14 M = M'
15 ΔScore = $\max_{l \in L, v_i \in V - Var(l),\, o \in O} Score(o(M, l, v_i))$ - Score(M)
16 IF ΔScore > 0
17 M' = $\arg\max_{l \in L, v_i \in V - Var(l),\, o \in O} Score(o(M, l, v_i))$
18 END IF
19 WHILE (ΔScore > 0)
20 IF Score(M') > *Score*(M_F)
21 M_F = M'
22 ENDWHILE
23 Return M_F

Figure 3-9. The pseudocode for the Parallel Greedy Bayesian Local Structure Decision Tree Algorithm.

Input: A list of variables V = {v_1, v_2, v_3, v_m} where v_i is variable i
Variable T which is the target variable
Output: A Bayesian Local Structure M
O = Set of operators BinarySplit(M, l,v_i) and CompleteSplit(M, l,v_i)
Merge(M,l_i,l_j) = Merges the counts of leaves i and leaves j and connects all parents
L = All leaves of Model M
Var(l) = All v_i that are on path from l to root
B = Priority Queue of Size m that sorts Bayesian Networks by their K2 Score

1 B = {}
2 For all v_i in V
3 For all *o* in O
4 For all l in L({t})
5 B = B U *o* ({T},l,v_i)
6 ENDFOR
7 ENDFOR
8 ENDFOR
9 M_F = M'={T}
10 WHILE (B != Ø)
11 M'= First Model on B
12 B = B – M'
13 DO
14 M = M'
15 ΔScore =
$\max(\max_{l\in L, v_i \in V-Var(l),\, o\in O} Score(o(M,l,v_i)), \max_{l_i,l_j\in L, l_i\neq l_j} Score(Merge(M,l_i,l_j)))$
- Score(M)
16 IF ΔScore > 0
17 M' =
$\arg\max($
$\max(\max_{l\in L, v_i \in V-Var(l),\, o\in O} Score(o(M,l,v_i)), \max_{l_i,l_j\in L, l_i\neq l_j} Score(Merge(M,l_i,l_j))))$
18 END IF
19 WHILE (ΔScore > 0)
20 IF Score(M') > *Score*(M_F)
21 M_F = M'
22 ENDWHILE
23 Return M_F

Figure 3-10. The pseudocode for the Parallel Greedy Bayesian Local Structure Decision Graph algorithm.

3.2 COMPUTATIONAL COMPLEXITY FOR THE ALGORITHMS

3.2.1 Greedy Search

3.2.1.1 BGRL

As shown in Figure 3-3, if we do not limit the number of possible parents so the maximum number of loops is $u = n$ where u is the maximum number of parents and n is the number of variables. Following the time complexity from Cooper and Herskovits [70], we can derive our implementation's time complexity with modification for elimination of the outer loop. Therefore the unrestricted time complexity for the constrained Bayesian global structure search algorithm is $O(m+r-1)+O(mn^2r)O(\text{n})$ which is equivalent to $O(mn^3r)$. We can reduce the machine run time if we use factorial caching, logarithmic calculations, and the database look up by using breadth first marker propagation. Practically, however, setting u = n is not feasible due to the dimensionality. This means that ultimately that the computational complexity is only $O(mu^2nr)$ where u is maximum specified parent. The space of the greedy search is $O(mn)$.

3.2.1.2 BLSRL_DT and BLSRL_DG

As shown in [67, 68, 124], learning the possible combinations is exponential or $O(2^n)$, however given that this is a constrained space and using the algorithm in Figure 3-4 and Figure 3-6, we have the worst case scenario of $O(mn2^n)$ for time complexity when there is no maximum number of parents set. The machine run time is reduced by the use of breadth first marker propagation and by limiting the maximum number of parents. The space complexity is equivalent to the BGRL space complexity in that it only has to store a single model and hence its complexity is only $O(\underline{mn})$.

3.2.2 Beam Search

Given the computational complexity expressed in section 3.2.1.2, it is not feasible to perform beam search for Bayesian local structure. Therefore the only Bayesian rule generation (BRG) algorithm that currently utilizes a beam search with a beam greater than size 1 (greedy search can be considered beam search of size 1) uses a constrained BN model with global structure (Bayesian local structure decision tree with the constraint of a complete tree).

3.2.2.1 BGRL

If we assume each model can generate at most p possible successors, a beam of size b, and the computational complexity derived in the greedy search, we arrive at a time complexity of $O(bnp)O(mu^2nr) = O(bpmu^2n^2r)$. This time complexity includes repetitive models, and so actual run time can be significantly less due to the incorporation of Line 13 in Figure 3-7. This removes a lot of the overlap thus reducing p. The space complexity, due to implementation using the priority queue, increases from $O(mn)$ to $O(bmn)$.

3.2.3 Parallel Greedy Search

3.2.3.1 BGRL

This method allows the usage of multiple seeds to start, so the time complexity is only $O(bmu^2nr)$, where b is the number of seeds. If done serially, the total space complexity at any one time is $O(mn)$. However, when performing it in parallel, the space complexity is $O(bmn)$.

3.2.3.2 BLSRL-DT and BLSRL-DG

Similarly to the BGRL parallel greedy search, these algorithms use multiple seeds to start so the time complexity is $O(bmn2^n)$ where b is the number of seeds used to start. We can reduce the machine run time by using breadth first marker propagation, limiting the maximum number of parents, and caching the factorials. If done serially, the total space complexity at any one time is $O(mn)$. However, when performing it in parallel, the space complexity is $O(bmn)$.

3.3 METHODOLOGY FOR EVALUATION OF BRGF

This section explains the methodology of testing the Bayesian global and Bayesian local structure algorithms that have been modified to only search over parents of the target variable. We first show in section 3.3.1 the simulated data that will be used to test the correctness of the algorithm. That is, we know the generating model for the data and we want to verify that the algorithms produce this model. Then in section 3.3.2, we show the biomedical 'omic' datasets that we will analyze with the various algorithms. In section 3.3.3, we show the experimental workflow that was used including the pseudo-code to run the experiments. Section 3.3.4, describes the performance measures used to evaluate the model. The final section in the chapter, section 3.3.5, explains the statistical analysis used for comparing the performance measures of the experiment.

3.3.1 Simulated Data

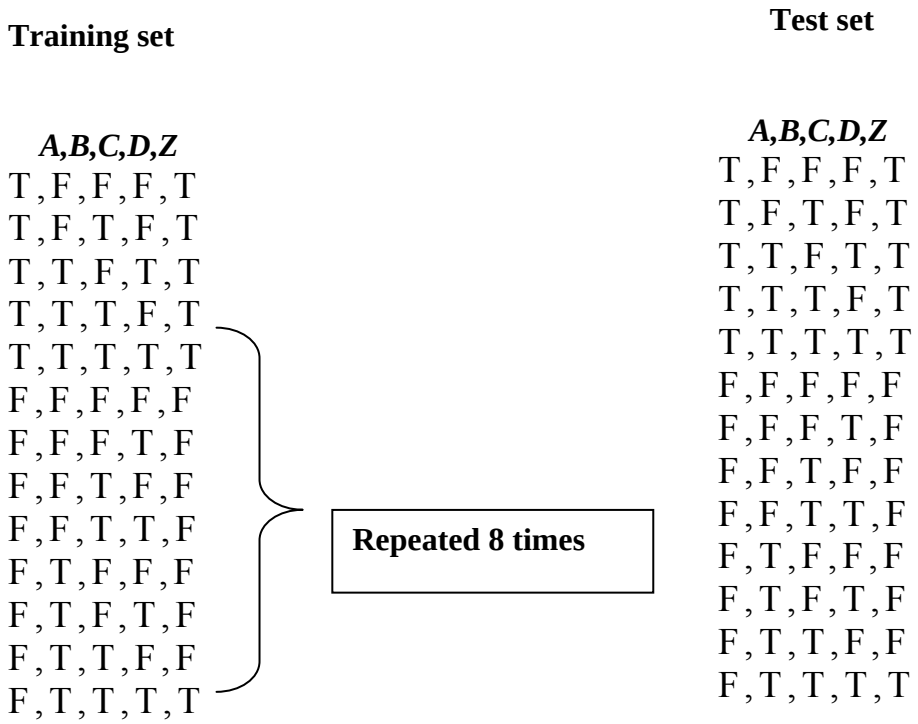


Figure 3-11. The simulated data used to verify the Bayesian Structure Search algorithms. Originally developed by Dr. Visweswaran in [124].

Verification of the ability of the algorithms to produce known generating structure was done using simulated data created by Visweswaran [124]. We utilized his simulated data since it was developed for the use of testing Bayesian local and global structures. The simulated data was derived from the deterministic function expressed in the equation below.

$$Z = A \vee (B \wedge C \wedge D) \quad \textbf{(16)}$$

This equation, where Z is the target variable, A, B, C, and D are the observed variables and all variables are binary, was used to generate the following data in Figure 3-11. Using the simulated data, there are multiple structures that are generated by the global and local structure searches. All of the various search strategies (greedy, beam, and parallel greedy) return very similar structures as seen in Figure 3-12.

Table 3-2. The counts from the model in Figure 3-12 (1) which is the Bayesian Global Structure. Some of the parameters are 0.5 since in the training data, there is no training case that exemplifies that specific parent state. A = {*A*=T}, ‘A = {*A*=F}, B = {*B*=T}, ‘B = {*B*=F}, C = {*C*=T}, ‘C = {*C*=F}, D = {*D*=T}, ‘D = {*D*=F}.

Z	A								‘A							
	B				‘B				B				‘B			
	C		‘C		C		‘C		C		‘C		C		‘C	
	D	‘D	D	‘D	D	‘D	D	‘D	D	‘D	D	‘D	D	‘D	D	‘D
T	2	2	2	1	1	2	1	2	1	1	1	1	1	1	1	1
F	1	1	1	1	1	1	1	1	9	9	9	9	9	9	9	9

Table 3-3. The counts from the model in Figure 3-12 (2) which is the Bayesian Local Structure Decision Tree. The combination of multiple cells, e.g., the collapse of all the cells under A results in fewer cells in the table decreasing the number of parameters needed.

Z	*A* = T	*A* = F			
		B = T			*B* = F
		C = T		*C* = F	
		D = T	*D* = F		
T	6	9	1	1	1
F	1	1	9	17	33

Table 3-4. The counts from the model in Figure 3-12 (3) which is the Bayesian Local Structure Decision Graph. The difference between the decision graphs and the decision tree is that the graphs combine the counts.

Z	*A* = T	*A* = F			
		B = T			*B* = F
		C = T		*C* = F	
		D = T	*D* = F		
T	15	15	1	1	1
F	1	1	59	59	59

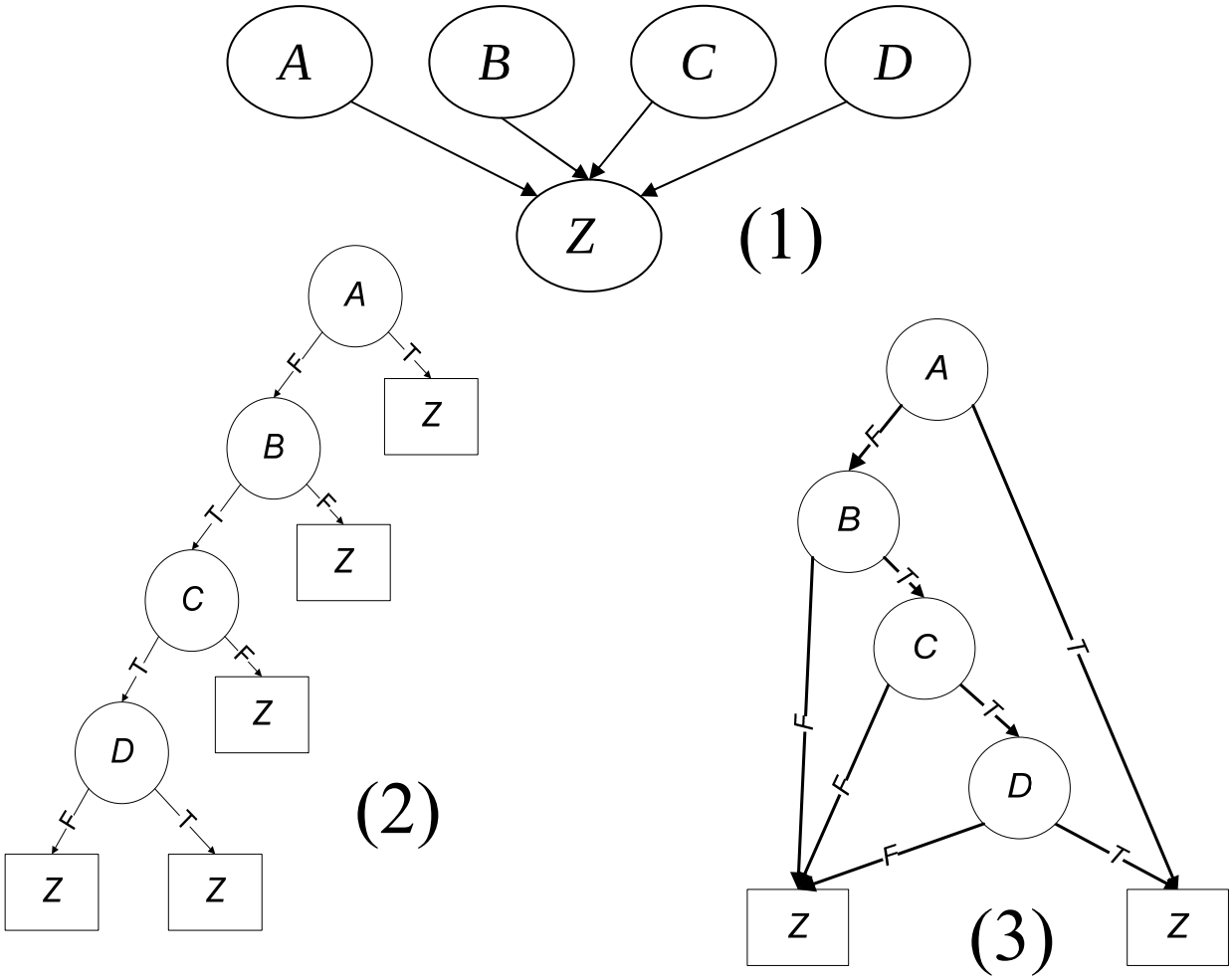


Figure 3-12. Examples of the structures generated from the simulated data. (1) is the global structure, (2) is the Bayesian Local Structure Decision Tree model, and (3) is the Bayesian Local Structure Decision Graph model. Z represents the target variable while A, B, C, and D are the observed variables.

As seen by Figure 3-12 and their respective counts in Table 3-2, Table 3-3, and Table 3-4, there are distinctive differences in the parameterization between the global structures and the local structures after learning the model on the training data. Specifically, in Table 3-2, there are cells, which only have the prior counts of 1 in them. These cells are empty due to the sparseness within the data. When performing inference and constructing rules, those rules are defaulted to

Z=T. As we see in Table 3-5, the choice of the default can dramatically affect the classification results.

Table 3-5. The performance results of application of Bayesian Rule Generation using various structures. These results were achieved using the default classification of P(*Z*=T | E) when P(*Z*=T | E) = P(*Z* = F | E).

Structure type	Accuracy	Robustness	RCI
Bayesian Global Rule Learning	100%	100%	100%
Bayesian Local Structure Decision Tree Rule Learning	100%	100%	100%
Bayesian Local Structure Decision Graph Rule Learning	100%	100%	100%

As we see, all the algorithms perform equally well when using the model derived from the training data to perform inference on the test data. However, this result could be from the default prediction of the first target value encountered in the data when building the model. If the algorithm set the default when choosing between two equivalent probabilities for two target values to the second value, the Bayesian Global Rule Learning would have an incorrectly predicted on all of the test cases leading to an accuracy of 0% and a Robustness of 0%; however, both Bayesian Local Structure rule learning methods would have the same performance on this dataset regardless of the default choice when given equivalent probabilities. This highlights the importance of context-specific independencies due to the ability to differentiate the generating model using them.

3.3.2 Biological Data

To test the BRGF, I have utilized 24 publicly available datasets that span the spectrum of genomic and proteomic data. There are many more datasets available within the genomic area; however, there are few publicly available proteomic datasets. To increase the number of proteomic datasets, I utilized six non-public datasets generated at the University of Pittsburgh.

These datasets are detailed in Table 3-6 and Table 3-7. It also lists the number of variables and the percentage of the samples that occupy the majority class. In Table 3-6, at least 17 out of the 24 listed datasets have majority class samples and are significantly skewed.

Table 3-6. Publicly Available Biological Datasets that will be used in the evaluation of the BRGF Framework. Type refers to the type of data where G is genomic and P is proteomic. P/D is used to express whether the data is Prognostic (survival) or Diagnostic. M is the percentage of the data that is represented by the majority class. An M of 0.75 would represent that 75% of the data is of one class.

Dataset	Type	P/D	# Classes	# Variables	# Samples	M	Ref #
Alon et al.	G	D	2	6584	61	0.651	[147]
Armstrong et al.	G	D	3	12582	72	0.387	[7]
Beer et al.	G	P	2	5372	86	0.795	[92]
Bhattacharjee et al.	G	D	5	12600	203	0.657	[85]
Bhattacharjee et al.	G	P	2	5372	69	0.746	[85]
Golub et al.	G	D	4	7129	72	0.513	[83]
Hedenfalk et al.	G	D	2	7464	36	0.500	[84]
Iizuka et al.	G	P	2	7129	60	0.661	[91]
Khan et al.	G	D	4	2308	83	0.345	[86]
Nutt et al.	G	D	4	12625	50	0.296	[148]
Pomeroy et al.	G	D	5	7129	90	0.642	[149]
Pomeroy et al.	G	P	2	7129	60	0.645	[149]
Ramaswamy et al.	G	D	26	16063	280	0.574	[89]
Rosenwald et al.	G	P	2	7399	240	0.145	[79]
Staunton et al.	G	D	9	7129	60	0.506	[150]
Shipp et al.	G	D	2	7129	77	0.746	[8]
Singh et al.	G	D	2	10510	102	0.510	[9]
Su et al.	G	D	11	12533	174	0.150	[87]
Veer et al.	G	P	2	24481	78	0.562	[93]
Welsch et al.	G	D	2	7039	39	0.878	[81]
Yeoh et al.	G	P	2	12625	249	0.805	[94]
Petricoin et al.	P	D	2	11003	322	0.784	[103]
Pusztai et al.	P	D	3	11170	159	0.364	[151]
Ranganathan et al.	P	D	2	36778	52	0.556	[152]

Table 3-7. Additional proteomic datasets that are not publicly available. #C is the cardinality of the target variable and #V is the number of variables within the dataset. JLL refers to Jonathan L. Lustgarten. For the MALDI-TOF dataset, I was the one who created the dataset using samples from the Bowser lab.

Dataset	Type	P/D	# C	# V	# Samples	M	Lab
SELDI-TOF ALS Plasma	P	D	2	42543	67	0.507	Bowser
MALDI-TOF ALS CSF	P	D	2	34400	22	0.545	JLL/Bowser
SELDI-TOF Lung Cancer	P	D	2	19581	239	0.560	Lung SPORE
SELDI-TOF ALS CSF	P	D	2	17914	168	0.667	Bowser

3.3.3 Experimental Design

To test the different algorithms and different discretization methods, I utilized an experimenter that had the following logical flow where a run refers to a new random stratification and a fold is a specific split of the data between training and testing:

```
Di = The i^th dataset of all datasets
Mj = The j^th discretization method of M where M = {MDLPC, EBD}
Ck = The k^th classifier that consists of a classification algorithm and search type
Res_CkMjDirf = Stores the predictions made by the k^th classifier, j^th discretization method, and the
specific train test pair of the D_irf datasets (i^th dataset, r^th run, f fold)

1 For all Di in D
2  For r = 1 to R runs
3    For f = 1 to F folds
4      For all M_j in M
5        For all C_k in C
6                  Res_CiMiDirf = Run Classifier C_i using discretization Method M_i on Dataset D_irf
7                  Store Res_CiMiDirf
8          END FOR
9     END FOR
10  END FOR
11 END FOR
12 Calculate all the performance measures for all Results
13 Output Summary Results
```

Figure 3-13. The pseudocode for the Experimenter used in testing the different algorithmic and discretization method combinations.

As we see this is an exhaustive combination iterating over every possible combination of classifier, discretization method, and dataset which are passed as parameters. For the purpose of this experiment, we used 10 runs of 10 fold cross-validation over 24 datasets using 9 different classifiers and 2 different discretization methods explained in Sections 2.2.2, 3.1.1.1 and 3.1.2-4. This results in 50,400 models generated using the flow shown in Figure 3-13.

3.3.4 Performance Measures

The following measures represent a chosen subset of possible measures [153, 154]. Since there are both probabilistic and discrete classifiers within the chosen classification algorithms, the performance measures need to be chosen accordingly. All of these measures will be tested using different statistical tests that are explained in Section 3.3.5. The discrimination measures used include accuracy (ACC), balanced accuracy (BACC), and Relative Classifier Information (RCI). The discrimination measures evaluate how well the classifier differentiates between the values of the target variable (classes). The Timing (TIME) and Model Complexity (MC) measures are more descriptive than discriminative since these measure look at characteristics of the algorithm instead of prediction results.

Table 3-8. A brief description of the performance measures used. For the Timing measure, the closer it is to 0 mins, the better. For Model Complexity it is preferable to have a less complex model though there is no absolutely best score. For all other measure, 100 is the best score. N represents the number of variables within a dataset.

Measures of Performance	Abbreviation	Range	Best Score
accuracy	ACC	[0 - 100]	100
balanced accuracy	BACC	[0 - 100]	100
Relative Classifier Information	RCI	[0 - 100]	100
Timing	TIME	[0 - ∞]	0
Model Complexity	MC	$[0 – 2^n]$	NA

3.3.4.1 Accuracy

Accuracy measures how many correct predictions a particular classifier makes. This measure, while descriptive, does not account for skewed distributions (more of one type of class within the data), which as seen in Table 3-6, can result in high predictive accuracies. This resulted in our use of alternative measures such as balanced accuracy and Relative Classifier Information. An ACC score of 100% represents perfect discrimination.

3.3.4.2 Balanced Accuracy

Balanced accuracy (BACC) compensates for skewed distribution of classes in a dataset. Balanced accuracy is defined as follows:

$$BACC = \frac{\sum_{C} Sensitivity(c) + Specificity(c)}{|C|}$$
$$Sensitivity(c) = \frac{TP_{(c|c)}}{TP_{(c|c)} + FN_{(\neg c|c)}}$$
$$Specificity(c) = \frac{TN_{(\neg c|\neg c)}}{TN_{(\neg c|\neg c)} + FP_{(c|\neg c)}} \quad (17)$$

where the |C| is the cardinality of the target variable and Sensitivity(c) and Specificity(c) refer to the sensitivity of target value c versus all other values of the target. $TP_{(c|c)}$ is the number of samples predicted to be c given that its observed target value is c, $FN_{(\neg c|c)}$ is the number of samples predicted to be another target value that is not c given that it's observed value is c, $TN_{(\neg c|\neg c)}$ is the number of samples predicted to be another target value besides c given that its observed value is not c, and $FP_{(c|\neg c)}$ is the number of samples predicted to be target value c given that it's observed value is not c. One characteristic of this score is that it measures sensitivity and specificity as a one vs. rest problem, which means the positive class is c and the negative class is $\neg\, c$ for all c within the dataset. Unlike accuracy however, BACC is not easily translatable into the number of predictions correct due to the described one vs. rest approach. A BACC score of 100% represents perfect discrimination.

3.3.4.3 Relative Classifier Information

Relative Classifier Information (RCI) is an entropy-based performance measure of a classifier that quantifies the amount by which the uncertainty of a decision problem is reduced by a

classifier compared to predicting the majority class for the target for every instance [122]. For a confusion matrix (a table of showing the distribution of predictions over the various classes) of J columns and I rows representing the predictions of the classifier compared to the observed value (this includes the ability to have "No prediction"), let q_{ij} = the number of times Class i was called Class j by the classifier and I be a previously unseen case.

$$P(I \in C_i) = \frac{\sum_j q_{ij}}{\sum_{ij} q_{ij}} \tag{18}$$

and the uncertainty of the classification of that case is:

$$H_d(I) = \sum_i -P(I \in C_i)\log(P(I \in C_i)) \tag{19}$$

The probability that the new sample is an Element of C_i given that the Output from the classifier is C_j is:

$$P(I \in C_i \mid O \in C_j) = p_{ij} = \frac{q_{ij}}{\sum_i q_{ij}} \tag{20}$$

Therefore the uncertainty of the unseen case I given that the Output is C_j is:

$$H_{O_j}(I \mid O \in C_j) = \sum_i -P(I \in C_i \mid O \in C_j) Log(P(I \in C_i \mid O \in C_j)) \in = \sum_i -p_{ij}\log(p_{ij}) \tag{21}$$

The probability of the output O being of Class j:

$$P(O \in C_j) = p_j^{out} = \frac{\sum_i q_{ij}}{\sum_{ij} q_{ij}}$$

(22)

Combining Equation 22 and Equation 23, we calculate the uncertainty of the classifier output as:

$$H_o(I \mid O) = \sum_j P(O \in C_j) H_{O_j}(I \mid O \in C_j)$$

(23)

Using Equation 20 and subtracting the results of Equation 24, one gets how much uncertainty is captured by the classifier.

$$H_{classifier} = H_d - H_O$$

(24)

To get the RCI, Equation 25 is divided by equation 20 and multiplied by 100%.

Equation 22 can be interpreted as information gain and thus RCI can be reworded to say it measures the information gain of using the classifier over a majority classification algorithm which always predicts the most abundant class. Its range is from 0%, denoting the worst performance, to 100%, denoting perfect discrimination. It is similar to the Area Under the Receiver Operator Characteristic curve (AUROC) in that it measures the discriminating power of the classifier while minimizing the effect of the class distribution.

3.3.4.4 Running Time

The time for classification is an interesting measure that incorporates multiple aspects of the classification algorithm. It not only incorporates running time, e.g., how long it takes in minutes the algorithm to build a model from the input data, but also inference.

3.3.4.5 Model Complexity (Number of Variables and Rules)

Model complexity, like TIME, is also a qualitative measure. Ideally, the simpler the model, the more general it is; however, given noise, variability, and the complexity of the information within the data, sometimes the simpler model does worse because it does not capture the correct interactions between the variables of the data [155]. For the purposes of this experiment, the MC is expressed by two numbers: the number of variables and the total number of rules used in the model. This is a qualitative measure given that we limit the total number of possible parents in the Bayesian global methods to 8 and the maximum number of variables per a path to 8 for Bayesian local methods, which means it is possible to have more than 8 variables used in the model.

3.3.5 Statistical Analysis

For this thesis I used two common statistical tests, the Wilcoxon paired samples signed rank test and the paired samples t-test. The Wilcoxon paired samples signed rank test is a non-parametric procedure used to test whether there is sufficient evidence that the median of two probability distributions differ in location. In evaluating algorithms, it is used to test whether two algorithms differ significantly in performance on a specified measure. Being a non-parametric test, it does not make any assumptions about the form of the underlying probability distribution of the sampled population. The paired samples t-test is a parametric procedure used to determine whether there is a significant difference between the average values of the same performance measure for two different algorithms. The test assumes that the paired differences are independent and identically normally distributed. Although the measurements themselves may not be normally distributed, the pair-wise differences often are.

These two tests cover two different approaches (normal, and no-distributional) to testing for significance. The normal assumption is often used since it is well characterized. If the results are not normally distributed, instead of trying to match the most likely distribution, most scientists utilize a test that requires no normality assumption, such as the Wilcoxon paired samples signed rank test.

4.0 SUFFICIENCY OF THE BAYESIAN RULE GENERATION FRAMEWORK

For the purposes of demonstrating sufficiency of the BRGF for 'omic' data mining, I will give examples of the two most important components, discretization and probabilistic rule learning. The first example will be showing the sufficiency of Bayesian discretization using the EBD method defined in Section 3.1.1.1. The second example will be on a proteomic dataset showing the results both in rule form and the BN form from the analysis of a publicly available prognostic genomic dataset [85].

4.1 BAYESIAN DISCRETIZATION

We compared the performance of the EBD method (see Section 3.1.1.1) with the performance of the MDLPC (see Section 2.2.2) method on 23 biomedical genomic and proteomic datasets (see Table 4-1) using two measures: accuracy and relative classifier information (RCI). The two selected measures evaluate the performance of classifiers that are learned from the discretized variables. For classification, we used the naïve Bayes classifier which is simple, efficient and robust and accepts both continuous and discrete variables. Although the naïve Bayes classifier accepts continuous predictor variables, the performance of the classifier tends to be better when the predictors are discretized than when the predictors are modeled with a normal distribution [45] .

Table 4-1. The datasets used for the discretization comparison between MDLPC and EBD to prove sufficiency. In the Type column, G denotes *transcriptomic* and P denotes *proteomic*. In the P/D column, P denotes *prognostic* and D denotes *diagnostic*. # c is the number of values of the target variable and # n is the number of instances in the dataset. # V is the number of predictor variables. M is the proportion of the data that has the majority target-value.

Dataset	Dataset name	Type	P/D	# c	# n	# V	M
1	Alon et al.	G	D	2	61	6584	0.651
2	Armstrong et al.	G	D	3	72	12582	0.387
3	Beer et al.	G	P	2	86	5372	0.795
4	Bhattacharjee et al.	G	D	7	203	12600	0.657
5	Bhattacharjee et al.	G	P	2	69	5372	0.746
6	Golub et al.	G	D	4	72	7129	0.513
7	Hedenfalk et al.	G	D	2	36	7464	0.500
8	Iizuka et al.	G	P	2	60	7129	0.661
9	Khan et al.	G	D	4	83	2308	0.345
10	Nutt et al.	G	D	4	50	12625	0.296
11	Pomeroy et al.	G	D	5	90	7129	0.642
12	Pomeroy et al.	G	P	2	60	7129	0.645
13	Ramaswamy et al.	G	D	29	280	16063	0.100
14	Rosenwald et al.	G	P	2	240	7399	0.574
15	Staunton et al.	G	D	9	60	7129	0.145
16	Shipp et al.	G	D	2	77	7129	0.747
17	Su et al.	G	D	13	174	12533	0.150
18	Veer et al.	G	P	2	78	24481	0.562
19	Welsch et al.	G	D	2	39	7039	0.878
20	Yeoh et al.	G	P	2	249	12625	0.805
21	Petricoin et al.	P	D	2	322	11003	0.784
22	Pusztai et al.	P	D	3	159	11170	0.364
23	Ranganathan et al.	P	D	2	52	36778	0.556

The naïve Bayes classifier assumes that the variables are conditionally independent of each other given the target-value. Given an instance, it applies Bayes theorem to compute the probability distribution over the target-values. This classifier is very effective when the independence assumptions hold in the domain; however, even if this assumption is violated the classification performance is often excellent when compared to more sophisticated classifiers [156].

Table 4-2. The accuracies of EBD, MDLPC, and No Discretization using a Naïve Bayes classifier. The last column reports the mean accuracies obtained from the application of the naïve Bayes classifier to the continuous-valued variables without discretization where each continuous-valued variable is represented by a Gaussian distribution. The mean and the standard error of the mean (SEM) for the percent accuracy for each dataset is obtained by averaging over a total of 20 training and test datasets. For each dataset, the highest single accuracy is shown in bold font.

Dataset	Average Accuracy		
	EBD (SEM)	MDLPC (SEM)	No Discretization (SEM)
1	97.95% (2.20)	**98.12**% (2.18)	95.00% (2.13)
2	97.99% (3.24)	97.99% (3.20)	95.89% (1.83)
3	**96.40**% (4.48)	95.47% (4.43)	73.75% (4.20)
4	**80.86**% (1.92)	78.65% (2.22)	79.79% (1.61)
5	**96.96**% (4.82)	96.74% (5.13)	68.57% (4.73)
6	97.92% (3.15)	**98.19**% (3.08)	80.18% (4.34)
7	99.72% (1.72)	99.72% (1.72)	97.50% (2.48)
8	95.83% (3.55)	**96.25**% (3.26)	65.00% (4.67)
9	98.43% (1.57)	**98.80**% (0.76)	96.39% (1.05)
10	**94.50**% (3.80)	92.10% (3.58)	70.00% (4.69)
11	95.72% (2.61)	**97.44**% (2.61)	64.33% (1.66)
12	93.58% (4.44)	**93.83**% (3.68)	84.44% (3.75)
13	**85.61**% (4.43)	73.86% (4.46)	58.33% (4.27)
14	83.81% (2.68)	**84.40**% (2.96)	44.71% (3.00)
15	**97.25**% (4.20)	94.17% (4.59)	60.00% (3.46)
16	**92.53**% (2.91)	91.69% (2.91)	55.83% (2.05)
17	97.81% (2.76)	**98.33**% (2.51)	36.67% (4.81)
18	**97.44**% (2.32)	97.18% (2.54)	71.33% (4.84)
19	100.00% (0.00)	100.00% (0.00)	83.21% (3.70)
20	**95.82**% (3.94)	95.08% (4.25)	78.20% (3.72)
21	**84.69**% (1.62)	84.12% (0.02)	60.71% (4.02)
22	58.52% (3.23)	**59.18**% (0.06)	90.00% (4.59)
23	**98.71**% (2.53)	97.98% (2.40)	70.33% (4.07)
Overall Avg.	**92.96**% (1.93)	92.14% (2.08)	72.73% (3.42)

The mean accuracies for EBD and MDLPC using the Naïve Bayes classifier are given in Table 4-2. For each dataset, the mean accuracy is obtained from 10-fold cross-validation done twice for a total of 20 folds. EBD has higher mean accuracy on 11 datasets, MDLPC has higher mean accuracy on 9 datasets, and both have the same mean accuracy on 3 datasets. Overall, EBD shows an increase in accuracy of 0.88% over MDLPC and this increase in accuracy is

Table 4-3. The average RCI for EBD, MDLPC and No Discretization using a Naïve Bayes classifier. The last column reports the mean RCI when the variables are not discretized. The mean for the percent RCI for each dataset is obtained by averaging over two runs where each run contains 10 training and test datasets. Since the mean is obtained from only two runs the standard error of the mean was not be computed. For each dataset, the highest RCI is shown in bold font.

Dataset	Average RCI		
	EBD	FI	No Discretization
1	**85.95**%	80.95%	69.54%
2	86.40%	86.40%	82.26%
3	33.44%	**34.97**%	1.04%
4	**75.69**%	75.55%	53.03%
5	**12.24**%	7.92%	4.24%
6	82.25%	**83.59**%	55.26%
7	100.00%	100.00%	79.12%
8	**32.57**%	27.83%	2.98%
9	**98.44**%	93.48%	92.63%
10	**83.51**%	83.12%	44.71%
11	**74.70**%	72.36%	50.13%
12	22.13%	**35.69**%	1.93%
13	**84.16**%	83.35%	62.90%
14	3.25%	**3.37**%	1.38%
15	**86.62**%	82.29%	28.42%
16	**59.54**%	56.31%	20.03%
17	**100.00**%	98.89%	68.18%
18	64.20%	**66.56**%	0.96%
19	100.00%	100.00%	0.00%
20	**5.94**%	3.77%	0.09%
21	22.02%	**23.43**%	7.40%
22	**26.98**%	22.48%	3.65%
23	**45.19**%	36.95%	10.28%
Overall Average	**60.22**%	59.10%	32.18%

statistically significant at the 5% significance level on the Wilcoxon signed rank test (see Table 4-4). Table 4-2 also gives the mean accuracies of the Naïve Bayes classifier that uses the continuous predictors directly without discretizing them. The results show that its accuracy of the classifier using the continuous variables is inferior to the accuracy obtained by the application of the either discretization methods.

The RCI is a measure of the discriminative performance of a classifier that is similar to the area under the Receiver Operating Characteristic curve (see Section 3.3.4.3). The mean RCIs for EBD and MDLPC using the naïve Bayes classifier are given in Table 4-3. For each dataset, the mean RCI is obtained from 10-fold cross-validation done twice for a total of 20 folds. EBD has higher RCI on 14 datasets, MDLPC has higher RCI on 6 datasets, and both have the same RCI on 3 datasets. Overall, EBD shows an improvement of 1.12% in RCI over MDLPC, and this increase in RCI is statistically significant at the 5% level on the Wilcoxon signed rank test (see Table 4-4). Table 4-3 also gives the mean RCIs of the Naïve Bayes classifier without variable

Table 4-4. Statisical comparison of EBD and MDLPC across accuracy and RCI. Results of the Wilcoxon paired samples signed rank test for accuracy and RCI over all datasets. All tests are two sided. A positive Z-score indicates a result that is favourable to EBD. Bolded p-values are below 0.05. The range of a measure is given in square brackets where *n* is the number of instances in the dataset.

Measure		Mean (SEM)	Mean Diff.	Wilcoxon p-value (Z-Score)
Accuracy [0, 100]	EBD FI	92.96 (9.17) 92.14 (9.86)	0.82	**0.026** (2.219)
RCI [0, 100]	EBD FI	60.22 (33.10) 59.10 (32.93)	1.13	**0.020** (2.320)

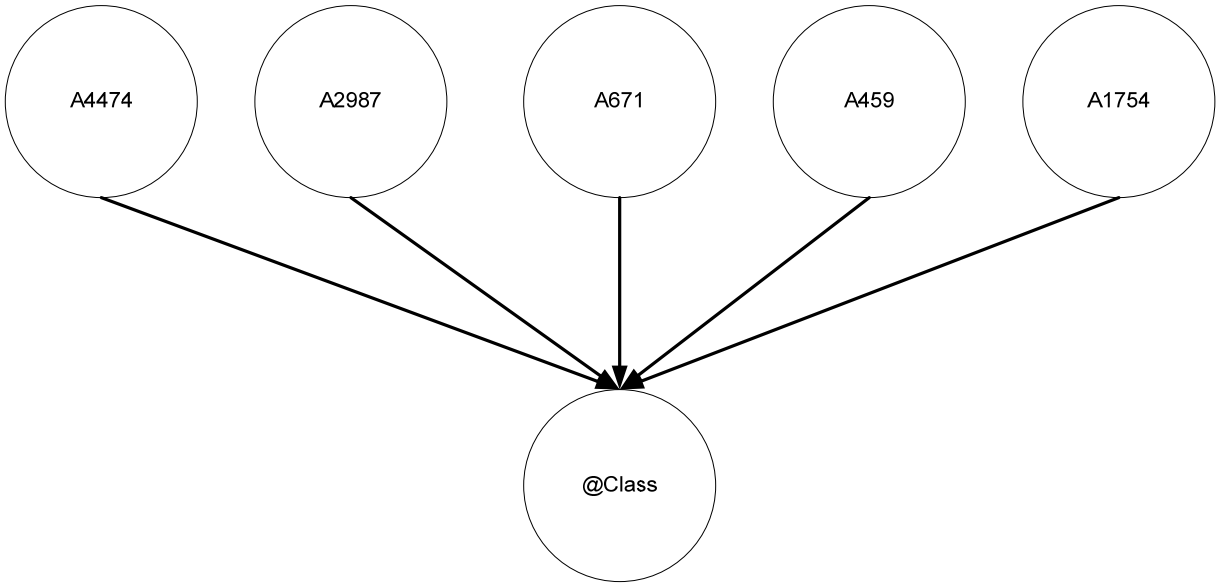
discretization which show that the RCIs obtained using the continuous variables are inferior to the RCIs obtained by the application of the discretization methods.

As seen from the performance of EBD when compared to no discretization and the de facto standard discretization method MDLPC, not only is EBD sufficient for discretization, but also it gives a performance boost that is statistically significant.

4.2 BAYESIAN RULE GENERATION

To prove sufficiency of Bayesian rule generation, we have to show that we can 1) produce a Bayesian structure that can be interpreted into rules, and 2) show that the rules have a probabilistic certainty factor. For this we will utilize the Bhattacharjee, et al. [85] prognostic genomic dataset. We will also use two different search algorithms: Bayesian global structure with a beam search (BGRL-B), and Bayesian local structure decision graph with a parallel greedy search (BLSRL-DG-PG). These examples will both highlight the fact that you can generate probabilistic rules from the constrained BNs, and that using local structure reduces the number of rules generated.

As seen in both Figure 4-1 and Figure 4-2, extracting rules allows understandability while expressing the probabilistic information contained within the CPT. It is also shown, when comparing Figure 4-1 and Figure 4-2, that using the special case of the Bayesian local structure decision tree which forces a complete tree (Bayes global structure) produces significantly more rules than using the most general local structure – the Bayesian decision graph. Appendix A contains all of the rules generated from both models. We have therefore demonstrated that it is possible to use a BN to generate rules, which are probabilistic, when constrained to only parents of the target class model.



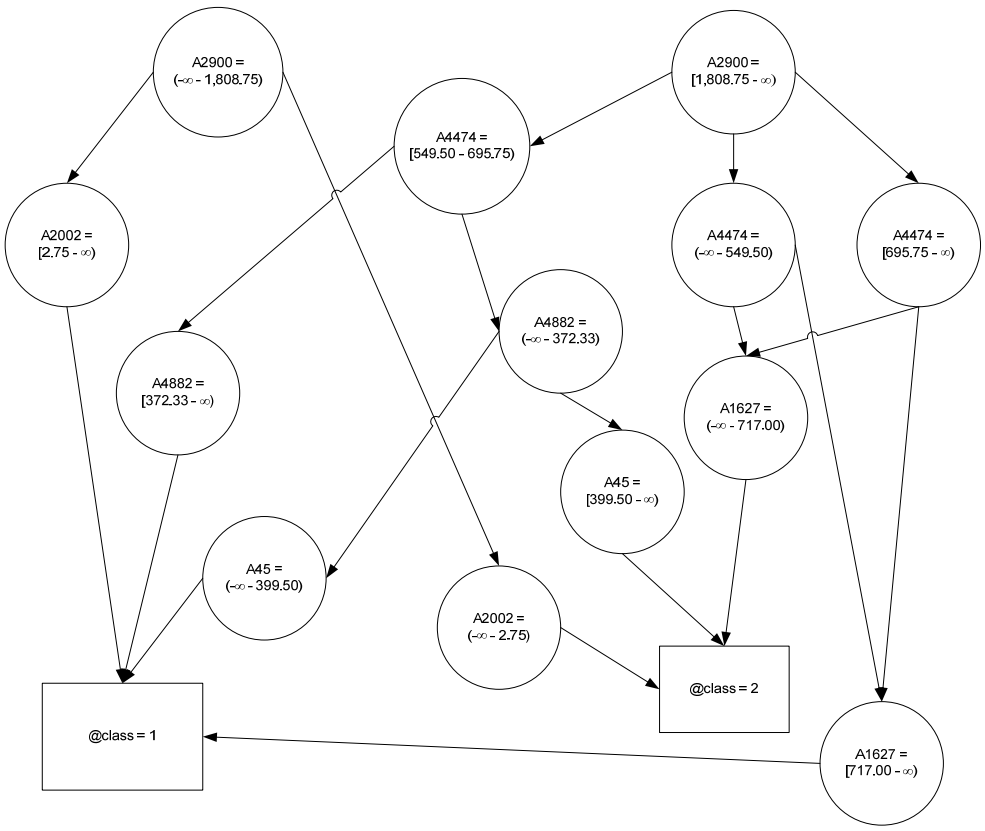
@ Class	A4474 = [695.750.. ∞) & A2987 = [-31.750.. ∞) & A671 = (-∞..51.500) & A459 = [-40.000.. ∞) & A1754 = [124.750.. ∞)	A4474 = [549.500..695.750) & A2987 = [-31.750.. ∞) & A671 = [51.500..∞) & A459 = [-40.000.. ∞) & A1754 = [124.750.. ∞)	…
Survive = 1	0	2	…
Died = 2	19	0	…

Rule 1: ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 2)
CF=0.952, Av.Cost=1.0, CF/Cost=0.952, P=0.002, TP=19, FP=0, Pos=52, Neg=17

Rule 2: ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.75, Av.Cost=1.0, CF/Cost=0.75, P=0.058, TP=2, FP=0, Pos=17, Neg=52

• • •

Figure 4-1. The Bayesian global structure learned, the parameterization, and the rule generated by the BGRF when combined with EBD discretization. The top is the Bayesian network constructed, the table in the middle represents part of the CPT for the target variable @class, and the rule list below the table represents sum of the 48 rules generated by the global structure. The complete set of rules is available in Appendix A.



@ Class	A2900 = (-∞ - 1,808.75) & A2002 = [2.75 - ∞)	A2900 = (-∞ - 1,808.75) & A4474 = [695.75 - ∞) & A1627 = (-∞ - 717.00)	…
Survive = 1	17	0	…
Died = 2	0	52	…

Rule 1: ((A2900 = Negative Infinity..1,808.750) (A2002 = 2.750..Infinity)) ==> (@Class = 1)
CF=0.947, Av.Cost=1.0, CF/Cost=0.947, P=0.0, TP=17, FP=0, Pos=17, Neg=52

Rule 2: ((A2900 = 1,808.750..Infinity) (A4474 = 695.750..Infinity) (A1627 = Negative Infinity..717.000)) ==> (@Class = 2)
CF=0.981, Av.Cost=1.0, CF/Cost=0.981, P=0.0, TP=52, FP=0, Pos=52, Neg=17

•••

Figure 4-2. The Bayesian local structure decision graph learned, the parameterization, and the rule generated by the BGRF when combined with EBD discretization. The top is the Bayesian local structure constructed, the table in the middle represents part of the CPT for the target variable @class, and the rule list below the table represents some of the 9 rules generated by the local structure. The complete set of rules is available in Appendix A.

5.0 EVALUATION OF THE BAYESIAN RULE GENERATION FRAMEWORK

For evaluating the BRGF, we will take the following approach to experimental design. Since there are many possible combinations between non-probabilistic, Bayesian global structure, and Bayesian local structure each with different search paradigms, we will first focus on comparing the two most general methods for generating a BN for rule generation: Bayesian local structure decision tree and Bayesian local structure decision graph. We will compare these two algorithms across the multiple search strategies defined within Section 3.1.4 in Section 5.1. Upon experimentation, whichever method (structure type and search method) is shown to be statistically significant (or both if they are equivalent) will be selected for comparison against the Bayesian global structure, which is viewed as a very specific case of local structure as explained in Section 2.2.4. These comparisons will be discussed in Section 5.2. We will then compare in Section 5.3 each combination of the local structure and search paradigm, global structure and search paradigm that prove to be the highest performing to two non-probabilistic rule learning methods: C4.5 and RL. Once a pattern for which type of rule generation produces the highest performance has been established in Section 5.4, we will test it on 4 proteomic datasets that are not publicly available to determine if the pattern observed on the publicly available biomedical genomic and proteomic data is still apparent.

5.1 BAYESIAN LOCAL STRUCTURE COMPARISON DECISION TREE VS. DECISION GRAPH

5.1.1 MDLPC

5.1.1.1 Results

Table 5-1 through Table 5-3 report the means of the accuracy, the balanced accuracy, and the RCI respectively for using Fayyad and Irani's MDLPC discretization method on the four different Bayesian local structure rule learning algorithms: Bayesian Local Structure Rule Learning – Greedy Decision Tree (BLSRL_DT_G), Bayesian Local Structure Rule Learning – Parallel Greedy Decision Tree (BLSRL_DT_PG), Bayesian Local Structure Rule Learning – Greedy Decision Graph (BLSRL_DG_G), Bayesian Local Structure Rule Learning – Parallel Greedy Decision Graph (BLSRL_DG_PG). In each table, each row is a dataset, and each column is a learning method. The last row in each table represents the overall mean of the specified performance measure across the datasets.

In Table 5-1, we see that BLSRL_DG_PG has greater accuracy than BLSRL_DG_G on 13 datasets. Ten of those thirteen datasets show highest accuracy (among local structure methods) with the combination of parallel greedy search method with decision graph. BLSRL_DG_G achieves higher accuracy than BLSRL_DG_PG on eight datasets. On three of those eight datasets, BLSRL_DG_G produces the highest accuracy out of all tested Bayesian local structure methods. For Bayesian local structure decision trees, BLSRL_DT_PG achieves higher accuracy than BLSRL_DT_G on ten datasets; in 3 cases among those, BLS_DT_PG achieves the largest value across all tested Bayesian local structure methods. In thirteen datasets,

BLSRL_DT_G achieves greater accuracy than BLSRL_DT_PG with 6 of them having the greatest performance across all tested Bayesian local structure methods.

In Table 5-2, which shows the performance results with respect to balanced accuracy, we see that BLSRL_DG_PG achieves greater BACC than BLSRL_DG_G on 14 datasets; in 9 of those 14 datasets the parallel greedy search method for decision graph produces the highest accuracy out of all local structure methods. BLSRL_DG_G is greater than BLSRL_DG_PG on 7 datasets. On 4 of those 7 datasets, BLSRL_DG_G produces the highest accuracy out of all tested Bayesian local structure methods. There were three ties for the BLSRL_DG algorithms. For Bayesian local structure decision trees, BLSRL_DT_PG has higher BACC values than BLSRL_DT_G on 12 datasets; in 2 of those cases, BLS_DT_PG has the largest value across all tested Bayesian local structure methods. In 11 datasets BLSRL_DT_G achieves higher BACC than BLSRL_DT_PG, with 6 of them having the greatest performance across all tested Bayesian local structure methods. There was one tie between the BLSRL_DT algorithms.

In Table 5-3, which shows the performance results with respect to Relative Classifier Information (RCI), we see that BLSRL_DG_PG is greater than BLSRL_DG_G on 15 datasets; in 14 of those 15 datasets, the parallel greedy search method for decision graph produces the highest accuracy out of all local structure methods. BLSRL_DG_G is greater than BLSRL_DG_PG on 6 datasets, among which BLSRL_DG_G produces the highest accuracy on 1 dataset out of all tested Bayesian local structure methods. For Bayesian local structure decision trees, BLSRL_DT_PG has higher values than BLSRL_DT_G on 10 datasets; in 3 of those cases, BLS_DT_PG has the largest value across all tested Bayesian local structure methods. BLSRL_DT_G has 12 datasets in which it achieves greater RCI than BLSRL_DT_PG, with 2 of them having the greatest performance across all tested Bayesian local structure methods.

Table 5-4 through Table 5-6 report the results from pair-wise comparisons of the performance measures of the Bayesian local structure rule learning algorithms using MDLPC on the biomedical datasets. These tables show the statistical significance of the observed differences in the measures. Table 5-4 reports results from the analysis of accuracy compared in three tests, BLSRL_DG_G vs. BLSRL_DG_PG, BLSRL_DT_G vs. BLSRL_DT_PG, and since BLSRL_DG_PG and BLSRL_DT_G have higher average compared to the alternative search strategy within the same local structure, we compared them. Table 5-5 reports results from the analysis of balanced accuracy comparing the same tests as in Table 5-4. Table 5-6 reports results from the analysis of Relative Classifier Information (RCI) using the same comparisons as in Table 5-4 and Table 5-5.

We noticed that when comparing these algorithms using MDLPC as the discretization method, two measures, accuracy and balanced accuracy show no significant difference between any of the comparisons, while Relative Classifier Information (RCI) shows significant difference between two of the comparisons: BLSRL_DG_G vs. BLSRL_DG_PG and BLSRL_DG_PG vs. BLSRL_DT_G. The comparison of BLSRL_DT_G to BLSRL_DT_PG shows no significance difference in any of the measures.

5.1.1.2 Discussion

Overall, the BLSRL_DG_PG algorithm increased performance on average across the various measures, though only RCI had statistical significance. This could be due to many reasons. The lack of statistical difference in accuracy can be explained when considering the highly skewed nature of the data. While measures like RCI and balanced accuracy attempt to compensate for skewed distributions among classes, accuracy reflects total correct or incorrect. This means that if there are a large number of samples from one class, equivalent accuracy can be achieved in

many ways. Balanced accuracy showed that considering multi-class problems as one vs. rest for performance evaluation can be problematic. We see that even though BLSRL_DG_PG achieves greater BACC than BLSRL_DG_G 14 out of 24 times with 9 of them being the greatest out of all Bayesian local structure algorithms tested with MDLPC, the one vs. rest comparisons leads to difficultly in showing statistical difference between the classifiers. RCI, however, considers the classification problem as a whole where the distribution of the predictions in the classifier is considered against the predictions a majority classifier calculating the information gain when using the classifier over the majority class prediction. This results in a very sensitive measure that allows distinction between classification methods. Using this measure, BLSRL_DG_PG was shown to be statistically significant and greater than the decision tree search methods and BLSRL_DG_G. This means that the most amount of information is gained over majority classification when using BLSRL_DG_PG when using MDLPC as the discretization method.

Table 5-1. The percent accuracy averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers. Those values which are greater within the same local structure type are in bold and the largest value of all structures is italicized.

Bayesian Local Structure	*Decision Tree*		*Decision Graph*	
Search Strategy	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	**81.43**	61.07	70.18	***84.46***
Beer et al.	78.06	**79.17**	***81.81***	78.19
Bhattacharjee et al.	50.10	**52.98**	***72.01***	71.98
Bhattacharjee et al.	65.48	**70.00**	62.62	***72.86***
Golub et al.	***69.29***	55.36	63.57	**67.86**
Hedenfalk et al.	**97.50**	90.00	97.50	97.50
Iizuka et al.	**63.33**	60.00	58.33	***65.00***
Khan et al.	**70.72**	60.15	69.29	***71.10***
Nutt et al.	***62.00***	54.00	**54.00**	52.00
Pomeroy et al.	67.78	***72.22***	**65.56**	60.00
Pomeroy et al.	50.00	**55.00**	55.00	***56.67***
Ramaswamy et al.	42.50	**46.81**	45.00	***51.97***
Rosenwald et al.	**60.83**	60.00	61.25	***65.00***
Staunton et al.	***31.67***	25.00	20.00	**28.33**
Shipp et al.	***87.50***	83.57	**86.07**	81.07
Singh et al.	**83.18**	74.45	80.27	***84.18***
Su et al.	32.27	**52.98**	***59.34***	57.89
Veer et al.	**74.58**	73.98	84.92	***87.92***
Welsch et al.	87.50	***90.00***	87.50	87.50
Yeoh et al.	***74.40***	74.37	**68.72**	66.67
Petricoin et al.	69.23	***69.55***	60.50	**67.20**
Pusztai et al.	46.57	**46.77**	59.07	***59.56***
Ranganathan et al.	***67.27***	66.91	**65.27**	61.64
Average	**67.22**	65.60	67.82	***69.86***

Table 5-2. The percent balanced accuracy averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers. Those values which are greater within the same local structure type are in bold and the largest value of all structures is italicized.

Bayesian Local Structure	*Decision Tree*		*Decision Graph*	
Search Strategy	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	**85.61**	69.63	79.44	***87.01***
Beer et al.	51.21	**52.21**	***54.03***	51.65
Bhattacharjee et al.	**54.84**	53.80	58.15	***59.20***
Bhattacharjee et al.	43.39	**46.19**	42.26	***49.94***
Golub et al.	***63.96***	55.36	61.38	**62.46**
Hedenfalk et al.	**97.50**	90.00	97.50	97.50
Iizuka et al.	**51.67**	47.08	49.17	**51.67**
Khan et al.	71.17	**72.25**	***73.14***	71.63
Nutt et al.	***72.15***	67.54	**67.21**	64.50
Pomeroy et al.	54.54	**56.59**	53.71	***55.41***
Pomeroy et al.	40.00	**45.00**	44.17	***46.67***
Ramaswamy et al.	***56.24***	55.93	53.67	**54.89**
Rosenwald et al.	57.50	***57.57***	52.68	**56.25**
Staunton et al.	**56.28**	53.39	51.38	***56.73***
Shipp et al.	***62.71***	59.29	***61.79***	56.37
Singh et al.	***83.33***	73.58	80.50	***84.25***
Su et al.	55.74	**64.47**	64.85	***66.78***
Veer et al.	70.65	***73.60***	85.00	***86.34***
Welsch et al.	48.75	***50.00***	48.75	48.75
Yeoh et al.	46.34	**47.34**	***47.85***	43.26
Petricoin et al.	53.28	**54.98**	48.75	***55.88***
Pusztai et al.	66.93	**67.23**	***68.95***	68.45
Ranganathan et al.	***69.25***	65.17	**67.25**	63.92
Average	**63.04**	61.59	62.98	***64.15***

Table 5-3. The percent RCI averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers. Those values which are greater within the same local structure type are in bold and the largest value of all structures is italicized.

Bayesian Local Structure	*Decision Tree*		*Decision Graph*	
Search Strategy	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	**32.24**	25.97	33.73	***57.74***
Beer et al.	2.83	**6.40**	2.66	***7.16***
Bhattacharjee et al.	**57.98**	57.88	58.98	***59.12***
Bhattacharjee et al.	**0.88**	0.58	0.17	***1.69***
Golub et al.	***36.79***	17.36	**35.82**	34.62
Hedenfalk et al.	72.10	72.10	72.10	72.10
Iizuka et al.	0.53	**1.09**	**0.92**	0.83
Khan et al.	**32.00**	18.35	29.35	***36.56***
Nutt et al.	30.14	***33.25***	29.24	**31.78**
Pomeroy et al.	**22.89**	20.87	22.94	***25.99***
Pomeroy et al.	**1.66**	1.40	***1.73***	1.22
Ramaswamy et al.	29.21	**33.98**	49.53	***52.99***
Rosenwald et al.	14.14	**15.98**	18.09	***27.74***
Staunton et al.	**32.69**	30.07	32.16	***33.43***
Shipp et al.	**24.33**	17.47	24.98	***25.31***
Singh et al.	***34.71***	17.68	**30.76**	29.97
Su et al.	**60.27**	59.67	60.84	***66.99***
Veer et al.	**19.68**	18.91	42.94	***48.94***
Welsch et al.	23.27	***27.89***	23.27	23.27
Yeoh et al.	7.12	**7.89**	**7.79**	7.26
Petricoin et al.	1.70	**2.65**	2.06	***5.89***
Pusztai et al.	24.02	**27.02**	25.94	***28.11***
Ranganathan et al.	5.25	***6.89***	**5.57**	5.50
Average	**27.77**	25.89	29.65	***32.68***

Table 5-4. The comparison of accuracies of the different search strategies and local structure type using MDLPC discretization. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
DG_G Vs DG_PG	67.82 (17.27) 69.86 (16.10)	-2.03	0.079 (-1.755)	0.058 (-1.995)
DT_G Vs DT_PG	67.22 (18.31) 65.60 (16.58)	1.62	0.330 (0.973)	0.329 (0.997)
DG_PG Vs DT_G	69.86 (16.10) 67.22 (18.31)	-2.64	0.289 (-1.060)	0.162 (-1.444)

Table 5-5. The balanced accuracy comparison of the different search strategies and the local structure type using MDLPC discretization. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
DG_G Vs DG_PG	62.98 (16.06) 64.15 (15.92)	-1.16	0.159 (1.408)	0.123 (-1.601)
DT_G Vs DT_PG	63.04 (15.90) 61.59 (13.63)	1.45	0.378 (0.882)	0.191 (1.348)
DG_PG Vs DT_G	64.15 (15.92) 63.04 (15.90)	-1.10	0.370 (-0896)	0.301 (1.057)

Table 5-6. The Statistical comparison of RCI of the different search strategies and local structure type using MDLPC discretization. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
DG_G Vs DG_PG	29.65 (25.10) 32.68 (25.75)	-3.03	**0.003** (-3.007)	**0.010** (-2.790)
DT_G Vs DT_PG	27.77 (24.72) 25.89 (24.52)	1.88	0.649 (0.455)	0.166 (1.431)
DG_PG Vs DT_G	32.68 (25.75) 27.77 (24.72)	4.91	**0.003** (2.972)	**0.013** (2.687)

5.1.2 EBD

5.1.2.1 Results

Table 5-7 through Table 5-9 report the means of the accuracy, the balanced accuracy, and the RCI respectively for using Efficient Bayesian Discretization (EBD) method on the four different Bayesian local structure rule learning algorithms: Bayesian Local Structure Rule Learning – Greedy Decision Tree (BLSRL_DT_G), Bayesian Local Structure Rule Learning – Parallel Greedy Decision Tree (BLSRL_DT_PG), Bayesian Local Structure Rule Learning – Greedy Decision Graph (BLSRL_DG_G), Bayesian Local Structure Rule Learning – Parallel Greedy Decision Graph (BLSRL_DG_PG). In each table, each row is a dataset, and each column is a learning method. The last row in each table represents the overall mean of the specified performance measure across the datasets.

In Table 5-7, we see that BLSRL_DG_PG has greater accuracy than BLSRL_DG_G on 12 datasets. Ten of those twelve datasets show highest accuracy (among local structure methods) with the combination of parallel greedy search method with decision graph. BLSRL_DG_G achieves higher accuracy than BLSRL_DG_PG on six datasets. On two of those six datasets, BLSRL_DG_G produces the highest accuracy out of all tested Bayesian local structure methods. There were six ties between BLSRL_DG_PG and BLSRL_DG_G. For Bayesian local structure decision trees, BLSRL_DT_PG achieves higher accuracy than BLSRL_DT_G on eight datasets; in 3 cases among those, BLS_DT_PG achieves the largest value across all tested Bayesian local structure methods. In twelve datasets, BLSRL_DT_G achieves greater accuracy than BLSRL_DT_PG with 5 of them having the greatest performance across all tested Bayesian local structure methods.

In Table 5-8, which shows the performance results with respect to BACC, we see that BLSRL_DG_PG achieves greater BACC than BLSRL_DG_G on 18 datasets; in all of those 18 datasets, the parallel greedy search method for decision graph produces the highest BACC out of all local structure methods. BLSRL_DG_G is greater than BLSRL_DG_PG on 2 datasets. BLSRL_DG_G produces the highest accuracy on none of the datasets out of all tested Bayesian local structure methods. There were four ties for the BLSRL_DG algorithms. For Bayesian local structure decision trees, BLSRL_DT_PG has higher BACC values than BLSRL_DT_G on 7 datasets; in 1 of those cases, BLS_DT_PG has the largest value across all tested Bayesian local structure methods. In 10 datasets BLSRL_DT_G achieves higher BACC than BLSRL_DT_PG, with 3 of them having the greatest performance across all tested Bayesian local structure methods. There were seven ties between the BLSRL_DT algorithms

In Table 5-9, which shows the performance results with respect to RCI, we see that BLSRL_DG_PG is greater than BLSRL_DG_G on 18 datasets; in 16 of those 18 datasets, the parallel greedy search method for decision graph produces the highest accuracy out of all local structure methods. BLSRL_DG_G is greater than BLSRL_DG_PG on 2 datasets, among which BLSRL_DG_G produced the highest accuracy out of all tested Bayesian local structure methods. There were four ties between BLSRL_DG_G and BLSRL_DG_PG. For Bayesian local structure decision trees, BLSRL_DT_PG has higher values than BLSRL_DT_G on 14 datasets; in 3 of those cases, BLS_DT_PG has the largest value across all tested Bayesian local structure methods. BLSRL_DT_G has 7 datasets in which it achieves greater RCI than BLSRL_DT_PG, with 1 of them having the greatest performance across all tested Bayesian local structure methods. There were three ties between BLSRL_DT_G and BLSRL_DT_PG.

Table 5-10 through Table 5-12 report the results from pair-wise comparisons of the performance measures of the Bayesian local structure rule learning algorithms using EBD on the biomedical datasets that is aimed at measuring the statistical significance of the observed differences in the measures. Table 5-10 reports results from the analysis of accuracy compared in three tests, BLSRL_DG_G vs. BLSRL_DG_PG, BLSRL_DT_G vs. BLSRL_DT_PG, and since BLSRL_DG_PG and BLSRL_DT_G has the higher average compared to the alternative search strategy within the same local structure, we compared them. Table 5-11 reports results from the analysis of balanced accuracy comparing the same tests as in Table 5-10. Table 5-12 reports results from the analysis of RCI using the same comparisons as in Table 5-10 and Table 5-11.

When comparing these algorithms using EBD as the discretization method, one measure, accuracy, showed no significant difference between any of the comparisons, while BACC and RCI show significant difference between two of the comparisons: BLSRL_DG_G vs. BLSRL_DG_PG and BLSRL_DG_PG vs. BLSRL_DT_G. The comparison of BLSRL_DT_G to BLSRL_DT_PG shows no significant difference using any of the measures.

5.1.2.2 Discussion

Overall, the BLSRL_DG_PG algorithm shows improved performance on average across all performance measures with BACC and RCI having statistically significant differences. The lack of statistical difference in accuracy can be explained when considering the highly skewed nature of the data. While the number of correct predictions is equivalent across the datasets (as seen by no statistical difference in the methods), as we see with BACC and RCI, the distributions of the predictions are not. BACC and RCI take sample distribution of the datasets into account when measuring the performance of the classifier. BACC has the approach of one vs. rest when calculating sensitivity and specificity while RCI measures the distributions of the predictions of

the classifier compared to the predictions a majority classifier would make. Both these performance measures show significant difference, illustrating that with EBD, BLSRL_DG_PG can predict the same number correct samples, but trades the correct predictions for the majority class with the correct predictions for the sparse classes.

Table 5-7. The percent accuracy averaged over 10x10 fold cross-validation for EBD discretization and Bayesian local structure classifiers. Those values which are greater within the same local structure type are in bold and the largest value of all structures is italicized.

Bayesian Local Structure	*Decision Tree*		*Decision Graph*	
Search Strategy	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	***78.93***	52.86	76.25	76.25
Beer et al.	73.06	***76.53***	72.08	**72.22**
Bhattacharjee et al.	53.44	**56.68**	76.37	***78.98***
Bhattacharjee et al.	***74.29***	64.29	**74.05**	64.29
Golub et al.	***70.89***	48.39	69.46	69.46
Hedenfalk et al.	97.50	97.50	97.50	97.50
Iizuka et al.	55.00	**58.33**	***55.00***	51.67
Khan et al.	**61.81**	54.90	61.54	***64.25***
Nutt et al.	**58.00**	52.00	50.00	**58.00**
Pomeroy et al.	74.44	**77.78**	78.89	***81.11***
Pomeroy et al.	***60.00***	58.33	58.33	58.33
Ramaswamy et al.	**48.57**	47.49	50.36	***52.55***
Rosenwald et al.	**59.17**	57.50	63.75	***69.17***
Staunton et al.	**36.67**	31.67	40.67	***45.00***
Shipp et al.	*84.82*	*84.82*	**82.14**	80.54
Singh et al.	***83.09***	70.73	**80.27**	76.09
Su et al.	25.92	**50.92**	65.93	***67.92***
Veer et al.	**77.08**	77.00	82.17	***89.17***
Welsch et al.	87.50	***90.00***	87.50	87.50
Yeoh et al.	74.77	***76.78***	**72.31**	70.72
Petricoin et al.	69.40	70.03	70.20	***79.99***
Pusztai et al.	47.18	**47.91**	68.99	***71.32***
Ranganathan et al.	**63.64**	63.27	***67.64***	65.64
Average	**67.30**	65.24	70.89	***71.99***

Table 5-8. The percent BACC averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers. Those values which are greater within the same local structure type are in bold and the largest value of all structures is italicized.

Bayesian Local Structure	*Decision Tree*		*Decision Graph*	
Search Strategy	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	**83.25**	63.46	79.62	***89.62***
Beer et al.	52.90	**53.00**	58.67	***53.54***
Bhattacharjee et al.	54.92	56.32	59.63	***65.32***
Bhattacharjee et al.	***52.14***	47.66	**52.02**	48.24
Golub et al.	***65.85***	55.68	65.00	65.00
Hedenfalk et al.	97.50	97.50	97.50	97.50
Iizuka et al.	43.33	**48.33**	47.08	***52.50***
Khan et al.	**67.62**	61.12	77.41	***78.29***
Nutt et al.	**74.56**	67.64	68.35	***69.29***
Pomeroy et al.	**54.58**	54.42	52.17	***59.58***
Pomeroy et al.	**48.33**	47.50	46.67	***47.83***
Ramaswamy et al.	51.65	**54.89**	52.20	***64.81***
Rosenwald et al.	56.46	**56.74**	55.54	***59.92***
Staunton et al.	49.65	**54.97**	50.06	***59.44***
Shipp et al.	***63.12***	59.54	**59.20**	57.98
Singh et al.	**83.25**	74.50	80.67	***88.00***
Su et al.	54.27	**62.13**	67.40	***73.95***
Veer et al.	**75.24**	74.95	86.32	***89.32***
Welsch et al.	48.75	***50.00***	48.75	48.75
Yeoh et al.	47.08	48.12	47.94	***49.42***
Petricoin et al.	55.12	54.92	49.98	***64.99***
Pusztai et al.	67.21	68.23	74.66	***76.50***
Ranganathan et al.	64.83	**66.33**	68.83	***69.58***
Average	**62.98**	61.58	64.40	***67.89***

Table 5-9. The percent RCI averaged over 10x10 fold cross-validation for MDLPC discretization and Bayesian local structure classifiers. Those values which are greater within the same local structure type are in bold and the largest value of all structures is italicized.

Bayesian Local Structure	*Decision Tree*		*Decision Graph*	
Search Strategy	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	52.57	22.26	58.10	***61.55***
Beer et al.	2.10	***5.51***	0.73	3.12
Bhattacharjee et al.	56.99	**59.00**	61.30	***64.99***
Bhattacharjee et al.	**2.11**	0.40	***4.52***	3.70
Golub et al.	***39.78***	10.75	37.54	**39.54**
Hedenfalk et al.	72.10	72.10	72.10	72.10
Iizuka et al.	0.55	***2.17***	1.19	**1.52**
Khan et al.	**24.17**	12.56	28.00	***39.00***
Nutt et al.	**30.22**	25.52	29.49	***35.06***
Pomeroy et al.	24.73	**26.22**	22.83	***35.91***
Pomeroy et al.	**0.90**	0.73	1.20	***1.52***
Ramaswamy et al.	31.22	**35.99**	53.39	***61.99***
Rosenwald et al.	13.99	**15.80**	34.65	***37.01***
Staunton et al.	27.74	**33.29**	29.02	***41.58***
Shipp et al.	**25.93**	13.49	23.60	***27.63***
Singh et al.	**25.70**	15.27	***27.94***	27.54
Su et al.	55.49	**63.65**	64.03	***68.40***
Veer et al.	28.16	**32.29**	42.05	***52.05***
Welsch et al.	23.27	***27.89***	23.27	23.27
Yeoh et al.	7.07	**8.32**	8.24	***8.33***
Petricoin et al.	1.16	**2.47**	2.83	***9.65***
Pusztai et al.	28.56	**29.56**	29.62	***33.79***
Ranganathan et al.	3.86	**4.57**	5.77	***7.02***
Average	**28.27**	25.83	31.73	***35.68***

Table 5-10. The comparison of accuracies of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
DG_G Vs DG_PG	70.89 (14.25) 71.99 (13.78)	-1.09	0.122 (-1.546)	0.208 (-1.295)
DT_G Vs DT_PG	67.30 (17.91) 65.24 (17.14)	2.06	0.375 (0.866)	0.305 (1.048)
DG_PG Vs DT_G	71.99 (13.78) 67.30 (17.91)	4.69	0.131 (1.512)	0.063 (1.954)

Table 5-11. The comparison of BACC of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
DG_G Vs DG_PG	64.40 (15.90) 67.89 (15.80)	-3.49	**0.003** (-2.987)	**0.002** (-3.423)
DT_G Vs DT_PG	62.98 (15.63) 61.58 (13.91)	1.40	0.638 (0.471)	0.251 (1.178)
DG_PG Vs DT_G	67.89 (15.80) 62.98 (15.63)	4.91	**0.003** (2.937)	**0.001** (3.710)

Table 5-12. The comparison of RCI of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
DG_G Vs DG_PG	31.73 (25.88) 35.68 (26.41)	-3.95	**<0.001** (-3.702)	**<0.001** (-4.518)
DT_G Vs DT_PG	28.27 (24.94) 25.83 (25.30)	2.44	0.910 (0.114)	0.237 (1.214)
DG_PG Vs DT_G	35.68 (26.41) 28.27 (24.94)	7.41	**<0.001** (3.980)	**<0.001** (4.197)

5.1.3 Discretization Comparison

5.1.3.1 Results

Table 5-13 through Table 5-15 report the means of the accuracy, the balanced accuracy, and the RCI respectively comparing Efficient Bayesian Discretization (EBD) method and MDLPC discretization method on the four different Bayesian local structure rule learning algorithms: Bayesian Local Structure Rule Learning – Greedy Decision Tree (BLSRL_DT_G), Bayesian Local Structure Rule Learning – Parallel Greedy Decision Tree (BLSRL_DT_PG), Bayesian Local Structure Rule Learning – Greedy Decision Graph (BLSRL_DG_G), Bayesian Local Structure Rule Learning – Parallel Greedy Decision Graph (BLSRL_DG_PG). In each table, each row is a dataset, and each column is a learning method. The last row in each table represents the overall mean of the specified performance measure across the datasets.

In Table 5-13, which shows the results using the measure of accuracy, we see within the BLSRL_DG_PG algorithm EBD discretization is greater than MDLPC 14 times with 9 of those times having the highest accuracy out of all Bayesian local structure algorithms and discretization combinations. MDLPC combined with BLSRL_DG_PG is greater than EBD with BLSRL_DG_PG 7 times with 4 of those times having the highest accuracy out of all methods. There are 3 ties between EBD and MDLPC when using BLSRL_DG_PG. BLSRL_DG_G algorithm using EBD is greater than using MDLPC a total of 14 times with 1 of those having the highest accuracy across the different methods and discretizations. MDLPC and BLSRL_DG_G combined has a greater accuracy than BLSRL_DG_G with EBD a total of 6 times with 1 of them having the highest overall accuracy. EBD and MDLPC tie 4 times when using the BLSRL_DG_G algorithm. BLSRL_DT_PG algorithm EBD discretization is greater than MDLPC 11 times with none of those times having the highest accuracy out of all Bayesian local

structure algorithms and discretization combinations. MDLPC combined with BLSRL_DT_PG is greater than EBD with BLSRL_DG_PG 11 times with none of those times having the highest accuracy out of all methods. There are 2 ties between EBD and MDLPC when using BLSRL_DT_PG. BLSRL_DT_G algorithm using EBD is greater than using MDLPC a total of 10 times with 3 of those having the highest accuracy across the different methods and discretizations. MDLPC and BLSRL_DG_G combined has a greater accuracy than BLSRL_DG_G with EBD a total of 10 times with 2 of them having the highest overall accuracy. EBD and MDLPC tie 4 times when using the BLSRL_DT_G algorithm.

In Table 5-14, which shows the results using the measure of BACC, we see that BLSRL_DG_PG algorithm EBD discretization is greater than MDLPC 20 times with 18 of those times having the highest accuracy out of all Bayesian local structure algorithms and discretization combinations. MDLPC combined with BLSRL_DG_PG is greater than EBD with BLSRL_DG_PG once. There are 3 ties between EBD and MDLPC when using BLSRL_DG_PG. BLSRL_DG_G algorithm using EBD is greater than using MDLPC a total of 16 times with none of those having the highest accuracy across the different methods and discretizations. MDLPC and BLSRL_DG_G combined has a greater accuracy than BLSRL_DG_G with EBD a total of 5 times with none of them having the highest overall accuracy. EBD and MDLPC tie 3 times when using the BLSRL_DG_G algorithm. BLSRL_DT_PG algorithm EBD discretization is greater than MDLPC 15 times with none of those times having the highest accuracy out of all Bayesian local structure algorithms and discretization combinations. MDLPC combined with BLSRL_DT_PG is greater than EBD with BLSRL_DG_PG 7 times with none of those times having the highest accuracy out of all methods. There are 2 ties between EBD and MDLPC when using BLSRL_DT_PG.

BLSRL_DT_G algorithm using EBD is greater than BLSRL_DT_G using MDLPC a total of 13 times with 4 of those having the highest accuracy across the different methods and discretizations. MDLPC and BLSRL_DG_G combined has a greater accuracy than BLSRL_DG_G with EBD a total of 8 times with none of them having the highest overall accuracy. EBD and MDLPC tie 3 times when using the BLSRL_DT_G algorithm.

In Table 5-15, which shows the results using the measure of RCI, we see that BLSRL_DG_PG algorithm EBD discretization is greater than MDLPC 19 times with 15 of those times having the highest accuracy out of all Bayesian local structure algorithms and discretization combinations. MDLPC combined with BLSRL_DG_PG is greater than EBD with BLSRL_DG_PG 2 times with 1 of them having the highest accuracy across the discretization methods and classifiers. There are 3 ties between EBD and MDLPC when using BLSRL_DG_PG. BLSRL_DG_G algorithm using EBD is greater than using MDLPC a total of 13 times with 1 of those having the highest accuracy across the different methods and discretizations. MDLPC and BLSRL_DG_G combined has a greater accuracy than BLSRL_DG_G with EBD a total of 8 times with 1 of them having the highest overall accuracy. EBD and MDLPC tie 3 times when using the BLSRL_DG_G algorithm. BLSRL_DT_PG algorithm EBD discretization is greater than MDLPC 9 times with none of those times having the highest accuracy out of all Bayesian local structure algorithms and discretization combinations. MDLPC combined with BLSRL_DT_PG is greater than EBD with BLSRL_DG_PG 12 times with none of those times having the highest accuracy out of all methods. There are 3 ties between EBD and MDLPC when using BLSRL_DT_PG. BLSRL_DT_G algorithm using EBD is greater than BLSRL_DT_G using MDLPC a total of 10 times with 1 of those having the highest accuracy across the different methods and

discretizations. MDLPC and BLSRL_DG_G combined has a greater accuracy than BLSRL_DG_G with EBD a total of 11 times with none of them having the highest overall accuracy. EBD and MDLPC tie 3 times when using the BLSRL_DT_G algorithm.

Table 5-16 through Table 5-18 report the results from pair-wise comparisons of the performance measures of the Bayesian local structure rule learning algorithms using EBD on the biomedical datasets that is aimed at measuring the statistical significance of the observed differences in the measures. Table 5-16 reports results from the statistical significance of the analysis of accuracy compared in four tests, BLSRL_DG_PG algorithm with EBD vs. BLSRL_DG_PG with MDLPC, BLSRL_DG_G with EBD vs. BLSRL_DG_G with MDLPC, BLSRL_DT_PG algorithm with EBD vs. BLSRL_DT_PG with MDLPC, BLSRL_DT_G with EBD vs. BLSRL_DT_G with MDLPC. Table 5-17 reports results from the analysis of balanced accuracy comparing the same tests as in Table 5-16. Table 5-18 reports results from the analysis of Relative Classifier Information (RCI) using the same comparisons as in Table 5-16 and Table 5-17. Figure 5-1 to Figure 5-3, show the overall average across all datasets of the 4 different Bayesian local structure search algorithms with SEM.

5.1.3.2 Discussion

We noticed a definitive interaction between classifier and discretization method when comparing the algorithms using EBD to the algorithms using MDLPC. For Parallel Greedy search we noticed that using EBD provides a statistically significant advantage in both balanced accuracy and RCI. It also provides a greater magnitude of average accuracy, although it is not statistically significant. When using a greedy search with Decision graphs, two measures show statistical difference (ACC, BACC), while RCI shows no difference. For Decision Trees, there was no significant difference between the discretization methods on any measure.

As seen in the results, there is a definitive interaction between the machine learning algorithms, search strategies and discretization methods. However, it is difficult to identify the exact cause. When we look at the overall behavior of the different discretization methods in terms of the number of rules generated, the number of variables used in the model, and the time it takes to run a fold of one of the machine learning methods (see Appendix B, Table 6-7 - Table 6-9), an interesting trend appears. There is an equivalent amount of variables used as well as rules generated, however combined with the increase in performance observed, it suggests that EBD picks better variables for the rule generation. The cost, however, is that on average you nearly double the running time. This could be due to implementation (the parallel greedy algorithms do not have a completely parallel implementation of model generation), however since the pattern is observable across multiple types of search strategies, it is a reasonable assumption that the increase in the number of variables discretized by EBD creates a longer run time for the model construction.

It is difficult to evaluate how the variables interact with the model construction. However, empirically, Bayesian local structure (BLS) with decision graph produces better results in combination with EBD while BLS with decision tree structure performs equivalently with either MDLPC or EBD with no clear winner. Researchers have focused on this question detailing why it is critical that when choosing an algorithm that requires discretization. They concluded that the discretization method interacts with the learning algorithm to produce the models, and choosing the right discretization method for the learning algorithm is an important problem [44, 45, 157]. The results from these four different Bayesian local structure algorithms support their claim that the discretization method interacts with the learning algorithm.

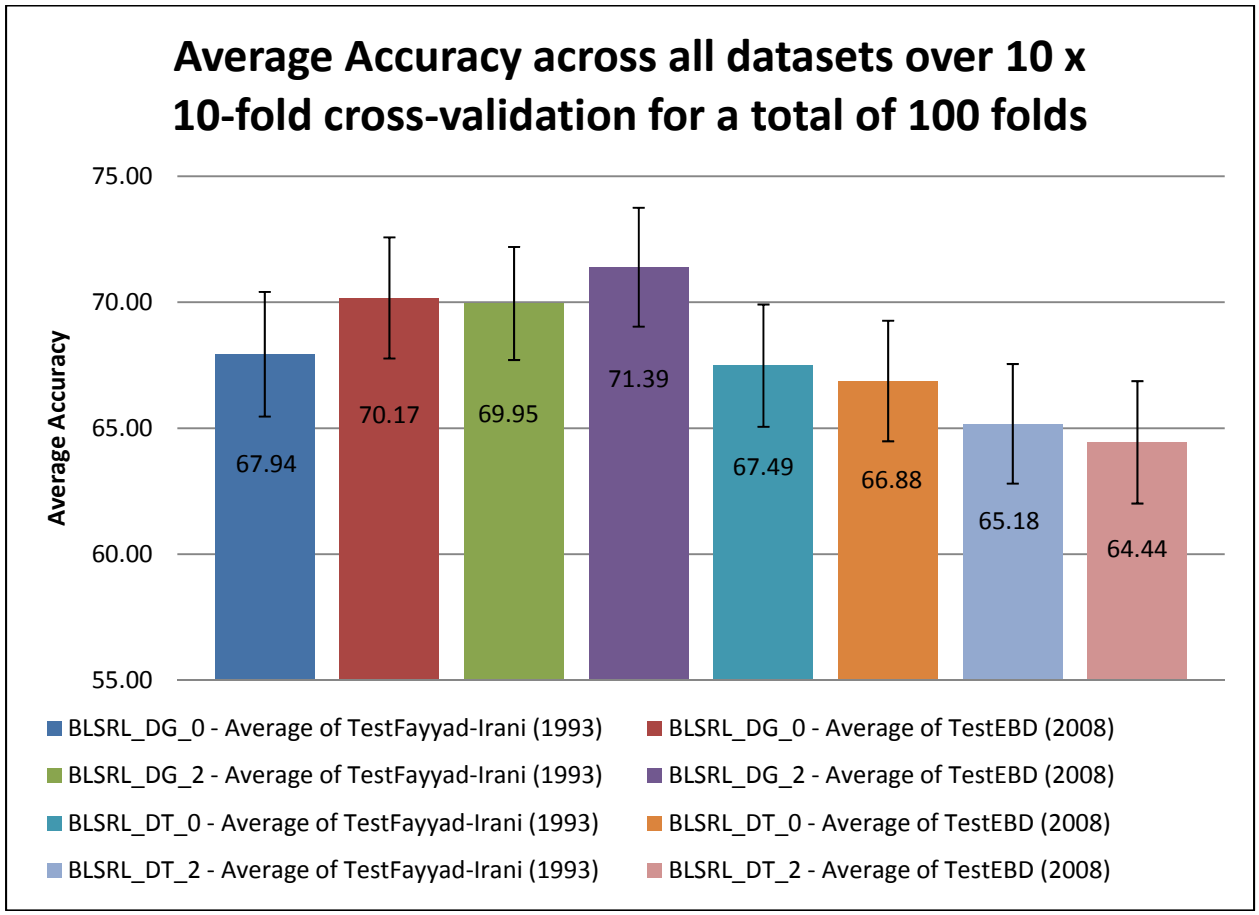


Figure 5-1. The Accuracy with standard errors for all datasets using MDLPC or EBD as the discretization techniques and the four different Bayesian local structure algorithms.

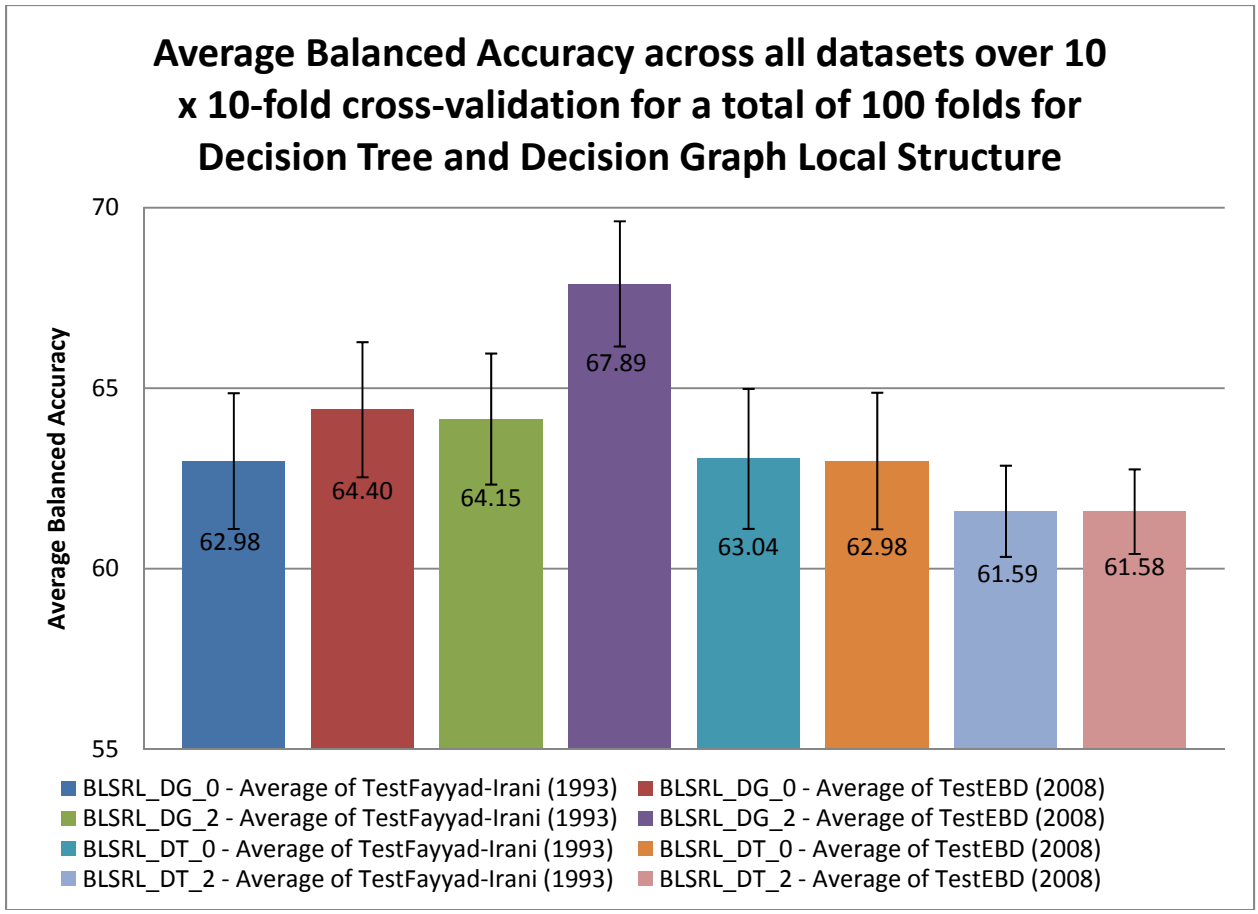


Figure 5-2. The Balanced Accuracy with standard errors for all datasets using MDLPC or EBD as the discretization techniques and the four different Bayesian local structure algorithms.

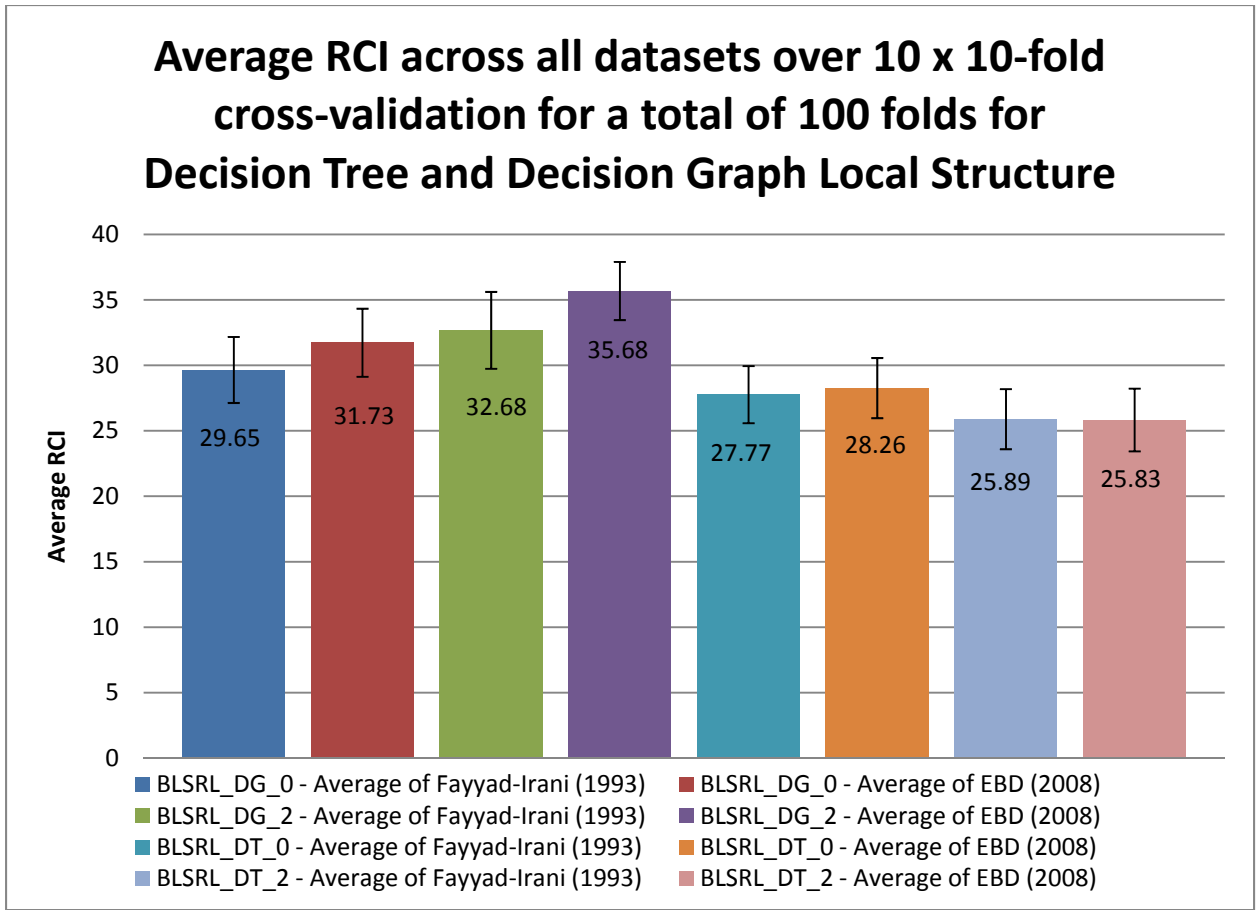


Figure 5-3. The Relative Classifier Information with standard errors for all datasets using MDLPC and EBD as the discretization technique and the four different Bayesian local structure algorithms.

Table 5-13. Comparison of percent accuracy of the Discretization Methods across Bayesian Local Structure Rule Learning (BLSRL) algorithms. The higher score of the two compared discretization methods within a BLSRL type is in bold font with the highest across all methods being italicized.

Bayesian Local Structure	*DT_G*		*DT_PG*		*DG_G*		*DG_PG*	
Discretization	EBD	MDLPC	EBD	MDLPC	EBD	MDLPC	EBD	MDLPC
Alon et al.	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	78.93	**81.43**	52.86	**61.07**	**76.25**	70.18	76.25	***84.46***
Beer et al.	73.06	**78.06**	76.53	**79.17**	72.08	***81.81***	72.22	**78.19**
Bhattacharjee et al.	**53.44**	50.10	**56.68**	52.98	**76.37**	72.01	***78.98***	71.98
Bhattacharjee et al.	***74.29***	65.48	64.29	**70.00**	**74.05**	62.62	64.29	**72.86**
Golub et al.	***70.89***	69.29	48.39	**55.36**	**69.46**	63.57	**69.46**	67.86
Hedenfalk et al.	97.50	97.50	**97.50**	90.00	97.50	97.50	97.50	97.50
Iizuka et al.	55.00	**63.33**	58.33	**60.00**	55.00	**58.33**	51.67	***65.00***
Khan et al.	61.81	**70.72**	54.90	**60.15**	61.54	**69.29**	64.25	***71.10***
Nutt et al.	58.00	***62.00***	52.00	**54.00**	50.00	**54.00**	**58.00**	52.00
Pomeroy et al.	**74.44**	67.78	**77.78**	72.22	**78.89**	65.56	***81.11***	60.00
Pomeroy et al.	***60.00***	50.00	**58.33**	55.00	**58.33**	55.00	**58.33**	56.67
Ramaswamy et al.	**48.57**	42.50	**47.49**	46.81	**50.36**	45.00	***52.55***	51.97
Rosenwald et al.	59.17	**60.83**	57.50	**60.00**	**63.75**	61.25	***69.17***	65.00
Staunton et al.	36.67	31.67	**31.67**	25.00	**40.67**	20.00	***45.00***	28.33
Shipp et al.	84.82	***87.50***	**84.82**	83.57	82.14	**86.07**	80.54	**81.07**
Singh et al.	83.09	**83.18**	70.73	**74.45**	80.27	80.27	76.09	***84.18***
Su et al.	25.92	**32.27**	50.92	**52.98**	**65.93**	59.34	***67.92***	57.89
Veer et al.	**77.08**	74.58	**77.00**	73.98	82.17	**84.92**	***89.17***	87.92
Welsch et al.	87.50	87.50	*90.00*	*90.00*	87.50	87.50	87.50	87.50
Yeoh et al.	**74.77**	74.40	***76.78***	74.37	**72.31**	68.72	**70.72**	66.67
Petricoin et al.	**69.40**	69.23	**70.03**	69.55	**70.20**	60.50	***79.99***	67.20
Pusztai et al.	**47.18**	46.57	**47.91**	46.77	**68.99**	59.07	***71.32***	59.56
Ranganathan et al.	63.64	**67.27**	63.27	**66.91**	***67.64***	65.27	**65.64**	61.64
Average	66.88	**67.49**	64.44	**65.18**	**70.17**	67.94	***71.39***	69.95

Table 5-14. Comparison of percent balanced accuracy of the Discretization Methods across Bayesian Local Structure Rule Learning (BLSRL) algorithms. The higher score of the two compared discretization methods within a BLSRL type is in bold font with the highest across all methods being italicized.

Bayesian Local Structure	*DT_G*		*DT_PG*		*DG_G*		*DG_PG*	
Discretization	EBD	MDLPC	EBD	MDLPC	EBD	MDLPC	EBD	MDLPC
Alon et al.	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	**83.25**	85.61	63.46	**69.63**	**79.62**	79.44	***89.62***	87.01
Beer et al.	**52.90**	51.21	**53.00**	52.21	**58.67**	54.03	***53.54***	51.65
Bhattacharjee et al.	**54.92**	54.84	**56.32**	53.80	**59.63**	58.15	***65.32***	59.20
Bhattacharjee et al.	***52.14***	43.39	**47.66**	46.19	**52.02**	42.26	48.24	**49.94**
Golub et al.	***65.85***	63.96	**55.68**	55.36	**65.00**	61.38	**65.00**	62.46
Hedenfalk et al.	97.50	97.50	**97.50**	90.00	97.50	97.50	97.50	97.50
Iizuka et al.	43.33	**51.67**	**48.33**	47.08	47.08	**49.17**	***52.50***	51.67
Khan et al.	67.62	**71.17**	61.12	**72.25**	**77.41**	73.14	***78.29***	71.63
Nutt et al.	**74.56**	72.15	**67.64**	67.54	**68.35**	67.21	***69.29***	64.50
Pomeroy et al.	**54.58**	54.54	54.42	**56.59**	52.17	**53.71**	***59.58***	55.41
Pomeroy et al.	***48.33***	40.00	**47.50**	45.00	**46.67**	44.17	**47.83**	46.67
Ramaswamy et al.	51.65	**56.24**	54.89	**55.93**	52.20	**53.67**	***64.81***	54.89
Rosenwald et al.	56.46	**57.50**	56.74	**57.57**	**55.54**	52.68	***59.92***	56.25
Staunton et al.	49.65	**56.28**	**54.97**	53.39	50.06	**51.38**	***59.44***	56.73
Shipp et al.	***63.12***	62.71	**59.54**	59.29	59.20	**61.79**	**57.98**	56.37
Singh et al.	83.25	**83.33**	**74.50**	73.58	**80.67**	80.50	***88.00***	84.25
Su et al.	54.27	**55.74**	62.13	**64.47**	**67.40**	64.85	***73.95***	66.78
Veer et al.	**75.24**	70.65	**74.95**	73.60	**86.32**	85.00	***89.32***	86.34
Welsch et al.	48.75	48.75	50.00	50.00	48.75	48.75	48.75	48.75
Yeoh et al.	**47.08**	46.34	**48.12**	47.34	**47.94**	47.85	***49.42***	43.26
Petricoin et al.	**55.12**	53.28	54.92	**54.98**	**49.98**	48.75	***64.99***	55.88
Pusztai et al.	**67.21**	66.93	**68.23**	67.23	**74.66**	68.95	***76.50***	68.45
Ranganathan et al.	64.83	**69.25**	**66.33**	65.17	**68.83**	67.25	***69.58***	63.92
Average	62.98	**63.04**	61.58	**61.59**	**64.40**	62.98	***67.89***	64.15

Table 5-15. Comparison of percent Relative Classifier Information (RCI) of the Discretization Methods across Bayesian Local Structure Rule Learning (BLSRL) algorithms. The higher score of the two compared discretization methods within a BLSRL type is in bold font with the highest across all methods being italicized.

Bayesian Local Structure	*DT_G*		*DT_PG*		*DG_G*		*DG_PG*	
Discretization	EBD	MDLPC	EBD	MDLPC	EBD	MDLPC	EBD	MDLPC
Alon et al.	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	**52.57**	32.24	22.26	**25.97**	**58.10**	33.73	***61.55***	57.74
Beer et al.	2.10	**2.83**	5.51	**6.40**	0.73	**2.66**	3.12	***7.16***
Bhattacharjee et al.	56.99	**57.98**	**59.00**	57.88	**61.30**	58.98	***64.99***	59.12
Bhattacharjee et al.	**2.11**	0.88	0.40	**0.58**	***4.52***	0.17	**3.70**	1.69
Golub et al.	***39.78***	36.79	10.75	**17.36**	**37.54**	35.82	**39.54**	34.62
Hedenfalk et al.	72.10	72.10	72.10	72.10	72.10	72.10	72.10	72.10
Iizuka et al.	**0.55**	0.53	***2.17***	1.09	**1.19**	0.92	**1.52**	0.83
Khan et al.	24.17	**32.00**	12.56	**18.35**	28.00	**29.35**	***39.00***	36.56
Nutt et al.	**30.22**	30.14	25.52	**33.25**	**29.49**	29.24	***35.06***	31.78
Pomeroy et al.	**24.73**	22.89	**26.22**	20.87	22.83	**22.94**	***35.91***	25.99
Pomeroy et al.	0.90	**1.66**	0.73	**1.40**	1.20	***1.73***	**1.52**	1.22
Ramaswamy et al.	**31.22**	29.21	**35.99**	33.98	**53.39**	49.53	***61.99***	52.99
Rosenwald et al.	13.99	**14.14**	15.80	**15.98**	**34.65**	18.09	***37.01***	27.74
Staunton et al.	27.74	**32.69**	**33.29**	30.07	29.02	**32.16**	***41.58***	33.43
Shipp et al.	**25.93**	24.33	13.49	**17.47**	23.60	**24.98**	***27.63***	25.31
Singh et al.	25.70	**34.71**	15.27	**17.68**	27.94	**30.76**	27.54	**29.97**
Su et al.	55.49	**60.27**	**63.65**	59.67	**64.03**	60.84	***68.40***	66.99
Veer et al.	**28.16**	19.68	**32.29**	18.91	42.05	**42.94**	***52.05***	48.94
Welsch et al.	23.27	23.27	27.89	27.89	23.27	23.27	23.27	23.27
Yeoh et al.	7.07	**7.12**	**8.32**	7.89	**8.24**	7.79	***8.33***	7.26
Petricoin et al.	1.16	**1.70**	2.47	**2.65**	**2.83**	2.06	***9.65***	5.89
Pusztai et al.	**28.56**	24.02	**29.56**	27.02	**29.62**	25.94	***33.79***	28.11
Ranganathan et al.	3.86	**5.25**	4.57	**6.89**	**5.77**	5.57	***7.02***	5.50
Average	**28.26**	27.77	25.83	**25.89**	**31.73**	29.65	***35.68***	32.68

Table 5-16. Accuracy Comparison of EBD to MDLPC Across Search Strategies and Local Structure.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
EBD_DG_G Vs MDLPC_DG_G	70.89 (14.25) 67.82 (17.27)	3.07	**0.050** (1.960)	**0.041** (2.167)
EBD_DG_PG Vs MDLPC_DG_PG	71.99 (13.78) 69.86 (16.10)	2.13	0.244 (1.164)	0.227 (1.241)
EBD_DT_G Vs MDLPC_DT_G	67.30 (17.91) 67.22 (18.31)	0.08	0.958 (0.052)	0.936 (0.082)
EBD_DT_PG Vs MDLPC_DT_PG	65.24 (17.14) 65.60 (16.58)	-0.36	0.638 (-0.471)	0.676 (-0.423)

Table 5-17. BACC Comparison EBD to MDLPC across Search Strategies and Local Structure.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
EBD_DG_G Vs MDLPC_DG_G	64.40 (15.90) 62.98 (16.06)	1.42	**0.026** (2.225)	**0.021** (2.488)
EBD_DG_PG Vs MDLPC_DG_PG	67.89 (15.80) 64.15 (15.92)	3.74	**<0.001** (3.875)	**<0.001** (5.912)
EBD_DT_G Vs MDLPC_DT_G	62.98 (15.63) 63.04 (13.25)	-0.060	0.931 (-0.087)	0.942 (-0.074)
EBD_DT_PG Vs MDLPC_DT_PG	61.58 (13.91) 61.59 (13.63)	-0.010	0.277 (-1.088)	0.988 (-0.015)

Table 5-18. RCI Comparison to MDLPC across Search Strategies and Local Structure.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
EBD_DG_G Vs MDLPC_DG_G	31.72 (25.88) 29.65 (25.10)	2.08	0.159 (1.408)	0.109 (1.666)
EBD_DG_PG Vs MDLPC_DG_PG	35.68 (26.41) 32.68 (25.75)	3.00	**0.001** (3.250)	**<0.001** (4.079)
EBD_DT_G Vs MDLPC_DT_G	28.27 (24.94) 27.77 (24.72)	0.50	0.931 (0.087)	0.666 (0.437)
EBD_DT_PG Vs MDLPC_DT_PG	25.83 (25.30) 25.89 (24.53)	-0.06	0.754 (-0.313)	0.942 (-0.073)

5.1.4 Summary of Bayesian Local Structure Experiments

In summary, we showed that there are statistically significant differences between the local structure methods, search strategies used, and the different discretization methods. Parallel greedy search Bayesian local structure decision graph rule generation produces greater performance on average across the multiple measures when combined with EBD. Since there was no significance between EBD and MDLPC when comparing either of the decision tree search strategies, there is no unequivocal recommendation. We will therefore utilize BLSRL_DG_PG as the best, most general BN learning algorithm within the BRGF.

5.2 BGRL VS BLSRL-DG

The results from Section 5.1 allow us to choose Bayesian local structure Decision Graph using parallel greedy search as the method of choice for local structure to compare to the global algorithms. Since EBD was shown to produce the largest increase in performance for the Bayesian global structure methods (see Appendix B.1), we used only EBD as the discretization method of choice. This section tests whether the most general method, BLSRL_DG_PG is truly better than the standard Bayesian global structure. That is, we compare whether context-specific independencies assist in increasing the performance on average compared to the global structure which enforces a complete decision tree local structure.

5.2.1 Results

Table 5-19 through Table 5-21 report the means of the accuracy, the balanced accuracy, and the RCI respectively for using Efficient Bayesian Discretization (EBD) method on the three different Bayesian global structure rule learning algorithms and the Bayesian local structure method: Bayesian Global Rule Learning – Greedy (BGRL_ 0), Bayesian Global Rule Learning – Beam Search (BGRL_B), Bayesian Global Rule Learning – Parallel Greedy (BGRL_ 2), and Bayesian Local Structure Rule Learning – Parallel Greedy Decision Graph (BLSRL_DG_PG). In each table, each row is a dataset, and each column is a learning method. The last row in each table represents the overall mean of the specified performance measure across the datasets.

In Table 5-19, which shows the results using the measure of accuracy, BLSRL_DG_PG is greater than the Bayesian global structure algorithms on 12 datasets. BGRL_PG is greater than the other compared methods on 2 datasets. For BGRL_B, it has higher values on 4 datasets.

BGRL_G has 2 datasets that it is greater than BGRL_B, BGRL_PG, and BLSRL_DG_PG. There were 4 datasets which had more than one classifier performing equally.

In Table 5-20, which shows the results using the measure of BACC, BLSRL_DG_PG is greater than the Bayesian global structure algorithms on 14 datasets. BGRL_PG is greater than the other compared methods on 2 datasets. For BGRL_B, it has higher values on 2 datasets. BGRL_G has 2 datasets that it is greater than BGRL_B, BGRL_PG, and BLSRL_DG_PG. There were 4 datasets which had more than one classifier performing equally.

In Table 5-21, which shows the results using the measure of RCI, BLSRL_DG_PG is greater than the Bayesian global structure algorithms on 15 datasets. BGRL_PG and BGRL_G are not greater than the other compared methods on any dataset. For BGRL_B, it has higher values on 5 datasets. There were 4 datasets which had more than one classifier performing equally.

Table 5-22 through Table 5-24 report the results from pair-wise comparisons of the performance measures of the three different Bayesian global structure rule learning algorithms and the Bayesian local structure method on the biomedical datasets that is aimed at measuring the statistical significance of the observed differences in the measures.

Table 5-22 reports results from the analysis of accuracy compared in three tests, BGRL_G vs. BLSRL_DG_PG, BGRL_B vs. BLSRL_DG_PG, and BGRL_PG vs. BLSRL_DG_PG. Table 5-23 reports results from the analysis of balanced accuracy comparing the same tests as in Table 5-22. Table 5-24 reports results from the analysis of Relative Classifier Information (RCI) using the same comparisons as in Table 5-22 and Table 5-23.

When comparing these algorithms using EBD as the discretization method, one measure, accuracy, showed no significance between BGRL_B and BLSRL_DG_PG; however, both

BGRL_G and BGRL_PG performed statistically worse than BLSRL_DG_PG. Balanced accuracy and Relative Classifier Information (RCI) shows significance between all of the Bayesian global structure search algorithms and the Bayesian local structure decision graph search method favoring BLSRL_DG_PG.

5.2.2 Discussion

This section shows the comparison of Bayesian global structure search with Bayesian local structure decision graphs. As seen in multiple experiments [67, 68, 124], local structure simplifies the conditional probability table for the constrained BN by allowing for context-specific independencies as seen by the differences in the average number of rules generated (Appendix B.2). We see that indeed, this generalization (reduction in the number of parameters of the CPT) generally leads to greater performance across the multiple measures with balanced accuracy and RCI being statistically significant. This comparison is critical to make since the premise of using Bayesian local structure is the reduction in the number of parameters needs for the model since we are dealing with a sparse matrix (number of variable combinations vs. number of samples). While the benefit of using Bayesian local structure is indeed higher performance on average, fewer rules and simpler models, the build time is significantly different particularly when you compare BGRL_PG to BLSRL_DG_PG. BGRL_PG takes the longest of all the methods with an average of 4 minutes to build, while BLSRL_DG_PG takes an average of 18 minutes, which is more than four times longer. We also noticed that the BGRL algorithms performed better with EBD than MDLPC which again highlights the interaction between the machine learning algorithm and the discretization method (see Appendix B).

Table 5-19. The percent accuracies of the different search strategies for Bayesian global structure and the local structure BLSRL_DG_PG using EBD. Those values in bold is the largest value across the various methods and the largest value of all structures is italicized.

Bayesian Structure	*Global Structure*			*Local Structure*
Search Strategy	Greedy	Beam	Parallel Greedy	DG_PG
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	75.89	**77.68**	**77.68**	76.25
Beer et al.	71.81	70.97	70.69	**72.22**
Bhattacharjee et al.	64.79	72.81	67.26	**78.98**
Bhattacharjee et al.	58.10	62.86	54.52	**64.29**
Golub et al.	66.25	66.43	62.50	**69.46**
Hedenfalk et al.	97.50	97.50	97.50	97.50
Iizuka et al.	***60.00***	56.67	55.00	51.67
Khan et al.	69.39	***73.22***	66.76	64.25
Nutt et al.	***62.00***	50.00	42.00	58.00
Pomeroy et al.	76.67	74.97	78.89	**81.11**
Pomeroy et al.	58.33	56.67	58.33	58.33
Ramaswamy et al.	18.43	46.88	16.07	**52.55**
Rosenwald et al.	55.83	54.58	55.83	**69.17**
Staunton et al.	30.00	35.20	31.67	**45.00**
Shipp et al.	85.00	86.25	**87.32**	80.54
Singh et al.	80.64	82.45	81.91	76.09
Su et al.	38.09	***71.98***	48.29	67.92
Veer et al.	75.58	86.98	66.75	**89.17**
Welsch et al.	87.50	87.50	87.50	87.50
Yeoh et al.	67.53	***72.17***	70.71	70.72
Petricoin et al.	77.97	79.13	75.17	**79.99**
Pusztai et al.	54.07	57.20	55.32	**71.32**
Ranganathan et al.	48.00	57.11	50.36	**65.64**
Average	65.81	69.88	64.92	**71.99**

Table 5-20. The percent balanced accuracies of the different search strategies for Bayesian global structure and the local structure BLSRL_DG_PG using EBD. Those values in bold is the largest value across the various methods and the largest value of all structures is italicized.

Bayesian Structure	*Global Structure*			*Local Structure*
Search Strategy	Greedy	Beam	Parallel Greedy	DG_PG
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	80.57	82.10	87.82	**89.62**
Beer et al.	47.21	46.79	52.14	**53.54**
Bhattacharjee et al.	60.99	62.98	56.30	**65.32**
Bhattacharjee et al.	44.98	44.94	41.73	48.24
Golub et al.	**66.43**	61.79	63.58	65.00
Hedenfalk et al.	97.50	97.50	97.50	97.50
Iizuka et al.	50.83	47.50	47.92	**52.50**
Khan et al.	68.37	**79.99**	68.85	78.29
Nutt et al.	**69.65**	62.67	62.46	69.29
Pomeroy et al.	54.88	55.93	54.24	**59.58**
Pomeroy et al.	49.17	45.83	**50.00**	47.83
Ramaswamy et al.	57.01	58.23	55.17	**64.81**
Rosenwald et al.	57.44	56.19	59.05	**59.92**
Staunton et al.	57.20	59.32	55.61	**59.44**
Shipp et al.	59.83	61.08	**62.62**	57.98
Singh et al.	85.33	85.83	81.75	**88.00**
Su et al.	57.35	65.98	57.44	**73.95**
Veer et al.	81.64	85.98	84.31	**89.32**
Welsch et al.	48.75	48.75	48.75	48.75
Yeoh et al.	47.21	49.13	47.36	**49.42**
Petricoin et al.	53.89	**65.99**	50.53	64.99
Pusztai et al.	68.22	66.12	66.13	**76.50**
Ranganathan et al.	48.42	56.51	50.67	**69.58**
Average	63.04	64.46	62.58	**67.89**

Table 5-21. The Relative Classifier Information (RCI) of the different search strategies for Bayesian global structure and the local structure BLSRL_DG_PG using EBD. Those values in bold is the largest value across the various methods and the largest value of all structures is italicized.

Bayesian Structure	*Global Structure*			*Local Structure*
Search Strategy	Greedy	Beam	Parallel Greedy	DG_PG
Alon et al.	100.00	100.00	100.00	100.00
Armstrong et al.	51.15	46.53	46.46	**61.55**
Beer et al.	0.37	1.97	2.98	**3.12**
Bhattacharjee et al.	32.42	57.68	42.81	**64.99**
Bhattacharjee et al.	1.92	0.26	2.00	**3.70**
Golub et al.	38.25	**39.89**	29.67	39.54
Hedenfalk et al.	72.10	72.10	72.10	72.10
Iizuka et al.	0.79	0.86	0.90	**1.52**
Khan et al.	44.32	**45.95**	35.71	39.00
Nutt et al.	29.89	22.95	26.20	**35.06**
Pomeroy et al.	22.25	25.35	26.59	**35.91**
Pomeroy et al.	1.52	1.29	1.25	1.52
Ramaswamy et al.	27.38	58.13	17.30	**61.99**
Rosenwald et al.	26.11	32.95	29.08	**37.01**
Staunton et al.	31.18	31.87	30.56	**41.58**
Shipp et al.	17.98	19.68	19.83	**27.63**
Singh et al.	28.84	**30.24**	27.60	27.54
Su et al.	66.49	**70.82**	67.67	68.40
Veer et al.	24.67	46.12	35.71	**52.05**
Welsch et al.	23.27	23.27	23.27	23.27
Yeoh et al.	6.98	7.54	7.43	**8.33**
Petricoin et al.	5.66	**13.32**	6.54	9.65
Pusztai et al.	7.44	8.43	8.36	**33.79**
Ranganathan et al.	4.04	5.56	1.63	**7.02**
Average	27.71	31.78	27.57	**35.68**

Table 5-22. The comparison of Accuracies of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
BGRL_G Vs BLSRL_DG_PG	65.72 (19.73) 71.99 (13.78)	-6.26	**0.025** (-2.240)	**.012** (-2.739)
BGRL_B Vs BLSRL_DG_PG	69.88 (16.15) 71.99 (13.78)	-2.10	0.135 (-1.495)	0.107 (-1.678)
BGRL_PG Vs BLSRL_DG_PG	64.92 (19.86) 71.99 (13.78)	-7.07	**0.005** (-2.800)	**0.003** (-3.300)

Table 5-23. The comparison of BACC of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
BGRL_G Vs BLSRL_DG_PG	62.91 (15.80) 67.89 (15.80)	-4.98	**<0.001** (-3.623)	**<0.001** (-4.134)
BGRL_B Vs BLSRL_DG_PG	64.46 (16.07) 67.89 (15.80)	-3.43	**<0.001** (-3.493)	**<0.001** (-4.232)
BGRL_PG Vs BLSRL_DG_PG	62.58 (16.19) 67.89 (15.80)	-5.31	**<0.001** (-3.493)	**<0.001** (-4.441)

Table 5-24. The comparison of RCI of the different search strategies and local structure type using EBD. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
BGRL_G Vs BLSRL_DG_PG	27.39 (25.20) 35.68 (26.41)	-8.29	**0.001** (-3.398)	**0.003** (-3.361)
BGRL_B Vs BLSRL_DG_PG	31.78 (26.48) 35.68 (26.41)	-3.90	**0.010** (-2.589)	**0.012** (-2.721)
BGRL_PG Vs BLSRL_DG_PG	27.57 (24.94) 35.68 (26.41)	-8.11	**<0.001** (-3.980)	**<0.001** (-3.726)

5.3 COMPARISON WITH OTHER RULE LEARNERS

Although comparison among our novel Bayesian rule learning methods allows for the determination of the best rule learning method within the BRGF, comparison with existing state-of-the-art rule generating machine learning algorithms is central to this thesis. For this experiment, we used combinations of discretization method and machine learning algorithm that produced the highest overall accuracy among the classifiers C4.5, RL and BGRL. This resulted in the pairing of MDLPC with C4.5. While MDLPC with C4.5 is not statistically greater than EBD with C4.5 (See Appendix B.1), the MDLPC discretization method provides on average a greater performance across the various measures. With RL, EBD produces higher average performance that is also statistically significant (Lustgarten, et al. [28]). For Bayesian Global Rule Learning (BGRL), we used the beam search method combined with EBD that produced the highest overall performance across multiple methods. Both of these algorithms will be compared with Bayesian Local Structure Rule Learning with Decision Graph and Parallel Greedy Search (BLSRL_DG_PG).

5.3.1 Results

Table 5-25 through Table 5-27 report the means of accuracy, BACC, and RCI respectively for the rule learning algorithms paired with their respective discretization methods: C4.5 paired with MDLPC (C4.5-MDLPC), RL paired with EBD (RL-EBD) and Bayesian Local Structure Rule Learning Decision Graph Parallel Greedy paired with EBD (BLSRL_DG_PG-EBD). In each table, each row is a dataset, and each column is a learning method. The last row in each table represents the overall mean of the specified performance measure across the datasets.

In Table 5-25, which shows the results with respect to *accuracy*, BLSRL_DG_PG-EBD performs better than the C4.5-MDLPC and RL-EBD on 10 datasets. RL-EBD is better than the other compared methods on 8 datasets. C4.5-MDLPC has higher values on 5 datasets. There was 1 dataset which had more than one classifier performing equivalently.

In Table 5-26, which shows the results with respect to *BACC*, BLSRL_DG_PG-EBD performs better than the C4.5-MDLPC and RL-EBD on 14 datasets. RL-EBD is better than the other compared methods on 3 datasets. C4.5-MDLPC has higher values on 5 datasets. There were 2 datasets which had more than one classifier performing equally.

In Table 5-27, which shows the results with respect to RCI, BLSRL_DG_PG-EBD performs better than the C4.5-MDLPC and RL-EBD on 14 datasets. RL-EBD is better than the other compared methods on 2 datasets. C4.5-MDLPC has higher values on 5 datasets. There were 3 datasets which had more than one classifier performing equally.

Tables 5-28 to 5-30 report the results from pair-wise comparisons of the performance measures of the three different rule learning methods, C4.5-MDLPC, RL-EBD, and BLSRL_DG_PG-EBD. They are aimed at measuring the statistical significance of the observed differences in the measures.

Table 5-28 reports results from the analysis of accuracy compared in three tests, RL-EBD vs. C4.5-MDLPC, C4.5-EBD vs. BLSRL_DG_PG, and RL-EBD vs. BLSRL_DG_PG. Table 5-29 reports results from the analysis of BACC comparing the same tests as in Table 5-28. Table 5-30 reports results from the analysis of RCI using the same comparisons as in Table 5-28 and Table 5-29.

When comparing these algorithms using EBD as the discretization method, one measure, accuracy, showed no significant difference between any of the classifiers though both C4.5-

MDLPC and RL-EBD performed on average worse than BLSRL_DG_PG-EBD. BACC and RCI show significant difference between the comparison of C4.5-MDLPC vs. BLSRL_DG_PG-EBD as well as RL-EBD vs. BLSRL_DG_PG-EBD when using the Wilcoxon paired signed rank test. When using the t-test, only BACC showed significant difference between the two previously listed comparisons. None of the measures showed RL and C4.5 being statistically distinguishable, although RL has slightly better BACC (in magnitude) and a larger number of datasets where it had a higher RCI.

5.3.2 Discussion

Using the best pairing of discretization and rule learning algorithms allows different conclusions to be made. First we notice that while on average accuracies are greater for BLSRL_DG_PG when comparing with both C4.5-MDLPC and RL-EBD, they are not statistically significant. This suggests that the different rule learning algorithms are obtaining approximately the same number of samples correct across the 24 datasets. While this is important, the difference between the methods is clearer when we consider the distribution of the class labels among the different datasets.

When the comparison is made using BACC, BLSRL_DG_PG-EBD achieve a higher and statistically significant value than either C4.5-MDLPC or RL-EBD. This implies that while each algorithm may get similar total number of samples correctly predicted, BLSRL_DG_PG-EBD has the ability to accurately predict those classes with fewer representing samples.

RCI has an interesting characteristic. It is evident from the results that with the t-test none of the comparisons achieve significance; however, if the Wilcoxon paired signed rank test is

used, which makes no parametric distributional assumptions, both the comparisons of BLSRL_DG_PG-EBD with C4.5-MDLPC or RL-EBD are shown to be significant.

Comparing the classifiers using Fayyad and Irani MDLPC discretization, we notice another interesting trend. BLSRL_DG_PG paired with MDLPC performs better than C4.5 paired with MDLPC. This suggests a characteristic of the decision graph, the ability to combine paths, as seen in Figure 4-2, overrides the disadvantage of not maximizing the model via entropy. This result, the benefit of decision graphs in combining paths, is similar to the conclusions stated in [16] that an increase in performance over decision tree can be achieved by using an algorithm to ameliorate the small disjunct problem (small number of samples at leaf nodes). Qualitatively, BLSRL_DG_PG-EBD on average takes longer to run and produces more rules since its search space is significantly larger than either C4.5 or RL; however, it utilizes fewer variables than either C4.5-MDLPC or RL-EBD (see Appendix B). Another interesting result, when comparing the methodologies of C4.5 to BLSRL_DG, is that BLSRL does not have a trimming of cost-complexity pruning option currently implemented. This means that the BN algorithm assists in reigning in over-specialization without an explicit heuristic such as those used in the C4.5 algorithm.

In summary, I have shown that BLSRL_DG_PG paired with EBD generates on average greater performance that is statistically significant when compared to either C4.5 with MDLPC or RL with EBD. It generates greater number of rules on average, but with fewer variables. Although this comes at a cost in terms of running times, improvements in efficiency and optimization of the BLSRL_DG_PG can help alleviate the problem to some degree, as is seen by the running times of BLSRL_DG_G when compared to the other algorithms.

Table 5-25. The percent accuracies of the different rule learning methods paired with the discretization method that gives it the highest overall accuracy. Those values in bold are the largest values across the various methods.

Performance Measure	*Accuracy*		
Classifier Method	C4.5-MDLPC	RL-EBD	BLSRL_DG_PG-EBD
Alon et al.	100.00	100.00	100.00
Armstrong et al.	82.14	**82.86**	76.25
Beer et al.	66.39	**75.56**	72.22
Bhattacharjee et al.	**86.17**	77.13	78.98
Bhattacharjee et al.	62.62	60.71	**64.29**
Golub et al.	**77.50**	71.96	69.46
Hedenfalk et al.	95.00	94.17	**97.50**
Iizuka et al.	48.33	41.67	**51.67**
Khan et al.	84.44	**88.33**	64.25
Nutt et al.	52.00	**62.00**	58.00
Pomeroy et al.	71.11	**83.00**	81.11
Pomeroy et al.	**61.67**	53.33	58.33
Ramaswamy et al.	52.50	42.14	**52.55**
Rosenwald et al.	58.33	53.75	**69.17**
Staunton et al.	28.33	33.33	**45.00**
Shipp et al.	**83.39**	69.82	80.54
Singh et al.	79.45	**89.18**	76.09
Su et al.	65.46	**70.72**	67.92
Veer et al.	69.29	63.00	**89.17**
Welsch et al.	**95.00**	87.50	87.50
Yeoh et al.	71.08	**71.16**	70.72
Petricoin et al.	73.26	78.54	**79.99**
Pusztai et al.	52.79	40.40	**71.32**
Ranganathan et al.	55.67	51.82	**65.64**
Average	69.66	68.42	**71.99**

Table 5-26. The percent BACC of the different rule learning methods paired with the discretization method that gives it the highest overall accuracy. The values in bold are the largest values across the various methods.

Performance Measure	*Balanced Accuracy (BACC)*		
Classifier Method	C4.5-MDLPC	RL-EBD	BLSRL_DG_PG-EBD
Alon et al.	100.00	100.00	100.00
Armstrong et al.	85.42	80.88	**89.62**
Beer et al.	45.00	47.32	**53.54**
Bhattacharjee et al.	64.02	65.11	**65.32**
Bhattacharjee et al.	43.21	43.45	**48.24**
Golub et al.	**67.43**	59.79	65.00
Hedenfalk et al.	95.00	97.50	97.50
Iizuka et al.	43.33	35.83	**52.50**
Khan et al.	83.18	**85.80**	78.29
Nutt et al.	65.85	**71.67**	69.29
Pomeroy et al.	58.71	58.21	**59.58**
Pomeroy et al.	**54.17**	50.33	47.83
Ramaswamy et al.	**66.41**	59.47	64.81
Rosenwald et al.	53.82	51.10	**59.92**
Staunton et al.	54.25	56.55	**59.44**
Shipp et al.	**60.65**	48.75	57.98
Singh et al.	79.83	**90.33**	88.00
Su et al.	69.26	73.82	**73.95**
Veer et al.	66.32	64.40	**89.32**
Welsch et al.	**56.25**	52.50	48.75
Yeoh et al.	46.83	48.35	**49.42**
Petricoin et al.	51.98	61.07	**64.99**
Pusztai et al.	63.87	73.12	**76.50**
Ranganathan et al.	50.58	51.83	**69.58**
Average	63.56	63.63	**67.89**

Table 5-27. The percent RCI of the different rule learning methods paired with the discretization method that gives it the highest overall accuracy. Those values in bold are the largest values across the various methods.

Performance Measure	*Relative Classifier Information (RCI)*		
Classifier Method	C4.5-MDLPC	RL-EBD	BLSRL_DG_PG-EBD
Alon et al.	100.00	100.00	100.00
Armstrong et al.	**68.50**	54.30	61.55
Beer et al.	**4.04**	2.93	3.12
Bhattacharjee et al.	40.91	62.38	**64.99**
Bhattacharjee et al.	0.22	1.96	**3.70**
Golub et al.	**53.84**	49.56	39.54
Hedenfalk et al.	69.05	72.10	72.10
Iizuka et al.	0.50	0.80	**1.52**
Khan et al.	**61.57**	34.75	39.00
Nutt et al.	31.06	33.95	**35.06**
Pomeroy et al.	33.90	34.29	**35.91**
Pomeroy et al.	**3.60**	1.08	1.52
Ramaswamy et al.	61.44	**72.51**	61.99
Rosenwald et al.	1.22	2.61	**37.01**
Staunton et al.	32.15	39.61	**41.58**
Shipp et al.	14.82	3.33	**27.63**
Singh et al.	27.76	**49.62**	27.54
Su et al.	60.96	**70.17**	68.40
Veer et al.	23.05	3.92	**52.05**
Welsch et al.	23.27	23.27	23.27
Yeoh et al.	0.17	6.83	**8.33**
Petricoin et al.	3.32	9.28	**9.65**
Pusztai et al.	11.05	24.21	**33.79**
Ranganathan et al.	5.88	5.84	**7.02**
Average	30.51	31.64	**35.68**

Table 5-28. The comparison of accuracies of the different rule learning methods using the discretization method which gives them the greatest accuracy. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
RL-EBD Vs C4.5-MDLPC	68.42 (18.42) 69.66 (17.11)	-1.24	0.412 (-0.821)	0.424 (-0.814)
C4.5-MDLPC Vs BLSRL_DG_PG-EBD	69.66 (17.11) 71.99 (13.78)	-2.32	0.294 (-1.049)	0.234 (-1.223)
RL-EBD Vs BLSRL_DG_PG-EBD	68.42 (18.42) 71.99 (13.78)	-3.57	0.158 (-1.412)	0.147 (-1.500)

Table 5-29. The comparison of BACC of the different rule learning methods using the discretization method which gives them the greatest accuracy. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
RL-EBD Vs C4.5-MDLPC	63.59 (17.25) 63.56 (15.62)	0.03	0.903 (0.122)	0.979 (0.026)
C4.5-MDLPC Vs BLSRL_DG_PG-EBD	63.56 (15.62) 67.89 (15.80)	-4.33	**0.013** (-2.494)	**0.009** (-2.841)
RL-EBD Vs BLSRL_DG_PG-EBD	63.59 (17.25) 67.89 (15.80)	-4.60	**0.007** (-2.678)	**0.009** (-2.857)

Table 5-30. The comparison of RCI of the different rule learning methods using the discretization method which gives them the greatest accuracy. A positive score represents results that favor the first item in the comparison.

Classifier	Value (Std Dev)	Diff	Wilcoxon (Z-score)	t-test (t-score)
RL-EBD Vs C4.5-MDLPC	30.05 (28.60) 30.51 (28.27)	-0.46	0.758 (0.308)	0.838 (-0.207)
C4.5-MDLPC Vs BLSRL_DG_PG-EBD	30.51 (28.27) 35.68 (26.41)	-5.16	**0.026** (-2.224)	0.061 (-1.968)
RL-EBD Vs BLSRL_DG_PG-EBD	30.05 (28.60) 35.68 (26.41)	-5.63	**0.046** (-1.999)	0.098 (-1.726)

5.4 RESULTS ON UNPUBLISHED PROTEOMIC SETS

The previous experiments, described in Sections 5.1 - 5.3, show the differences in performance of C4.5, RL, Bayesian global rule learning with beam search, and Bayesian Local Structure Rule Learning with Decision Graph structure and Parallel Greedy search strategy. The comparisons in Section 5.2 and 5.3 show the differences between these algorithms when choosing the discretization method that give each the greatest overall performance. To make sure that the results are reflective of a general trend, BLSRL_DG_PG paired with EBD produces better performance on average, four proteomic datasets that were not previously analyzed will be used as test data.

The four proteomic datasets that are described in Table 3-7 will be used. The first dataset was produced by the Bowser lab on the Surface Enhanced Laser/Desorption Ionization Time of Flight (SELDI-TOF) platform. It contains 67 patient's blood plasma, of which approximately

one half have Amyotrophic Lateral Sclerosis (ALS). The second dataset was produced by me on the Matrix Assisted Laser/Desorption Ionization Time of Flight platform. It contains 22 patients' cerebrospinal fluid (CSF) of which a little less than half have ALS. The third dataset contains 239 patients' plasma of which a little less than half have a form of lung cancer. This set was derived from the Lung Spore project. The fourth and final dataset for comparison is also an ALS dataset created by the Bowser lab. It is a SELDI-TOF dataset containing 168 patients of which one third are diagnosed with ALS. All datasets are high dimensional, having greater than 17,900 observed variables.

5.4.1 Results

Table 5-32 - Table 5-34 report the means of accuracy, BACC and RCI, respectively, for the rule learning algorithms paired with their respective discretization method as described in Section 5.3. In each table, each row is a dataset, and each column is a learning method. The last row in each table represents the overall mean of the specified performance measure across the datasets.

In Table 5-32, which shows the results with respect to accuracy, BLSRL_DG_PG-EBD performs better than C4.5-MDLPC, RL-EBD, and BGRL_B-EBD on 2 datasets. C4.5-MDLPC achieves a higher value on 1 dataset. There was 1 dataset which had more than one classifier performing equivalently.

In Table 5-33, which shows the results with respect to BACC, BLSRL_DG_PG-EBD performs better than the C4.5-MDLPC, RL-EBD, and BGRL_B-EBD on 2 datasets. RL-EBD achieves a higher value on 1 dataset. There was 1 dataset which had more than one classifier performing equivalently.

Table 5-34, which shows the results with respect to RCI, BLSRL_DG_PG-EBD performs better than the C4.5-MDLPC, RL-EBD, and BGRL_B-EBD on 2 datasets. RL-EBD achieves a higher value on 1 dataset.

5.4.2 Discussion

As we see, the pattern exhibited in the Sections 5.1 - 5.3 continues. We see that overall, the accuracy on average is similar among the different classifiers, though BLSRL_DG_PG with EBD achieves higher accuracy on greater number of datasets while C4.5 with MDLPC achieves the highest average. When looking at balanced accuracy and RCI, on average, Bayesian local structure using a decision graph with parallel beam search combined with EBD out-performs the other compared classifiers, both in the number of datasets achieving highest accuracy as well as the overall average.

These results, combined with those in the previous sections, support the idea that using the BRGF allows you to increase performance on multiple measures when analyzing biomedical genomic and proteomic datasets. We saw a trend where choosing EBD allowed for greater overall performance in the Bayesian methods, and MDLPC allowed for greater performance when combined with C4.5.

Table 5-31, we summarize some of the important strengths and weaknesses of the different algorithms within the BRGF. This table summarizes the results of Chapter 5 highlighting in particular the conclusions from the overall classification performance, the speed, and the complexity of the model (number of rules in the model).

Table 5-31. A summary of the different rule learning algorithms in the BRGF strengths and weaknesses as seen from the results in Chapters 5.0 .

Learning Method	*Search Strategy*	*Summary*
RL	Beam	**Strength**: Can handle small sample sizes, sampling with replacement **Weakness**: Prone to over fitting, certainty factors derived from statistical measures such as Signal-To-Noise, Positive Predictive Value, requires bias space search to find "optimal" set of parameters
Bayesian Global Structure (BGS)	Greedy	**Strength**: Fast, Probabilistic **Weakness**: Prone to local maxima, combinatorial explosion of rules
	Beam	**Strength**: Faster than Parallel Greedy, Less prone to local maxima, Performs best of all Bayesian global methods tested, Probabilistic **Weakness**: combinatorial explosion of rules, slower than greedy
	Parallel Greedy	**Strength**: Better than greedy search, allows multiple start possibilities, Probabilistic **Weakness**: combinatorial explosion of rules, slowest of all the Bayesian global methods, performs worse than Beam search
Bayesian Local Structure Decision Tree (BLSDT)	Greedy	**Strength**: Fastest of all local search methods, allows for reduction of parameters in CPT, Probabilistic, simpler model than BGS **Weakness**: prone to local maxima, performs worse than any of the decision graph methods
	Parallel Greedy	**Strength**: reduction of parameters in CPT, Probabilistic, simpler model than BGS **Weakness**: Slower than greedy search, performs on average a little worse than greedy
Bayesian Local Structure Decision Graph (BLSDG)	Greedy	**Strength**: Faster than PG, better than BLSDT and BGS except Beam, combination of paths, allows for reduction of parameters in CPT, Probabilistic, simpler model than BGS **Weakness**: Prone to local maxima
	Parallel Greedy	**Strength**: Best method, combination of paths, allows for reduction of parameters in CPT, Probabilistic, fewer rules than BGS **Weakness**: Slower than greedy DG and many of the other methods.

Table 5-32. The accuracy of four different rule learning algorithms on four unpublished proteomic datasets. Those accuracies that are the highest among all classifiers on the dataset are in bold font.

Datasets	*C4.5-MDLPC*	*RL-EBD*	*BGRL_B-EBD*	*BLSRL_DG_PG-EBD*
SELDI-TOF ALS Plasma	48.30%	53.17%	39.53%	**55.63%**
MALDI-TOF ALS CSF	**69.00%**	60.00%	64.00%	58.00%
SELDI-TOF Lung Cancer	**72.27%**	55.53%	60.39%	68.41%
SELDI-TOF ALS CSF	61.44%	36.33%	62.05%	**62.62%**
Average	**62.75%**	51.26%	56.49%	61.17%

Table 5-33. The BACC of four different rule learning algorithms on four unpublished proteomic datasets. Those accuracies that are the highest among all classifiers on the dataset are in bold font.

Datasets	*C4.5-MDLPC*	*RL-EBD*	*BGRL_B-EBD*	*BLSRL_DG_PG-EBD*
SELDI-TOF ALS Plasma	48.14%	**53.21%**	39.29%	49.55%
MALDI-TOF ALS CSF	58.33%	54.17%	53.33%	58.33%
SELDI-TOF Lung Cancer	69.52%	67.72%	63.44%	**72.92%**
SELDI-TOF ALS CSF	49.16%	45.08%	51.37%	**53.77%**
Average	56.29%	55.04%	51.86%	**58.64%**

Table 5-34. The RCI of four different rule learning algorithms on four unpublished proteomic datasets. Those accuracies that are the highest among all classifiers on the dataset are in bold font.

Datasets	*C4.5-MDLPC*	*RL-EBD*	*BGRL_B-EBD*	*BLSRL_DG_PG-EBD*
SELDI-TOF ALS Plasma	0.05%	1.01%	1.32%	**1.65%**
MALDI-TOF ALS CSF	3.21%	**6.37%**	1.79%	5.70%
SELDI-TOF Lung Cancer	12.43%	9.27%	6.06%	**18.05%**
SELDI-TOF ALS CSF	0.08%	0.76%	0.23%	**1.33%**
Average	3.94%	4.35%	2.35**%**	**6.68%**

6.0 CONCLUSIONS AND FUTURE WORK

Using a Bayesian approach to generating rules is central to this body of work. We showed increased performance and decreased number of variables selected, which allows for a more parsimonious model while increasing performance as compared to the state-of-the-art rule generation techniques. We also established a framework for integrating different probabilistic measures allowing for easy combination of rule learning with probabilistic network learning techniques. The BRGF allows for future usage of decision theoretic models with rules due to the generation of probabilistic rule models, which was unavailable before from rule generating algorithms. While it might have been possible to assign a probability to a rule, the BRGF allows the rule model (all of the rules) to be optimized to be the most probable.

6.1 SPECIFIC FINDINGS

The first hypothesis, using the BRGF is sufficient to generate probabilistic rules, has been tested and proven correct through empirical determination in Chapter 4.0 . We were able to show that using Bayesian discretization combined with Bayesian rule learning, that is both components of the BRGF, can generate probabilistic rules that reflect the BN structure while encapsulating the uncertainty derived from the CPT within the certainty factor of the rule. This therefore also strongly supports claim 1.

The second hypothesis that using the BRGF leads to an increased classification performance on average over standard rule generation techniques, was shown to be true (as measured by BACC and RCI) in Sections 5.3 and 5.4. We not only tested for statistical significance (whether or not the BRGF had a greater performance on multiple measures) and showed that it did, but also showed that this pattern continued when BRGF methods were applied to 4 unpublished proteomic datasets.

Using the publicly available biomedical data, we were able to show strong support for claim 2 that under controlled conditions (constrained BNs), Bayesian local structure increases classification performance on average over Bayesian global structure for rule generation. Specifically, we showed that using Bayesian local structure decision graph combined with a parallel greedy search paradigm was statistically significant and greater on average than any of the Bayesian global structure methods, which can be considered a specific instance of Bayesian local structure decision tree (a complete decision tree).

Claim 3, under controlled conditions (constrained BNs), Bayesian discretization produces equivalent or greater performance on average than Fayyad and Irani's MDLPC [49] discretization when combined with a Bayesian rule generation methods, can be supported, but only in specific combinations. For the Bayesian global structure and the Bayesian decision graphs, claim 3 has strong support shown by the statistical significance and greater performance when using a Bayesian discretization method. The combination of a Bayesian discretization method and the decision tree algorithm results in poorer performance, albeit not statistically significant, when compared with the decision tree algorithm and an entropy based discretization (MDLPC). This could be due to multiple reasons including the sensitivity of decision trees to the number of intervals, EBD produces more than MDLPC, and its correlation to the sparseness of

the samples as expressed by Yang, et al. [157] and Carvalho, et al.[16]. The overwhelming performance of decision graphs with a Bayesian discretization method and the equivalent performance of decision trees with entropy-based discretization or Bayesian discretization provide enough support for the correctness of this claim.

Claim 4, under controlled conditions (constrained BNs), the BRGF leads to increased classifier performance on average over state-of-the-art traditional rule learning algorithms, is strongly supported by the statistical significance and greater performance on average of the BRGF. This was validated using a large collection of biomedical datasets (Chapter 5). However, this performance is not universal across all datasets tested as shown by the increased performance of RL on few datasets. The creation of probabilistic rules, the ability to do trivial inference, and the flexibility of being using a probabilistic model should overcome the difference in performance that might occur on some datasets.

During the evaluation and comparison of the BRGF with traditional rule learners, the BRGF showed that it balances the number of rules generated with the number of variables used while achieving (on average) greater performance. It was also argued that using purely the number of correct predictions is not sufficient enough as a performance measure. The BRGF trades predictions of the majority class for those samples belonging to the minority classes as shown by the increased RCI score.

The BRGF has impacts on both the experimental and clinical domains. For the experimental, the BRGF suggests fewer possible biomarkers (observed variables) creating a more parsimonious model than traditional rule generation techniques (see Appendix B) while increasing performance of the model on average. This assists in decreasing the number of possible targets a biologist has to explore using more time consuming techniques such as knock-

out studies for genomics and MS/MS and identification for proteomics [1, 4, 78, 96, 101, 158-161].

In the clinical domain, understandable models assist the clinician in understanding how the model arrives at the diagnosis [2, 14, 18, 25, 26, 115, 162], and the probabilistic nature of the rules derived from the BRGF can help a clinician understand the probability of the condition being predicted explicitly. This can help especially when integrating disparate data sources that each predict with a probability of a specific clinical finding [124, 163, 164]. It also allows for the possibility of integrating the BRGF into a decision theoretic workflow, such as decision support systems that can be utilized for clinical care, for the previously mentioned reasons of production of probabilistic rules. The idea of heterogeneous data sources as well as additional future expansions of the BRGF will be discussed in the next section below.

6.2 FUTURE WORK

While the potential benefits of these methods have been shown in this work, the method space explored is a small percentage of the possibilities. Certain heuristics have been used including limiting the number of parents (or number of variables per a path) and the choice of different search strategies. Future exploration of techniques that have been applied to BNs can also now be applied to the rule learning space.

6.2.1 Bayesian discretization methods

As explained in chapter three, there is an additional discretization method that is within the BRGF, but for focus on the framework, was not included within the analysis. Future experiments might show that changing the penalty (EBD utilizes a combinatorial penalty while EBD-D uses a distance penalty) when scoring a particular policy might increase the performance of the BRGF. We also, using the Bayesian framework, have the ability to manipulate the Dirichlet hyperparameters. By changing these priors, we would be injecting information into the discretization process. These possibilities, along with a further investigation into the interaction of the Bayesian discretization methods and a Bayesian rule generation method signifies that there are large areas within this small component that can be explored as future work.

6.2.2 Beam Search for Local Structure

As shown in Chapter 2 and 3, the overall space of BNs for both global and local structures is large. Usage of beam search heuristic therefore requires curtailing the possible combinations in some fashion. For Bayesian global structure search, the usage of the heuristic to eliminate repetitive model consideration helps reduce the model space that is considered; however the same heuristic cannot be applied due to the variable nature of the splitting that occurs with local structure. It might be possible to develop a heuristic to eliminate many different models from consideration by taking advantage of beam search that was shown to improve performance over the greedy or parallel greedy within the Bayesian global structure (Chapter 5.2).

6.2.3 Heuristics for stopping specialization for Bayesian Decision Trees and Graphs

The results of chapter 5 are particularly significant especially when considering that the specialization of both the Bayesian decision trees and decision graphs only stops specialization when the leaf node in the tree has a pure distribution (only one class of the target is represented). Algorithms such as C4.5 have pruning capabilities to eliminate some of the over-specialization to which decision trees are prone. As future work, we could experiment with different stopping heuristics such as purity of the distribution within the leaf node being less than 100% (more than one class represented).

6.2.4 Bayesian Model Averaging

An additional approach used by Visweswaran, which utilized Bayesian local structure, also utilized a technique called Bayesian model averaging (BMA) [124]. This technique utilizes multiple BNs to produce a weighted prediction, weighting each network by the probability of the data given the model. Since the parallel greedy method as explained in Chapter 3.1.4.2 generates 1000 models, it is possible to do a BMA without any extra overhead of additional generation. This requires extensive testing since it is not guaranteed that these additional models provide increased performance. There is a possibility that the model chosen is really the most probable of all the 1000 models generated, and so when weighting the predictions, the additional models will contribute little to the prediction.

6.2.5 Heterogeneous Data Sources for analysis

Given that the rule models are now probabilistic, the integration of heterogeneous data sources, e.g., genomic and proteomic data, can be done using the rigorous mathematics of probability incorporating the biologists beliefs. The idea is that building models from different data sources and then combining them for inference will increase the overall performance of the classifier. This idea has been explored previously and it was shown that analyzing the datasets separately achieved better performance [164].

6.2.6 Incorporation of Prior Knowledge

The usage of a BN for rule generation allows for incorporation of prior information through the weighting of interaction of variables and the target. Currently the BRGF weights each variable uniformly in its strength of interaction with the target, thus treating each interaction as equally likely. This is not necessarily the case especially in the analysis genomic and proteomic data. It is known that there are some interactions between genes and proteins as detailed by resources such as KEGG [165] and EPO-KB [111, 166]. Though the approach of integrating prior information into machine learning is often in the form of clustering [167-169], no experiment has recently been done using probabilistic weighting.

6.2.6.1 Prior Biological Knowledge derived from Literature

It is possible to mine relationships from literature that indicate gene-disease or protein-disease relationships. It might be possible to codify these relationships as probabilities and use them as weights for the interactions between the variable and target. This involves many difficulties

though, such as how exactly to codify the probabilities or what prior weights to assign to variables that are not given a prior probability from the literature. However prior work in the field shows promise for applicability of this method as an effective way to retrieve probabilities of interaction and their subsequent utilization in construction of BNs [119, 123] .

APPENDIX A

RULE MODELS PRODUCED BY RULE LEARNING ALGORITHMS

Bhattacharjee et al Prognostic Genomic Dataset:

C4.5 with MDLPC

```
A2090 = NegInfinity..317.25
|  A459 = NegInfinity..-40: 1 (2.0)
|  A459 = -40..Infinity
|  |  A1627 = NegInfinity..717
|  |  |  A2142 = NegInfinity..404.75
|  |  |  |  A5250 = NegInfinity..1657.375
|  |  |  |  |  A270 = NegInfinity..133: 1 (4.0)
|  |  |  |  |  A270 = 133..Infinity: 2 (1.0)
|  |  |  |  A5250 = 1657.375..Infinity: 2 (51.0)
|  |  |  A2142 = 404.75..Infinity: 1 (3.0)
|  |  A1627 = 717..Infinity: 1 (2.0)
A2090 = 317.25..Infinity: 1 (6.0)
```

RL with EBD

1. ((A522 = Negative Infinity..1,394.250)) ==> (@Class = 1)
   CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

2. ((A5124 = Negative Infinity..329.000)) ==> (@Class = 1)
   CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

3. ((A1073 = Negative Infinity..259.917)) ==> (@Class = 1)
   CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

4. ((A1697 = Negative Infinity..1,519.750)) ==> (@Class = 1)

CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

5. ((A4551 = Negative Infinity..-68.500)) ==> (@Class = 1)
CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

6. ((A501 = 195.000..Infinity)) ==> (@Class = 1)
CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

7. ((A201 = 466.000..Infinity)) ==> (@Class = 1)
CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

8. ((A4449 = 184.000..Infinity)) ==> (@Class = 1)
CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

9. ((A1627 = 717.000..Infinity)) ==> (@Class = 1)
CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

10. ((A1286 = 306.500..Infinity)) ==> (@Class = 1)
CF=1.012, Av.Cost=1.0, CF/Cost=1.012, P=0.003, TP=4, FP=0, Pos=17, Neg=52

11. ((A308 = 344.750..380.750)) ==> (@Class = 1)
CF=1.01, Av.Cost=1.0, CF/Cost=1.01, P=0.001, TP=5, FP=0, Pos=17, Neg=52

12. ((A3481 = 2,391.000..Infinity)) ==> (@Class = 2)
CF=1.008, Av.Cost=1.0, CF/Cost=1.008, P=0.17, TP=6, FP=0, Pos=52, Neg=17

13. ((A892 = Negative Infinity..-74.000)) ==> (@Class = 2)
CF=1.005, Av.Cost=1.0, CF/Cost=1.005, P=0.047, TP=10, FP=0, Pos=52, Neg=17

14. ((A308 = Negative Infinity..344.750)) ==> (@Class = 2)
CF=1.003, Av.Cost=1.0, CF/Cost=1.003, P=0.002, TP=19, FP=0, Pos=52, Neg=17

15. ((A2537 = Negative Infinity..-40.500)) ==> (@Class = 2)
CF=1.002, Av.Cost=1.0, CF/Cost=1.002, P=0.001, TP=21, FP=0, Pos=52, Neg=17

16. ((A2561 = Negative Infinity..1,032.250)) ==> (@Class = 2)
CF=1.002, Av.Cost=1.0, CF/Cost=1.002, P=0.0, TP=22, FP=0, Pos=52, Neg=17

17. ((A3711 = 2,194.500..Infinity)) ==> (@Class = 2)
CF=1.002, Av.Cost=1.0, CF/Cost=1.002, P=0.0, TP=22, FP=0, Pos=52, Neg=17

18. ((A4474 = 695.750..Infinity)) ==> (@Class = 2)
CF=0.955, Av.Cost=1.0, CF/Cost=0.955, P=0.009, TP=20, FP=1, Pos=52, Neg=17

0. ((A3405 = Negative Infinity..-62.500)) ==> (@Class = 1)
CF=0.783, Av.Cost=1.0, CF/Cost=0.783, P=0.0, TP=7, FP=2, Pos=17, Neg=52

Variables Used:
A522,A1073,A1627,A501,A892,A308,A3405,A3711,A4551,A4474,A4449,A201,A2561,A3481,A1697,A5124,A1286,A2537

Bayesian Global Rule Learning with Beam Search

1. ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.667, Av.Cost=1.0, CF/Cost=0.667, P=0.246, TP=1, FP=0, Pos=17, Neg=52

2. ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

3. ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

4. ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

5. ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 2)
CF=0.952, Av.Cost=1.0, CF/Cost=0.952, P=0.002, TP=19, FP=0, Pos=52, Neg=17

6. ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 2)
CF=0.667, Av.Cost=1.0, CF/Cost=0.667, P=0.754, TP=1, FP=0, Pos=52, Neg=17

7. ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

8. ((A4474 = 695.750..Infinity) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

9. ((A4474 = 695.750..Infinity) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

10. ((A4474 = 695.750..Infinity) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)

CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

11. ((A4474 = 695.750..Infinity) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

12. ((A4474 = 695.750..Infinity) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

13. ((A4474 = 695.750..Infinity) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

14. ((A4474 = 695.750..Infinity) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

15. ((A4474 = 695.750..Infinity) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

16. ((A4474 = 695.750..Infinity) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

17. ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.75, Av.Cost=1.0, CF/Cost=0.75, P=0.058, TP=2, FP=0, Pos=17, Neg=52

18. ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.75, Av.Cost=1.0, CF/Cost=0.75, P=0.058, TP=2, FP=0, Pos=17, Neg=52

19. ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

20. ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

21. ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 2)
CF=0.857, Av.Cost=1.0, CF/Cost=0.857, P=0.231, TP=5, FP=0, Pos=52, Neg=17

22. ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.857, Av.Cost=1.0, CF/Cost=0.857, P=0.001, TP=5, FP=0, Pos=17, Neg=52

23. ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.8, Av.Cost=1.0, CF/Cost=0.8, P=0.013, TP=3, FP=0, Pos=17, Neg=52

24. ((A4474 = 549.500..695.750) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

25. ((A4474 = 549.500..695.750) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

26. ((A4474 = 549.500..695.750) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

27. ((A4474 = 549.500..695.750) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

28. ((A4474 = 549.500..695.750) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

29. ((A4474 = 549.500..695.750) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

30. ((A4474 = 549.500..695.750) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

31. ((A4474 = 549.500..695.750) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.667, Av.Cost=1.0, CF/Cost=0.667, P=0.246, TP=1, FP=0, Pos=17, Neg=52

32. ((A4474 = 549.500..695.750) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

33. ((A4474 = Negative Infinity..549.500) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

34. ((A4474 = Negative Infinity..549.500) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

35. ((A4474 = Negative Infinity..549.500) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

36. ((A4474 = Negative Infinity..549.500) (A2987 = -31.750..Infinity) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

37. ((A4474 = Negative Infinity..549.500) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 2)
CF=0.963, Av.Cost=1.0, CF/Cost=0.963, P=0.0, TP=25, FP=0, Pos=52, Neg=17

38. ((A4474 = Negative Infinity..549.500) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 2)
CF=0.75, Av.Cost=1.0, CF/Cost=0.75, P=0.565, TP=2, FP=0, Pos=52, Neg=17

39. ((A4474 = Negative Infinity..549.500) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

40. ((A4474 = Negative Infinity..549.500) (A2987 = -31.750..Infinity) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

41. ((A4474 = Negative Infinity..549.500) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

42. ((A4474 = Negative Infinity..549.500) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

43. ((A4474 = Negative Infinity..549.500) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

44. ((A4474 = Negative Infinity..549.500) (A2987 = Negative Infinity..-31.750) (A671 = 51.500..Infinity) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

45. ((A4474 = Negative Infinity..549.500) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.75, Av.Cost=1.0, CF/Cost=0.75, P=0.058, TP=2, FP=0, Pos=17, Neg=52

46. ((A4474 = Negative Infinity..549.500) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = -40.000..Infinity) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

47. ((A4474 = Negative Infinity..549.500) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = 124.750..Infinity)) ==> (@Class = 1)
CF=0.667, Av.Cost=1.0, CF/Cost=0.667, P=0.246, TP=1, FP=0, Pos=17, Neg=52

48. ((A4474 = Negative Infinity..549.500) (A2987 = Negative Infinity..-31.750) (A671 = Negative Infinity..51.500) (A459 = Negative Infinity..-40.000) (A1754 = Negative Infinity..124.750)) ==> (@Class = 1)
CF=0.5, Av.Cost=1.0, CF/Cost=0.5, P=1.0, TP=0, FP=0, Pos=17, Neg=52

Variables Used: A671,A1754,A4474,A2987,A459

Bayesian Local Structure Rule Learning with Decision Graphs and Parallel Greedy Search with EBD

1. ((A2900 = Negative Infinity..1,808.750) (A2002 = Negative Infinity..2.750)) ==> (@Class = 2)
CF=0.981, Av.Cost=1.0, CF/Cost=0.981, P=0.0, TP=52, FP=0, Pos=52, Neg=17

2. ((A2900 = 1,808.750..Infinity) (A4474 = Negative Infinity..549.500) (A1627 = Negative Infinity..717.000)) ==> (@Class = 2)
CF=0.981, Av.Cost=1.0, CF/Cost=0.981, P=0.0, TP=52, FP=0, Pos=52, Neg=17

3. ((A2900 = 1,808.750..Infinity) (A4474 = 695.750..Infinity) (A1627 = Negative Infinity..717.000)) ==> (@Class = 2)
CF=0.981, Av.Cost=1.0, CF/Cost=0.981, P=0.0, TP=52, FP=0, Pos=52, Neg=17

4. ((A2900 = 1,808.750..Infinity) (A4474 = 549.500..695.750) (A4882 = Negative Infinity..372.333) (A45 = 399.500..Infinity)) ==> (@Class = 2)
CF=0.981, Av.Cost=1.0, CF/Cost=0.981, P=0.0, TP=52, FP=0, Pos=52, Neg=17

5. ((A2900 = Negative Infinity..1,808.750) (A2002 = 2.750..Infinity)) ==> (@Class = 1)
CF=0.947, Av.Cost=1.0, CF/Cost=0.947, P=0.0, TP=17, FP=0, Pos=17, Neg=52

6. ((A2900 = 1,808.750..Infinity) (A4474 = 549.500..695.750) (A4882 = 372.333..Infinity)) ==> (@Class = 1)
CF=0.947, Av.Cost=1.0, CF/Cost=0.947, P=0.0, TP=17, FP=0, Pos=17, Neg=52

7. ((A2900 = 1,808.750..Infinity) (A4474 = 549.500..695.750) (A4882 = Negative Infinity..372.333) (A45 = Negative Infinity..399.500)) ==> (@Class = 1)
CF=0.947, Av.Cost=1.0, CF/Cost=0.947, P=0.0, TP=17, FP=0, Pos=17, Neg=52

8. ((A2900 = 1,808.750..Infinity) (A4474 = Negative Infinity..549.500) (A1627 = 717.000..Infinity)) ==> (@Class = 1)
CF=0.947, Av.Cost=1.0, CF/Cost=0.947, P=0.0, TP=17, FP=0, Pos=17, Neg=52

9. ((A2900 = 1,808.750..Infinity) (A4474 = 695.750..Infinity) (A1627 = 717.000..Infinity)) ==> (@Class = 1)
CF=0.947, Av.Cost=1.0, CF/Cost=0.947, P=0.0, TP=17, FP=0, Pos=17, Neg=52

Variables Used: A2002,A4474,A1627,A2900,A45,A4882

APPENDIX B

B.1 THE RESULTS OF MDLPC VS. EBD ON BAYESIAN GLOBAL STRUCTURE RULE LEARNING AND C4.5

Table 6-1. The percent accuracies of various rule learning methods using MDLPC. BGRL_G is the Bayesian Global Rule Learning with Greedy Search.

Datasets	***BGRL_G***	***BGRL_B***	***BGRL_PG***	***C4.5***	***RL***
Alon et al.	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	68.75	70.18	84.46	82.14	84.29
Beer et al.	76.39	73.19	74.58	66.39	75.42
Bhattacharjee et al.	78.50	81.68	82.39	86.17	74.11
Bhattacharjee et al.	62.62	65.24	63.81	62.62	67.86
Golub et al.	71.96	67.50	67.50	77.50	69.11
Hedenfalk et al.	97.50	97.50	97.50	95.00	89.17
Iizuka et al.	51.67	56.67	48.33	48.33	53.33
Khan et al.	72.21	76.55	66.58	84.44	84.63
Nutt et al.	52.00	36.00	42.00	52.00	54.00
Pomeroy et al.	65.56	69.17	67.78	71.11	82.22
Pomeroy et al.	55.00	56.67	53.33	61.67	55.00
Ramaswamy et al.	19.29	39.30	25.00	52.50	36.43
Rosenwald et al.	57.92	57.08	55.83	58.33	57.50
Staunton et al.	35.00	32.67	36.67	28.33	25.00
Shipp et al.	83.75	83.75	91.25	83.39	68.04
Singh et al.	87.27	84.45	82.55	79.45	89.00
Su et al.	40.91	61.98	57.03	65.46	62.76
Veer et al.	82.17	81.98	83.33	69.29	55.25
Welsch et al.	87.50	87.50	87.50	95.00	87.50
Yeoh et al.	69.18	70.13	71.97	71.08	76.39
Petricoin et al.	75.49	77.92	76.74	73.26	76.40
Pusztai et al.	55.95	59.67	58.39	52.79	32.00
Ranganathan et al.	44.36	52.98	50.18	55.67	56.00
Average	**66.74**	**68.96**	**67.61**	**69.66**	**68.49**

Table 6-2. The percent BACC of various rule learning methods using MDLPC. BGRL_G is the Bayesian Global Rule Learning with Greedy Search

Datasets	***BGRL_G***	***BGRL_B***	***BGRL_PG***	***C4.5***	***RL***
Alon et al.	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	76.73	77.90	86.43	85.42	91.41
Beer et al.	50.19	47.71	49.40	45.00	51.03
Bhattacharjee et al.	59.61	58.44	58.56	64.02	57.18
Bhattacharjee et al.	45.77	47.20	45.95	43.21	48.51
Golub et al.	65.21	63.43	62.06	67.43	66.27
Hedenfalk et al.	97.50	97.50	97.50	95.00	92.50
Iizuka et al.	42.50	48.33	39.58	43.33	46.08
Khan et al.	72.48	77.20	67.65	83.18	72.82
Nutt et al.	64.85	55.96	60.23	65.85	66.46
Pomeroy et al.	51.69	53.55	55.57	58.71	61.55
Pomeroy et al.	46.67	45.83	44.17	54.17	47.00
Ramaswamy et al.	56.14	57.40	58.16	66.41	59.47
Rosenwald et al.	53.92	57.08	52.80	53.82	56.74
Staunton et al.	56.65	58.95	57.30	54.25	53.28
Shipp et al.	58.17	58.17	66.25	60.65	50.15
Singh et al.	87.50	84.50	82.67	79.83	89.33
Su et al.	57.96	59.92	65.12	69.26	69.74
Veer et al.	80.48	84.93	83.39	66.32	59.50
Welsch et al.	48.75	48.75	48.75	56.25	50.00
Yeoh et al.	47.10	48.98	47.14	46.83	47.81
Petricoin et al.	51.81	54.97	52.73	51.98	56.94
Pusztai et al.	67.61	68.12	69.06	63.87	67.60
Ranganathan et al.	47.75	54.70	50.00	50.58	51.33
Average	**61.96**	**62.90**	**62.52**	**63.56**	**63.03**

Table 6-3. The percent RCI of various rule learning methods using MDLPC. BGRL_G is the Bayesian Global Rule Learning with Greedy Search.

Datasets	*BGRL_G*	*BGRL_B*	*BGRL_PG*	*C4.5*	*RL*
Alon et al.	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	30.78	28.04	46.12	68.50	52.60
Beer et al.	1.04	0.49	1.78	4.04	3.42
Bhattacharjee et al.	31.59	37.68	38.65	40.91	17.96
Bhattacharjee et al.	1.59	3.01	0.80	0.22	2.53
Golub et al.	42.05	42.68	32.70	53.84	51.22
Hedenfalk et al.	72.10	72.10	72.10	69.05	72.10
Iizuka et al.	0.24	0.19	2.46	0.50	1.10
Khan et al.	46.95	49.61	38.85	61.57	32.86
Nutt et al.	28.40	26.82	28.13	31.06	37.08
Pomeroy et al.	21.43	23.39	26.29	33.90	37.05
Pomeroy et al.	1.73	0.76	1.58	3.60	1.34
Ramaswamy et al.	20.29	54.35	57.74	61.44	71.58
Rosenwald et al.	15.10	25.10	14.29	1.22	2.33
Staunton et al.	30.92	31.65	30.86	32.15	32.45
Shipp et al.	19.85	19.23	30.58	14.82	4.92
Singh et al.	37.62	33.00	31.76	27.76	42.17
Su et al.	61.00	62.24	63.46	60.96	63.84
Veer et al.	26.69	45.98	47.84	23.05	3.12
Welsch et al.	23.27	23.27	23.27	23.27	23.27
Yeoh et al.	6.23	6.89	6.91	0.17	5.33
Petricoin et al.	1.21	1.89	1.44	3.32	2.54
Pusztai et al.	12.34	11.99	11.65	11.05	22.84
Ranganathan et al.	4.68	5.46	3.18	5.88	4.76
Average	**26.55**	**29.41**	**29.68**	**30.51**	**28.68**

Table 6-4. The percent accuracies of various rule learning methods using EBD. BGRL_G is the Bayesian Global Rule Learning with Greedy Search.

Datasets	*BGRL_G*	*BGRL_B*	*BGRL_PG*	*C4.5*	*RL*
Alon et al.	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	75.89	77.68	77.68	82.14	82.86
Beer et al.	71.81	70.97	70.69	68.89	75.56
Bhattacharjee et al.	64.79	72.81	67.26	85.69	77.13
Bhattacharjee et al.	58.10	62.86	54.52	62.62	60.71
Golub et al.	66.25	66.43	62.50	77.86	71.96
Hedenfalk et al.	97.50	97.50	97.50	95.00	94.17
Iizuka et al.	60.00	56.67	55.00	56.67	41.67
Khan et al.	69.39	73.22	66.76	80.83	88.33
Nutt et al.	62.00	50.00	42.00	54.00	62.00
Pomeroy et al.	76.67	74.97	78.89	70.00	83.00
Pomeroy et al.	58.33	56.67	58.33	60.00	53.33
Ramaswamy et al.	16.43	46.88	16.07	43.21	42.14
Rosenwald et al.	55.83	54.58	55.83	56.67	53.75
Staunton et al.	30.00	35.20	31.67	25.00	33.33
Shipp et al.	85.00	86.25	87.32	75.89	69.82
Singh et al.	80.64	82.45	81.91	78.36	89.18
Su et al.	38.09	71.98	48.29	59.71	70.72
Veer et al.	75.58	86.98	66.75	74.46	63.00
Welsch et al.	87.50	87.50	87.50	95.00	87.50
Yeoh et al.	67.53	72.17	70.71	71.50	71.16
Petricoin et al.	77.97	79.13	75.17	73.29	78.54
Pusztai et al.	54.07	57.20	55.32	54.67	40.40
Ranganathan et al.	48.00	57.11	50.36	53.33	51.82
Average	**66.01**	**70.14**	**65.06**	**68.95**	**69.95**

Table 6-5. The percent BACC of various rule learning methods using EBD. BGRL_G is the Bayesian Global Rule Learning with Greedy Search.

Datasets	*BGRL_G*	*BGRL_B*	*BGRL_PG*	*C4.5*	*RL*
Alon et al.	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	80.57	82.10	87.82	85.51	80.88
Beer et al.	47.21	46.79	52.14	46.25	47.32
Bhattacharjee et al.	60.99	62.98	56.30	64.10	60.12
Bhattacharjee et al.	44.98	44.94	41.73	43.21	43.45
Golub et al.	66.43	61.79	63.58	69.42	59.79
Hedenfalk et al.	97.50	97.50	97.50	95.00	97.50
Iizuka et al.	50.83	47.50	47.92	50.83	35.83
Khan et al.	68.37	79.99	68.85	81.12	85.80
Nutt et al.	69.65	62.67	62.46	66.79	71.67
Pomeroy et al.	54.88	55.93	54.24	56.20	58.21
Pomeroy et al.	49.17	45.83	50.00	52.50	50.33
Ramaswamy et al.	54.01	58.23	55.17	62.08	63.36
Rosenwald et al.	57.44	56.19	59.05	52.64	51.10
Staunton et al.	57.20	59.32	55.61	52.98	56.55
Shipp et al.	59.83	61.08	62.62	53.15	48.75
Singh et al.	85.33	85.83	81.75	78.33	90.33
Su et al.	57.35	65.98	57.44	67.35	73.82
Veer et al.	81.64	85.98	84.31	71.32	64.40
Welsch et al.	48.75	48.75	48.75	56.25	52.50
Yeoh et al.	47.21	49.13	47.36	46.76	48.35
Petricoin et al.	53.89	65.99	50.53	53.02	61.07
Pusztai et al.	68.22	66.12	66.13	66.30	73.12
Ranganathan et al.	48.42	56.51	50.67	50.92	51.83
Average	**62.91**	**64.46**	**62.58**	**63.42**	**63.59**

Table 6-6. The percent RCI of various rule learning methods using EBD. BGRL_G is the Bayesian Global Rule Learning with Greedy Search.

Datasets	*BGRL_G*	*BGRL_B*	*BGRL_PG*	*C4.5*	*RL*
Alon et al.	100.00	100.00	100.00	100.00	100.00
Armstrong et al.	51.15	46.53	46.46	69.42	54.30
Beer et al.	0.37	1.97	2.98	1.63	2.93
Bhattacharjee et al.	32.42	57.68	42.81	41.07	24.33
Bhattacharjee et al.	1.92	0.26	2.00	0.12	1.96
Golub et al.	38.25	39.89	29.67	53.33	49.56
Hedenfalk et al.	72.10	72.10	72.10	69.05	72.10
Iizuka et al.	0.79	0.86	0.90	1.06	0.80
Khan et al.	44.32	45.95	35.71	55.58	34.75
Nutt et al.	29.89	22.95	26.20	30.72	33.95
Pomeroy et al.	22.25	25.35	26.59	36.00	34.29
Pomeroy et al.	1.52	1.29	1.25	3.24	1.08
Ramaswamy et al.	19.78	58.13	17.30	54.73	72.51
Rosenwald et al.	26.11	32.95	29.08	0.72	2.61
Staunton et al.	31.18	31.87	30.56	34.51	39.61
Shipp et al.	17.98	19.68	19.83	8.59	3.33
Singh et al.	28.84	30.24	27.60	29.18	49.62
Su et al.	66.49	70.82	67.67	58.32	70.17
Veer et al.	24.67	46.12	35.71	21.60	3.92
Welsch et al.	23.27	23.27	23.27	23.27	23.27
Yeoh et al.	6.98	7.54	7.43	0.08	6.83
Petricoin et al.	5.66	13.32	6.54	0.81	9.28
Pusztai et al.	7.44	8.43	8.36	11.96	24.21
Ranganathan et al.	4.04	5.56	1.63	5.35	5.84
Average	**27.39**	**31.78**	**27.57**	**29.60**	**30.05**

B.2 THE QUALITATIVE RESULTS OF THE DIFFERENT MACHINE LEARNING ALGORITHMS

Table 6-7. The number of rules generated using MDLPC across the four Bayesian local structure algorithms with a maximum of 8 possible variables per a path within the *local structure*.

Bayesian Local Structure Search Strategy	*Decision Tree*		*Decision Graph*	
	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	2	2	2	10
Armstrong et al.	17	10	31	26
Beer et al.	13	7	18	34
Bhattacharjee et al.	47	48	51	49
Bhattacharjee et al.	14	10	26	12
Golub et al.	13	13	22	33
Hedenfalk et al.	2	2	2	13
Iizuka et al.	13	10	25	37
Khan et al.	18	19	13	51
Nutt et al.	12	17	35	49
Pomeroy et al.	18	13	36	87
Pomeroy et al.	15	12	19	11
Ramaswamy et al.	120	88	85	72
Rosenwald et al.	32	36	16	16
Staunton et al.	23	24	20	52
Shipp et al.	9	7	21	17
Singh et al.	16	27	18	46
Su et al.	73	69	70	63
Veer et al.	10	9	9	28
Welsch et al.	2	2	2	3
Yeoh et al.	52	34	25	29
Petricoin et al.	33	37	38	24
Pusztai et al.	21	25	31	22
Ranganathan et al.	11	11	13	11
Average	**24.417**	**22.167**	**26.177**	**33.125**

Table 6-8. The number of variables used in the model over the different methods using EBD with a maximum of 8 possible variables per a path within the *local structure.*

Bayesian Local Structure	*Decision Tree*		*Decision Graph*	
Search Strategy	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	2	2	2	15
Armstrong et al.	10	10	27	50
Beer et al.	17	13	18	44
Bhattacharjee et al.	33	32	20	37
Bhattacharjee et al.	17	10	21	13
Golub et al.	13	13	21	25
Hedenfalk et al.	2	2	2	17
Iizuka et al.	13	12	25	25
Khan et al.	25	25	27	45
Nutt et al.	15	37	26	56
Pomeroy et al.	22	20	45	46
Pomeroy et al.	15	11	20	21
Ramaswamy et al.	321	411	67	266
Rosenwald et al.	29	36	18	24
Staunton et al.	31	29	32	33
Shipp et al.	11	7	14	10
Singh et al.	17	19	22	38
Su et al.	139	117	97	101
Veer et al.	9	9	12	23
Welsch et al.	2	2	2	3
Yeoh et al.	61	57	20	25
Petricoin et al.	52	45	40	35
Pusztai et al.	30	30	35	63
Ranganathan et al.	11	9	15	16
Average	**37.375**	**39.917**	**26.167**	**42.958**

Table 6-9. The time it takes to run one fold of the classifier with EBD with a maximum of 8 possible variables per a path within the local structure.

Bayesian Local Structure	*Decision Tree*		*Decision Graph*	
Search Strategy	Greedy	Parallel Greedy	Greedy	Parallel Greedy
Alon et al.	0.65	3.77	0.66	5.37
Armstrong et al.	11.56	61.83	11.33	64.87
Beer et al.	0.14	0.68	0.14	0.78
Bhattacharjee et al.	0.03	0.07	0.03	0.08
Bhattacharjee et al.	1.09	12.21	1.10	13.07
Golub et al.	0.74	1.50	0.74	2.12
Hedenfalk et al.	0.11	0.36	0.11	0.41
Iizuka et al.	0.47	13.50	0.48	13.06
Khan et al.	13.70	37.76	13.84	55.84
Nutt et al.	3.01	25.27	3.02	25.71
Pomeroy et al.	0.06	0.14	0.06	0.15
Pomeroy et al.	0.07	0.48	0.07	0.35
Ramaswamy et al.	1.76	29.24	1.79	24.50
Rosenwald et al.	0.65	4.06	0.58	5.49
Staunton et al.	1.94	20.71	1.91	16.45
Shipp et al.	2.54	9.82	2.52	12.35
Singh et al.	0.95	1.22	0.94	1.28
Su et al.	26.42	329.86	28.40	251.84
Veer et al.	46.90	604.34	47.36	306.61
Welsch et al.	0.14	0.40	0.14	0.24
Yeoh et al.	102.71	2486.72	99.61	931.01
Petricoin et al.	0.44	8.56	0.43	6.29
Pusztai et al.	3.65	69.35	3.63	41.85
Ranganathan et al.	5.55	12.61	5.00	15.03
Average	**9.39**	**155.60**	**9.33**	**74.78**

Table 6-10. The average number of rules over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using MDLPC.

Rule Learning	*Bayesian Global Structure*			C4.5	RL
Search Strategy	Greedy	Beam Search	Parallel Greedy		
Alon et al.	2.00	2.00	2.00	2.00	2.00
Armstrong et al.	16.00	16.00	8.00	4.00	10.00
Beer et al.	16.00	16.00	32.00	5.00	20.00
Bhattacharjee et al.	360.00	360.00	72.00	11.00	23.00
Bhattacharjee et al.	128.00	128.00	256.00	7.00	16.00
Golub et al.	16.00	16.00	16.00	7.00	13.00
Hedenfalk et al.	2.00	2.00	2.00	2.00	2.00
Iizuka et al.	16.00	16.00	32.00	6.00	19.00
Khan et al.	32.00	32.00	32.00	9.00	13.00
Nutt et al.	12.00	24.00	16.00	6.00	18.00
Pomeroy et al.	192.00	288.00	128.00	9.00	35.00
Pomeroy et al.	32.00	32.00	16.00	6.00	23.00
Ramaswamy et al.	1152.00	1196.00	1296.00	82.00	106.00
Rosenwald et al.	256.00	256.00	256.00	26.00	56.00
Staunton et al.	64.00	64.00	256.00	15.00	27.00
Shipp et al.	8.00	8.00	4.00	5.00	9.00
Singh et al.	48.00	48.00	24.00	7.00	22.00
Su et al.	1152.00	418.00	576.00	21.00	47.00
Veer et al.	16.00	16.00	16.00	5.00	37.00
Welsch et al.	2.00	2.00	2.00	2.00	33.00
Yeoh et al.	256.00	256.00	256.00	18.00	41.00
Petricoin et al.	256.00	138.00	384.00	20.00	79.00
Pusztai et al.	64.00	64.00	128.00	29.00	8.00
Ranganathan et al.	8.00	8.00	8.00	4.00	9.00
Average	**171.08**	**141.92**	**159.08**	**12.83**	**27.83**

Table 6-11. The average number of rules over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using EBD.

Rule Learning	*Bayesian Global Structure*			C4.5	RL
Search Strategy	Greedy	Beam Search	Parallel Greedy		
Alon et al.	2.00	2.00	2.00	2.00	2.00
Armstrong et al.	8.00	12.00	12.00	4.00	10.00
Beer et al.	16.00	32.00	24.00	6.00	21.00
Bhattacharjee et al.	48.00	96.00	144.00	9.00	24.00
Bhattacharjee et al.	64.00	64.00	16.00	8.00	19.00
Golub et al.	32.00	32.00	64.00	7.00	14.00
Hedenfalk et al.	2.00	2.00	2.00	2.00	2.00
Iizuka et al.	16.00	16.00	16.00	6.00	19.00
Khan et al.	32.00	32.00	32.00	9.00	10.00
Nutt et al.	12.00	24.00	24.00	6.00	16.00
Pomeroy et al.	192.00	192.00	256.00	8.00	63.00
Pomeroy et al.	16.00	16.00	16.00	6.00	10.00
Ramaswamy et al.	1152.00	562.00	576.00	109.00	120.00
Rosenwald et al.	256.00	256.00	256.00	31.00	53.00
Staunton et al.	96.00	96.00	256.00	14.00	28.00
Shipp et al.	8.00	8.00	18.00	5.00	25.00
Singh et al.	16.00	24.00	16.00	7.00	20.00
Su et al.	1304.00	1532.00	1944.00	31.00	55.00
Veer et al.	16.00	16.00	8.00	5.00	34.00
Welsch et al.	2.00	2.00	2.00	2.00	2.00
Yeoh et al.	256.00	256.00	256.00	19.00	36.00
Petricoin et al.	640.00	284.00	384.00	30.00	75.00
Pusztai et al.	64.00	144.00	384.00	34.00	12.00
Ranganathan et al.	12.00	12.00	8.00	5.00	7.00
Average	**177.58**	**154.67**	**196.50**	**15.21**	**28.21**

Table 6-12. The average number of variables over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using MDLPC.

Rule Learning Search Strategy	*Bayesian Global Structure*			C4.5	RL
	Greedy	Beam Search	Parallel Greedy		
Alon et al.	1.00	1.00	1.00	1.00	1.00
Armstrong et al.	3.00	3.00	3.00	3.00	10.00
Beer et al.	4.00	4.00	5.00	4.00	20.00
Bhattacharjee et al.	7.00	7.00	8.00	6.00	16.00
Bhattacharjee et al.	4.00	4.00	4.00	6.00	12.00
Golub et al.	1.00	1.00	1.00	1.00	1.00
Hedenfalk et al.	4.00	4.00	5.00	5.00	19.00
Iizuka et al.	5.00	5.00	5.00	7.00	13.00
Khan et al.	3.00	4.00	4.00	5.00	18.00
Nutt et al.	5.00	5.00	4.00	5.00	23.00
Pomeroy et al.	6.00	6.00	8.00	13.50	26.00
Pomeroy et al.	3.00	3.00	2.00	4.00	9.00
Ramaswamy et al.	5.00	5.00	4.00	6.00	22.00
Rosenwald et al.	1.00	1.00	1.00	1.00	33.00
Staunton et al.	7.00	7.00	7.00	8.00	32.00
Shipp et al.	8.00	8.00	8.00	25.00	61.00
Singh et al.	4.00	4.00	4.00	4.00	37.00
Su et al.	8.00	8.00	8.00	19.00	45.00
Veer et al.	8.00	8.00	8.00	17.00	46.00
Welsch et al.	6.00	6.00	5.00	9.00	23.00
Yeoh et al.	8.00	8.00	8.00	74.00	93.00
Petricoin et al.	6.00	6.00	7.00	28.00	13.00
Pusztai et al.	8.00	8.00	8.00	18.50	82.00
Ranganathan et al.	3.00	3.00	3.00	3.00	9.00
Average	**4.92**	**4.96**	**5.04**	**11.38**	**27.67**

Table 6-13. The average number of variables over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using EBD.

Rule Learning Search Strategy	*Bayesian Global Structure* Greedy	Beam Search	Parallel Greedy	C4.5	RL
Alon et al.	1.00	1.00	1.00	1.00	1.00
Armstrong et al.	3.00	3.00	3.00	3.00	10.00
Beer et al.	4.00	5.00	4.00	5.00	20.00
Bhattacharjee et al.	6.00	6.00	4.00	7.00	18.00
Bhattacharjee et al.	5.00	5.00	6.00	6.00	15.00
Golub et al.	1.00	1.00	1.00	1.00	2.00
Hedenfalk et al.	4.00	4.00	4.00	5.00	19.00
Iizuka et al.	5.00	5.00	5.00	7.00	10.00
Khan et al.	3.00	4.00	4.00	5.00	15.00
Nutt et al.	4.00	4.00	4.00	5.00	10.00
Pomeroy et al.	6.00	6.00	8.00	12.00	27.00
Pomeroy et al.	3.00	3.00	3.00	4.00	24.00
Ramaswamy et al.	4.00	4.00	4.00	8.00	20.00
Rosenwald et al.	1.00	1.00	1.00	1.00	2.00
Staunton et al.	7.00	7.00	8.00	7.00	63.00
Shipp et al.	8.00	8.00	8.00	30.00	70.00
Singh et al.	4.00	4.00	3.00	4.00	34.00
Su et al.	8.00	7.00	8.00	26.00	55.00
Veer et al.	8.00	8.00	8.00	18.00	35.00
Welsch et al.	5.00	6.00	6.00	15.00	24.00
Yeoh et al.	8.00	8.00	8.00	88.50	120.00
Petricoin et al.	6.00	6.00	8.00	32.00	21.00
Pusztai et al.	8.00	8.00	8.00	27.00	74.00
Ranganathan et al.	3.00	3.00	3.00	7.00	7.00
Average	**4.79**	**4.88**	**5.00**	**13.52**	**29.00**

Table 6-14. The run time(in minutes) for a single fold averaged over all folds for the Bayesian global rule learning algorithms (with a maximum of 8 possible parents), C4.5, and RL using MDLPC.

Rule Learning Search Strategy	*Bayesian Global Structure*			C4.5	RL
	Greedy	Beam Search	Parallel Greedy		
Alon et al.	0.00	0.20	0.61	0.00	2.73
Armstrong et al.	0.01	5.10	5.91	0.00	8.51
Beer et al.	0.00	0.02	0.11	0.00	1.33
Bhattacharjee et al.	0.00	0.00	0.00	0.00	0.37
Bhattacharjee et al.	0.00	0.14	0.93	0.00	5.40
Golub et al.	0.00	0.13	0.41	0.00	1.92
Hedenfalk et al.	0.00	0.01	0.03	0.00	0.41
Iizuka et al.	0.00	0.11	0.83	0.00	4.71
Khan et al.	0.01	4.44	6.01	0.00	7.48
Nutt et al.	0.01	1.35	6.47	0.00	11.63
Pomeroy et al.	0.00	0.00	0.01	0.00	0.42
Pomeroy et al.	0.00	0.12	0.04	0.00	3.89
Ramaswamy et al.	0.00	0.01	0.05	0.00	3.19
Rosenwald et al.	0.00	0.21	0.99	0.00	3.35
Staunton et al.	0.01	1.17	4.71	0.00	4.78
Shipp et al.	0.00	0.68	2.57	0.00	3.94
Singh et al.	0.00	0.12	0.37	0.00	2.74
Su et al.	0.12	54.24	78.02	0.01	132.60
Veer et al.	0.04	14.43	17.33	0.01	15.80
Welsch et al.	0.00	0.10	0.01	0.00	1.49
Yeoh et al.	0.06	20.89	48.13	0.01	34.44
Petricoin et al.	0.00	0.10	0.73	0.00	13.59
Pusztai et al.	0.03	1.46	12.13	0.01	30.30
Ranganathan et al.	0.00	0.51	1.58	0.00	2.87
Average	**0.01**	**4.40**	**7.83**	**0.00**	**12.41**

Table 6-15. The run time(in minutes) for a single fold averaged over all folds for the Bayesian global rule learning algorithms, C4.5, and RL using EBD with a maximum of 8 possible parents in minutes.

Rule Learning Search Strategy	*Bayesian Global Structure*			C4.5	RL
	Greedy	Beam Search	Parallel Greedy		
Alon et al.	0.00	0.25	0.67	0.00	2.90
Armstrong et al.	0.02	9.60	8.25	0.00	12.46
Beer et al.	0.00	0.04	0.26	0.00	2.86
Bhattacharjee et al.	0.00	0.00	0.01	0.00	0.46
Bhattacharjee et al.	0.00	0.76	3.47	0.00	8.29
Golub et al.	0.00	0.17	0.48	0.00	2.17
Hedenfalk et al.	0.00	0.01	0.09	0.00	0.89
Iizuka et al.	0.00	0.35	2.61	0.00	5.75
Khan et al.	0.01	11.93	10.77	0.00	12.58
Nutt et al.	0.02	4.03	11.73	0.00	22.76
Pomeroy et al.	0.00	0.00	0.02	0.00	0.53
Pomeroy et al.	0.00	0.33	0.14	0.00	1.73
Ramaswamy et al.	0.01	1.80	8.89	0.00	9.20
Rosenwald et al.	0.00	0.27	1.27	0.00	3.71
Staunton et al.	0.01	1.37	4.49	0.00	5.42
Shipp et al.	0.01	0.92	3.03	0.00	4.53
Singh et al.	0.00	0.30	0.57	0.00	2.84
Su et al.	0.09	34.93	76.23	0.01	198.52
Veer et al.	0.25	4.77	104.70	0.02	195.02
Welsch et al.	0.00	10.94	0.06	0.00	1.57
Yeoh et al.	0.73	83.73	681.93	0.07	814.49
Petricoin et al.	0.00	0.26	2.04	0.00	14.64
Pusztai et al.	0.03	4.28	14.70	0.01	32.30
Ranganathan et al.	0.00	1.13	2.50	0.00	3.66
Average	**0.05**	**7.17**	**39.12**	**0.01**	**56.64**

APPENDIX C

COMMAND LINE PROCEDURES FOR BRL

The template command line is:

Java –Xmx[Memory] –jar nbrl.jar –lp [learning parameters] –ppp [PreProcessing methods] –dp [data input parameters]

Sample command line:
For analysis of RL with EBD Discretization with no validation splits:
java –Xmx1300m –jar nbrl.jar –lp –rgm 0 –bss --d 0 --f 10 –d 7 1 –dp Dataset.txt
Analysis of RL with EBD with 10 fold Validation and 5-fold internal for parameter learning:
java –Xmx1300m –jar nbrl.jar –lp –rgm 0 –bss --d 0 --f 5 –cfv 10 –d 7 1 –dp Dataset.txt

Analysis using BLS-RL Parallel Greedy Decision Graph with EBD with 10-CFV
java –Xmx1300m –jar nbrl.jar –lp –rgm 2 5 –bss --f 10 –d 7 1 –dp Dataset.txt

- Since there is currently no parameter learning for the BLS-RL you can use the internal cross-fold as regular cross-fold validation

For Learning parameters

Input Type	Input Style	Parameter 1	Parameter 2
Rule Generation Mechanism	-rgm p1 p2	Learning Method -rgm 0: Rule Learner -rgm 1: Bayesian Global Rule Learning -rgm 2: Bayesian Local Structure RL	-rgm 0: No parameter 2 -rgm 1: -rgm 1 0: Greedy Search (GS) -rgm 1 1: Beam Search (BS) -rgm 1 2: Parallel Greedy Search (PGS) -rgm 2: -rgm 2 0: Decision Tree(DT) GS -rgm 2 1: DT BS -rgm 2 2: DT PGS -rgm 2 3: Decision Graph(DG) GS -rgm 2 4: DG BS -rgm 2 5: DG PGS
Bias Space Search	-bss --d X --f X	--d : Bias Space Depth For Rule Learning: --d 0: shallow (short – default) --d 1: medium --d 2: deep (takes a long time may run out of memory)	--f : Number of folds within bss Range: 1 – n Where n is the number of samples in dataset

Cross Fold Validation	-cfv p1	The number of folds for validation Range: 1 – n where n is the number of samples	
Random Resample Validation	-rrv p1 p2	The number of random resample validation sets to create Range: 1-1000	The percent size of the random resample validation. Range: 1% - 70%

For Preprocessing parameters

Input Type	Input Style	Parameter 1	Parameter 2
Chi Square Trim	-chi p1	Chi square level Range: 0 - ∞	None
Scale Data Univariate	-s p1,p1', p1*, …	Scale Methods: 0: 0-1 Scaling 1: Subtract Local Min 2: Subtract Global Min 3: Log 2 Transform 4: Square Root 5: exponent 2 6: Square 7: Normalize (μ=0,σ=1)	None
Remove Single Interval Discs via EBD discretization	-r	None	None
Discretize	-d p1 p2	Same as those in Learning parameters	Same as those in Learning parameters
Discretize and remove single intervals	-dr p1 p2	Same as those in Learning parameters	Same as those in Learning parameters
Combine Tech Replicates - Samples must have same name with a (#) next to it	-ctr	None	None
Create Independent Test	-itst p1	Percent size of test set	None
Create Random Resample Validation	-rrv p1 p2	Same as those in Learning parameters	Same as those in Learning parameters

For Data parameters

Input Type	Input Style	Parameter 1
Data output type	-o p1	-o arff: Variable Relation File Format -o csv: Comma Separated Values NO –o is a default of tab delimited
Random Seed	-rand p1	The default is 1, any specified changes the fold splits Standardizes fold generation so you don't need to keep GBs of folds
Input training file is format of CSV	-itrncsv	None
Input test file is format of CSV	-itstcsv	None
File is transposed: Variables as rows, Samples as columns	-t	None
Testing file is a directory of ASCII files where one column is variable the second column is values	-dtst p1	The Directory containing the training file - Within the directory files grouped by class folder - E.g., Within Dataset, there is a DISEASE folder and a CNTRL folder
Testing File	-tst p1	The Test File
Output Directory	-odir p1	The output directory to print all the results
Training File is a directory of ASCII files where one column is variable the second column is values	-dtr p1	The Directory containing the training file - Within the directory files grouped by class folder - E.g., Within Dataset, there is a DISEASE folder and a CNTRL folder
If no –dtr parameter last value is the training dataset	p1	The training file

PARAMETERS that allow no learning and no other parameters including the last one

Convert file from CSV to Tab or TAB to CSV	-c p1	The file to convert
Transpose the file	-tpf p1	The file to transpose

REFERENCES

1. Gopalakrishnan, V., et al., *Proteomic data mining challenges in identification of disease-specific biomarkers from variable resolution mass spectra.* Proceedings of SIAM Bioinformatics Workshop, 2004: p. 1-10.
2. Gopalakrishnan, V., et al., *Rule learning for disease-specific biomarker discovery from clinical proteomic mass spectra.* Springer Lecture Notes in Computer Science, 2006. **3916**: p. 93-105.
3. The UniProt, C., *The Universal Protein Resource (UniProt).* Nucl. Acids Res., 2007. **35**(suppl_1): p. D193-197.
4. Ressom, H.W., et al. *Analysis of MALDI-TOF Spectrometry Data for Detection of Glycan Biomarkers*. In: *Proceedings of the Pacific Symposium on Biocomputing*. Hawaii January 4-8, 2008. 216-227.
5. Vanhoutte, K.J.A., et al., *Biomarker discovery with SELDI-TOF MS in human urine associated with early renal injury: evaluation with computational analytical tools.* Nephrol. Dial. Transplant., 2007: p. gfm170.
6. Ranganathan, S., et al., *Proteomic profiling of cerebrospinal fluid identifies biomarkers for amyotrophic lateral sclerosis.* Journal of Neurochemistry, 2005. **95**(5): p. 1461-71.
7. Armstrong, S.A., et al., *MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia.* Nat Genet, 2002. **30**(1): p. 41-7.
8. Shipp, M.A., et al., *Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning.* Nat Med, 2002. **8**(1): p. 68-74.
9. Singh, D., et al., *Gene expression correlates of clinical prostate cancer behavior.* Cancer Cell, 2002. **1**(2): p. 203-9.
10. George Chambers, L.L.P.C.G.I.M., *Proteomics: a new approach to the study of disease.* The Journal of Pathology, 2000. **192**(3): p. 280-288.
11. Martin, D.B. and P.S. Nelson, *From genomics to proteomics: techniques and applications in cancer research.* Trends in Cell Biology, 2001. **11**(11): p. S60-S65.
12. Pandey, A. and M. Mann, *Proteomics to study genes and genomes.* Nature, 2000. **405**(6788): p. 837-846.
13. Ressom, H.W., et al. *Biomarker identification and rule extraction from mass spectral serum profiles*. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Bioinformatics and Compuational Biology*. Toronto, Ontario, Canada September 28-29, 2006. 1-7.
14. Heckerman, D. and E.J. Horvitz. *On the expressiveness of rule-based systems for reasoning under uncertainty*. in *Proceedings of the National Conference on Artificial Intelligence*. 1987. Palo Alto, CA.

15. Janssens, D., et al., *Integrating Bayesian networks and decision trees in a sequential rule-based transportation model.* European Journal of Operational Research, 2006. **175**(1): p. 16-34.
16. Carvalho, D.R. and A.A. Freitas, *A hybrid decision tree/genetic algorithm method for data mining.* Information Sciences, 2004. **163**(1-3): p. 13-35.
17. Livingston, G.R., J.M. Rosenberg, and B.G. Buchanan, *A framework for autonomously performing knowledge discovery in databases*. 2000.
18. Provost, F., J.M. Aronis, and B.G. Buchanan, *Rule-space search for knowledge-based discovery*, in *CIIO Working Paper IS 99-012*. 1999, Stern School of Business, New York University: NY, NY 10012.
19. Domingos, P. *The RISE system: conquering without separating*. in *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on*. 1994. New Orleans, LA, USA.
20. Clearwater, S.H. and F.J. Provost. *RL4: A tool for knowledge-based induction*. in *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence (TAI-90)*. 1990. Herndon, VA.
21. Quinlan, J.R., *Induction of Decision Trees.* Machine Learning, 1986(1): p. 81-106.
22. Cohen, W.W. *Fast effective rule induction*. in *Proceedings of the Twelfth International Conference on Machine Learning*. 1995. Tahoe City, CA: Morgan Kaufmann.
23. Hennessy, D., et al. *Induction of rules for biological macromolecule crystallization*. in *Proceedings of Second International Conference on Intelligent Systems for Molecular Biology*. 1994. Standford, Ca: AAAI Press.
24. Zhou, S. and K. Wang, *Localization Site Prediction for Membrane Proteins by Integrating Rule and SVM Classification.* IEEE Transactions on Knowledge and Data Engineering, 2005.
25. Frey, L., et al., *Using prior knowledge and rule induction methods to discover molecular markers of prognosis in lung cancer.* AMIA Annu Symp Proc, 2005: p. 256-60.
26. Xiuju, F., et al. *Extracting the knowledge embedded in support vector machines*. In: *Proceedings of the IEEE International Joint Conference on Neural Networks.* Budapest, Hungary July 25-29, 2004. 291-296.
27. Aronis, J.M. and F.J. Provost. *Increasing the efficiency of data mining algorithms with breadth-first marker propagation*. in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. 1997. Newport, CA.
28. Lustgarten, J.L., et al. *An evaluation of discretization methods for learning rules from biomedical datasets*. In: *Proceedings of the International Conference on Bioinformatics and Computational Biology (BIOCOMP-08)*. Las Vegas, NV 2008.
29. Cunningham, M.J., *Genomics and proteomics The new millennium of drug discovery and development.* Journal of Pharmacological and Toxicological Methods, 2000. **44**(1): p. 291-300.
30. Tyers, M. and M. Mann, *From genomics to proteomics.* Nature, 2003. **422**(6928): p. 193-197.
31. Edwards, A.M., C.H. Arrowsmith, and B. Pallieres, *Proteomics: New tools for a new era.* MODERN DRUG DISCOVERY, 2000. **3**: p. 34-45.
32. Kitano, H., *Computational systems biology.* Nature, 2002. **420**(6912): p. 206-210.
33. Brown, M.P.S., et al., *Knowledge-based analysis of microarray gene expression data by using support vector machines.* Proc. Natl. Acad. Sci., 2000. **97**: p. 262-267.

34. Allen, J.F., *Bioinformatics and discovery: induction beckons again.* BioEssays, 2001. **23**(1): p. 104-107.
35. Larranaga, P., et al., *Machine learning in bioinformatics.* Briefings in Bioinformatics, 2006. **7**(1): p. 86-112.
36. Lucas, P., *Certainty-factor-like structures in Bayesian belief networks.* Knowledge-Based Systems, 2001. **14**: p. 327-335.
37. Mahoney, J.J. and R.J. Mooney. *Comparing methods for refining certainty-factor rule-bases*. in *International Conference on Machine Learning*. 1994.
38. Heckerman, D., *Probabilistic interpretation for MYCIN's certainty factor.* Uncertainty in Artificial Intelligence, 1985: p. 167-196.
39. Freitas, A.A., *On rule interestingness measures.* Knowledge-Based Systems, 1999. **12**(5-6): p. 309-315.
40. Towell, G.G. and J.W. Shavlik, *Extracting refined rules from Knowledge-Based neural networks.* Machine Learning, 1993. **13**: p. 71-10.
41. Lustgarten, J.L., et al. *Improving classification performance with discretization on biomedical datasets*. In: *Proceedings of the AMIA Annual Symposium*. Washington, DC November 8-12, 2008. 445-449.
42. Lustgarten, J.L., *Machine learning with supervised discretization aids 'Omic'-data analysis*, in *Biomedical Informatics*. 2006, University of Pittsburgh: Pittsburgh.
43. Boulle, M., *A Bayesian approach for supervised discretization.* Data Mining, 2004.
44. Yang, Y. and G.I. Webb, *Discretization for Naive Bayes learning: managing discretization bias and variance*. 2003, Monash University. p. 131.
45. Yang, Y. and G. Webb, *On why discretization works for Naive-Bayes classifiers.* Lecture Notes in Computer Science, 2003. **2903**: p. 440-452.
46. Liu, H., et al., *Discretization: An enabling technique.* Data Mining and Knowledge Discovery, 2002. **6**: p. 393-423.
47. Kohavi, R. and M. Sahami. *Error-Based and Entropy-Based discretization of continuous features*. in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 1996. Portland, Oregon: AAAI Press.
48. Dougherty, J., R. Kohavi, and M. Sahami. *Supervised and Unsupervised discretization of continuous features*. In: *Proceedings of the the Twelfth International Conference Machine Learning*. Tahoe City, California, USA July 9-12, 1995. 194-202.
49. Fayyad, U.M. and K.B. Irani. *Multi-interval discretization of continuous-valued attributes for classification learning*. in *Proceedings of the Thirteenth International Joint Conference on AI (IJCAI-93)*. 1993. Chamberry, France.
50. Dan, Q. and J. Dudeck, *Certainty factor theory and its implementation in a medical expert system shell.* Med Inform (Lond), 1992. **17**(2): p. 87-103.
51. Fayyad, U.M., G. Piatelsky-Shapiro, and P. Smyth, *From data mining to knowledge discovery: An overview*, in *Advances in Knowledge Discovery and Data Mining*, e.a. U.M Fayyad, Editor. 1996, AAAI Press/The MIT Press: Menlow Park, CA. p. 1-34.
52. Ting, K.M. *The problem of small disjuncts: Its remedy in decision trees*. in *Tenth Canadian Conference on Artificial Intelligence*. 1994.
53. Weiss, G.M. *Learning with Rare Cases and Small Disjuncts*. in *International Conference on Machine Learning*. 1995. Tahoe City, California, USA: Morgan Kaufman.
54. Carvalho, D.R. and A.A. Freitas, *A genetic-algorithm for discovering small-disjunct rules in data mining.* Applied Software Computing, 2002. **2**(2): p. 75-88.

55. Mahoney, J.J. and R.K. Mooney, *Combining nerual and symbolic learning to revise probabilistic rule bases.* Advances in Neural Information Processing Systems, 1993. **5**: p. 107-114.
56. Hruschka, E.R.J., et al. *Markov-Blanket based strategy for translating a bayesian classifier into a reduced set of classification rules*. in *Seventh International Conference on Hybrid Intelligent Systems*. 2007. Kaiserslautern, Germany.
57. Liu, H. and R. Setiono, *Chi2: Feature selection and discretization of numeric attributes.* Proceedings of IEE 7th International Conference on Tools with Artificial Intelligence, 1995.
58. Kerber, R. *ChiMerge: Discretization of numeric attributes* In: *Proceedings of the International Conference of Artificial Intelligence*. Arlington, VA November 10-13, 1992. 123-128.
59. Liu, H. and R. Setiono, *Feature selection via discretization.* Knowledge and Data Engineering, 1997. **9**(4): p. 642-645.
60. Butterworth, R., et al., *A greedy algorithm for supervised discretization.* J. of Biomedical Informatics, 2004. **37**(4): p. 285-292.
61. Boullé, M., *MODL: A Bayes optimal discretization method for continuous attributes.* Machine Learning, 2006. **65**(1): p. 131-165.
62. Heckerman, D., D. Geiger, and D.M. Chickering, *Learning Bayesian networks: The combination of knowledge and statistical data.* Machine Learning, 1995. **20**(3): p. 197-243.
63. Neapolitan, R.E., *Learning Bayesian Networks*. 10 ed. Prentice Hall Series in Artificial Intelligence, ed. S. Russell and P. Norvig. 2004, Upper Saddle River: Alan Apt. 674.
64. Towell, G.G., *Symbolic Knowledge and Nerual Networks: Insertion, Refinement, and Extraction*, in *Computer Science*. 1992, University of Wisconsin-Madison: Madison.
65. Boutilier, C., et al. *Context-specific independence in bayesian networks*. in *Uncertainty in Artificial Intelligence, Proceedings of the Twelfth Annual Conference*. 1996. Portland Oregon.
66. desJardins, M., P. Rathod, and L. Getoor, *Bayesian network learning with abstraction hierarchies and context-specific independence*, in *Machine Learning: ECML 2005*. 2005. p. 485-496.
67. Chickering, D.M., D. Heckerman, and C. Meek. *A Bayesian approach to learning bayesian networks with local structure*. in *13th Conf. on Uncertainty in Artificial Intelligence*. 1997.
68. Friedman, N. and M. Goldszmidt, *Learning Bayesian networks with local structure*, in *Learning in Graphical Models*. 1996, MIT Cambridge, MA. p. 252-262.
69. Heckerman, D., D. Geiger, and D.M. Chickering, *Learning Bayesian Networks - the Combination of Knowledge and Statistical Data.* Machine Learning, 1995. **20**(3): p. 197-243.
70. Cooper, G.F. and E. Herskovits, *A Bayesian Method for the Induction of Probabilistic Networks from Data.* Machine Learning, 1992. **9**(4): p. 309-347.
71. Alam, R. and M. Gorska, *Genomic microarrays: Arraying order in biological chaos?* Am. J. Respir. Cell Mol. Biol., 2001. **25**(4): p. 405-408.
72. Quackenbush, J., *Computational analysis of microarray data.* Nature Review Genetics, 2001. **2**(6): p. 418-427.

73. Altman, R.B. and S. Raychaudhuri, *Whole-genome expression analysis: challenges beyond clustering.* Current Opinion in Structural Biology, 2001. **11**(3): p. 340-347.
74. Lockhart, D.J. and E.A. Winzeler, *Genomics, gene expression and DNA arrays.* Nature, 2000. **405**: p. 827-836.
75. Brown, P.O. and D. Botstein, *Exploring the new world of the genome with DNA microarrays.* Nature Genetics, 1999. **21**(1 Suppl): p. 33-37.
76. DeRisi, J.L., V.R. Iyer, and P.O. Brown, *Exploring the metabolic and genetic control of gene expression on a genomic scale.* Science, 1997. **278**(5338): p. 680-686.
77. Greenbaum, D., et al., *Interrelating different types of genomic data, from Proteome to Secretome:'Oming in on function.* Genome Research, 2001. **11**(9): p. 1463-1468.
78. Ramaswamy, S., et al., *A molecular signature of metastasis in primary solid tumors.* Nature Genetics, 2003. **33**(1): p. 49-54.
79. Rosenwald, A., et al., *The use of molecular profiling to predict survival after chemotherapy for diffuse Large-B-Cell Lymphoma.* N Engl J Med, 2002. **346**(25): p. 1937-1947.
80. To, C.C. and J. Vohradsky, *A parallel genetic algorithm for single class pattern classification and its application for gene expression profiling in Streptomyces coelicolor.* BMC Genomics, 2007. **8**: p. 49.
81. Welsh, J.B., et al., *Analysis of gene expression profiles in normal and neoplastic ovarian tissue samples identifies candidate molecular markers of epithelial ovarian cancer.* Proc Natl Acad Sci U S A, 2001. **98**(3): p. 1176-81.
82. McKusick, V.A., *Mendelian inheritance in man. A catalog of human genes and genetic disorders*. 12 ed. 1998, Baltimore: Johns Hopkins University Press.
83. Golub, T.R., et al., *Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.* Science, 1999. **286**(5439): p. 531-537.
84. Hedenfalk, I., et al., *Gene-expression profiles in hereditary breast cancer.* New England Journal of Medicine., 2001. **344**(8): p. 539-48.
85. Bhattacharjee, A., et al., *Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses.* Proc Natl Acad Sci U S A, 2001. **98**(24): p. 13790-5.
86. Khan, J., et al., *Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks.* Nat Med, 2001. **7**(6): p. 673-9.
87. Su, A.I., et al., *Molecular classification of human carcinomas by use of gene expression signatures.* Cancer Res, 2001. **61**(20): p. 7388-93.
88. Okabe, H., et al., *Genome-wide analysis of gene expression in human hepatocellular carcinomas using cDNA microarray: Identification of genes involved in viral carcinogenesis and tumor progression.* Cancer Research, 2001. **61**(5): p. 2129-2137.
89. Ramaswamy, S., et al., *Multiclass cancer diagnosis using tumor gene expression signatures.* Proc Natl Acad Sci U S A, 2001. **98**(26): p. 15149-54.
90. Wilson, C.L. and C.J. Miller, *Simpleaffy: a BioConductor package for Affymetrix Quality Control and data analysis.* Bioinformatics, 2005. **21**(18): p. 3683-3685.
91. Iizuka, N., et al., *Oligonucleotide microarray for prediction of early intrahepatic recurrence of hepatocellular carcinoma after curative resection.* Lancet, 2003. **361**(9361): p. 923-929.
92. Beer, D.G., et al., *Gene-expression profiles predict survival of patients with lung adenocarcinoma.* Nat Med, 2002. **8**(8): p. 816-824.

93. van 't Veer, L.J., et al., *Gene expression profiling predicts clinical outcome of breast cancer.* 2002. **415**(6871): p. 530-536.
94. Yeoh, E.-J., et al., *Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling.* Cancer Cell, 2002. **1**(2): p. 133-143.
95. Wang, C.C. and C.L. Tsou, *Post-genome Study----Proteomics.* Sheng Wu Hua Xue Yu Sheng Wu Wu Li Xue Bao (Shanghai), 1998. **30**(6): p. 533-539.
96. Issaq, H.J., et al., *The SELDI-TOF MS approach to proteomics: Protein profiling and biomarker identification.* Biochemical and Biophysical Research Communications, 2002. **292**(3): p. 587-592.
97. Hanash, S., *Disease proteomics.* Nature, 2003. **422**(6928): p. 226-232.
98. Lyons-Weiler, J., et al., *Assessing the statistical significance of the achieved classification error of classifiers constructed using serum peptide profiles, and a prescription for random sampling repeated studies for massive high-throughput genomic and proteomic studies.* Cancer Informatics, 2005. **1**(1): p. 53-77.
99. Pelikan, R., et al., *Intersession reproducibility of mass spectrometry profiles and its effect on accuracy of multivariate classification models.* Bioinformatics, 2007. **23**(22): p. 3065-72.
100. Harry, J.L., et al., *Proteomics: Capacity versus utility.* Electrophoresis, 2000. **21**: p. 1071-1081.
101. Stone, J.H., et al., *A serum proteomic approach to gauging the state of remission in Wegener's granulomatosis.* Arthritis & Rheumatism, 2005. **52**(3): p. 902-910.
102. Tolson, J., et al., *Serum protein profiling by SELDI mass spectrometry: detection of multiple variants of serum amyloid alpha in renal cancer patients.* Lab Invest, 2004. **84**(7): p. 845-856.
103. Petricoin, E.F., III, et al., *Serum proteomic patterns for detection of prostate cancer.* Journal of National Cancer Institute, 2002. **94**(20): p. 1576-1578.
104. Petricoin, E.F., III, et al., *Use of proteomic patterns in serum to identify ovarian cancer.* Lancet, 2002. **359**: p. 572-577.
105. Tiss, A., et al., *Serum Peptide Profiling using MALDI Mass Spectrometry.* Proteomics, 2007. **7**(S1): p. 77-89.
106. Hauskrecht, M., et al., *Feature selection for classification of SELDI-TOF-MS proteomic profiles.* Applied Bioinformatics, 2005. **4**(4): p. 227-246.
107. George L. Wright, J., et al., *ProteinChip surface enhanced laser desosrption/ionization (SELDI) mass spectrometry: A novel protein biochip technology for detection of prostate cancer biomarkers in complex protein mixtures.* Prostate Cancer & Prostate Diseases, 1999. **2**: p. 264-276.
108. Wong, J.W., G. Cagney, and H.M. Cartwright, *SpecAlign--processing and alignment of mass spectra datasets.* Bioinformatics, 2005. **21**(9): p. 2088-90.
109. Wong, J.W., C. Durante, and H.M. Cartwright, *Application of fast Fourier transform cross-correlation for the alignment of large chromatographic and spectral datasets.* Anal Chem, 2005. **77**(17): p. 5655-61.
110. Wong, J.W.H., et al. *Modeling and resolving overlapping peaks for accurate quantitation of mass spectrometry biomarker signals*. In: *Proceedings of the International Mass Spectrometry Conference*. Prague 2006.

111. Lustgarten, J.L., et al., *EPO-KB: A searchable knowledge base of biomarker to protein links.* Bioinformatics, 2008. **24**(11): p. 1418-1419.
112. Buchanan, B.G. and E.A. Feigenbaum, *Dendral and Meta-Dendral: Their applications dimension.* Artificial Intelligence, 1978. **11**(1-2): p. 5-24.
113. Feigenbaum, E.A. and B.G. Buchanan, *Dendral and Meta-Dendral - Roots of knowledge systems and expert system applications.* Artifical Intelligence, 1993. **59**(1-2): p. 233-240.
114. Feigenbaum, E.A., B.G. Buchanan, and J. Lederberg. *On generality and problem solving: A case study using the dendral program*. in *Machine Intelligence 6*. 1971. New York: American Elsevier.
115. Shortliffe, E.H., et al., *Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the MYCIN system.* Comput Biomed Res, 1975. **8**(4): p. 303-20.
116. Duda, R.O., J. Gaschnig, and P.E. Hart, *Model design in the PROSPECTOR consultant system for mineral exploration*, in *Expert Systems in the Microelectronic Age*, Michie, Editor. 1979, Edinburgh University Press: Edinburgh, Scotland. p. 154-167.
117. Durkin, J., *Application of expert systems in the sciences.* Ohio Journal of Science, 1990. **90**(5): p. 171-179.
118. Horvitz, E.J., J.S. Breese, and M. Henrion, *Decision theory in expert systems and artificial intelligence.* International Journal of Approximate Reasoning, 1988(2): p. 247-302.
119. Mascherini, M. and F.M. Stefanini, *Using weak prior information on structures to learn Bayesian networks* in *Knowledge-Based Intelligent Information and Engineering Systems*, B. Apolloni, R.J. Howlett, and L.C. Jain, Editors. 2008, Springer Berlin / Heidelberg. p. 413-420.
120. Stuart, J.R. and N. Peter, *Artificial Intelligence: A Modern Approach*. 2nd Edition ed. 2003: Pearson Education. 1132.
121. Dietterich, T. and E.B. Kong, *Machine learning bias, statistical bias, and statistical variance of decision tree algorithms*. 1995, Oregon State University.
122. Sindhwani, V., P. Bhattacharya, and S. Rakshit. *Information theoretic feature crediting in multiclass support vector machines*. in *Proceedings of the First SIAM International Conference on Data Mining*. 2001. Chicago, IL.
123. Djebbari, A. and J. Quackenbush, *Seeded Bayesian networks: Constructing genetic networks from microarray data.* BMC Syst Biol, 2008. **2**: p. 57.
124. Visweswaran, S., *Learning Patient-Specific Models from Clinical Data*, in *Deparment of Intelligent Systems*. 2007, University of Pittsburgh School of Arts and Sciences: Pittsburgh. p. 198.
125. Kontkanen, P., et al. *On supervised selection of Bayesian networks*. in *Fifteenth International Conference on Uncertainty in Artificial Intelligence*. 1999: Morgan Kaufmann.
126. Heckerman, D., *A tutorial on learning Bayesian networks*. 1996, Microsoft Research.
127. Witten, I.H. and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd ed. 2005, San Francisco: Morgan Kaufmann.
128. Li, J. and L. Wong, *Using Rules to Analyse Bio-medical Data: A Comparison between C4.5 and PCL*, in *Advances in Web-Age Information Management*. 2003. p. 254-265.
129. Robnik-Sikonja, M. and I. Kononenko, *Pruining regression trees with MDL.* ECAI 98: 13th European Conference on Artificial Intelligence, 1998: p. 455-459.

130. Buchanan, B.G. and G.R. Livingston, *Toward automated discovery in the biological sciences*, in *AI Magazine*. 2004. p. 69-84.
131. Livingston, G.R. and B.G. Buchanan. *Autonomous discovery in empirical domains*. in *AAAI*. 1999.
132. Cooper, G.F., *An overview of the representation and discovery of causal relationships using Bayesian networks*. Computation, causation, and discovery, ed. C.G.a.G.F. Cooper. 1999, Menlo Park, CA.: AAAI Press and MIT Press. 3-62.
133. Geiger, D., T. Verma, and J. Pearl, *Identifying independence in bayesian networks.* Networks, 1990. **20**(5): p. 507-534.
134. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 1988: Morgan Kaufmann Publishers Inc. 552.
135. Geiger, D., A. Paz, and J. Pearl, *Axioms and algorithms for inferences involving probabilistic independence.* Inf. Comput., 1991. **91**(1): p. 128-141.
136. Visweswaran, S. and G.F. Cooper. *Patient-specific models for predicting the outcomes of patients with community acquired pneumonia*. In: *Proceedings of the American Medical Informatics Association Annual Symposium*. Washington, DC 2005. 759-763.
137. Castillo, E., J. Gutiérrez, and A. Hadi, *Parametric structure of probabilities in Bayesian networks*, in *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*. 1995. p. 89-98.
138. Xu-wen, C., A. Gopalakrishna, and W. Xinkun, *An effective structure learning method for constructing gene networks.* Bioinformatics, 2006. **22**(11): p. 1367-1374.
139. Cooper, G.F., *The computational complexity of probabilistic inference using Bayesian belief networks.* Artificial Intelligence, 1990. **42**(2-3): p. 393-405.
140. Jensen, F.V. and F. Jensen. *Optimal junction trees*. In: *Proceedings of the Uncertainty and Artificial Intelligence: Proceedings of the Tenth Conference* San Mateo, CA July 1994, 1994. Morgan Kaufmann. 360-366.
141. Li, Z. and B. D'Ambrosio, *Efficient inference in Bayes networks as a combinatorial optimization problem.* International Journal of Approximate Reasoning, 1994. **11**: p. 55-81.
142. Ling, C.X., J. Huang, and H. Zhang. *AUC: A statistically consistent and more discriminating measure than accuracy*. in *International Joint Conference on Artificial Intelligence, Proceedings of the Eighteenth (IJCAI '03)*. 2003.
143. Ressom, H.W., et al. *Biomarker identification and rule extraction from mass spectral serum profiles*. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, 2006 (CIBCB '06)*. 2006. 1-7.
144. Towell, G.G., J.W. Shavlik, and N.O. Noordenier. *Refinement of approximate domain theories by knowledge based neural networks*. in *Proceedings of the 8th Nation Conference on AI*. 1990. Boston, MA.
145. Friedman, N. and M. Goldszmidt. *Learning Bayesian networks with local structure*. in *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*. 1998. Erice, Italy: Kluwer Academic Publishers.
146. Pettey, C.B., M.R. Leuze, and J.J. Grefenstette. *A parallel genetic algorithm*. 1987: Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA.
147. Alon, U., et al., *Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays.* Proceedings of the

National Academy of Sciences of the United States of America., 1999. **96**(12): p. 6745-50.
148. Nutt, C.L., et al., *Gene expression-based classification of malignant gliomas correlates better with survival than histological classification.* Cancer Res, 2003. **63**(7): p. 1602-7.
149. Pomeroy, S.L., et al., *Prediction of central nervous system embryonal tumour outcome based on gene expression.* Nature, 2002. **415**(6870): p. 436-42.
150. Staunton, J.E., et al., *Chemosensitivity prediction by transcriptional profiling.* Proc Natl Acad Sci U S A, 2001. **98**(19): p. 10787-92.
151. Pusztai, L., et al., *Pharmacoproteomic analysis of pre-and post-chemotherapy plasma samples from patients receiving neoadjuvant or adjuvant chemotherapy for breast cancer.* Journal of Clinical Oncology, 2004. **22**(14S).
152. Ranganathan, S., *Proteomic profiling of cerebrospinal fluid identifies diagnostic biomarkers for Amyotrophic Lateral Sclerosis*, in *Pathalogy*. 2003, University of Pittsburgh: Pittsburgh, PA.
153. Caruana, R. and A. Niculescu-Mizil. *Data mining in metric space: an empirical analysis of supervised learning performance criteria*. 2004: ACM New York, NY, USA.
154. Forbes, A.D., *Classification-algorithm evaluation: Five performance measures based on confusion matrices.* Journal of Clinical Monitoring and Computing, 1995. **11**(3): p. 189-206.
155. Domingos, P., *The role of Occam's Razor in knowledge discovery.* Data Mining and Knowledge Discovery, 1999. **3**(4): p. 409-425.
156. Pedro, D. and P. Michael, *On the optimality of the simple Bayesian classifier under zero-one loss.* Machine Learning, 1997. **29**(2-3): p. 103-130.
157. Yang, Y. and G.I. Webb, *Discretization for data mining.* Encyclopedia of data warehousing and mining. Idea Group Reference, 2005: p. 1-16.
158. Pak, J.H., et al., *Involvement of neurogranin in the modulation of calcium/calmodulin-dependent protein kinase II, synaptic plasticity, and spatial learning: A study with knockout mice.* Proc Natl Acad Sci U S A, 2000. **97**(21): p. 11232-11237.
159. Sinclair, G.B., et al., *Generation of a conditional knockout of murine glucocerebrosidase: Utility for the study of Gaucher disease.* Molecular Genetics and Metabolism, 2007. **90**(2): p. 148-156.
160. Listgarten, J. and A. Emili, *Statistical and computational methods for comparative proteomic profiling using liquid chromatography-tandem mass spectrometry.* Molecular & Cellular Proteomics, 2005: p. 419-434.
161. States, D.J., et al., *Challenges in deriving high-confidence protein identifications from data gathered by a HUPO plasma proteome collaborative study.* Nature Biotechnology, 2006. **24**(3): p. 333-338.
162. Ehlert, P., Q.M. Mouthaan, and L. Rothkrantz. *A rule-based and a probabilistic system for situation recognition in a flight simulator*. In: *Proceedings of the Intelligent Games and Simulation* 2003. Eurosis.
163. Onisko, A., P. Lucas, and M.J. Druzdzel, *Comparison of rule-based and Bayesian network approaches in medical diagnostic systems.* Lecture Notes in Computer Science, 2001. **2101**: p. 283-292.
164. Valko, M., R. Pelikan, and M. Hauskrecht. *Learning predictive models for multiple heterogeneous proteomic data sources*. In: *Proceedings of the Proceedings of the Summit on Translational Bioinformatics*. San Francisco, CA 2008.

165. Kanehisa, M., et al., *From genomics to chemical genomics: new developments in KEGG.* Nucleic Acids Research, 2006. **34**(D354-357).
166. Lustgarten, J.L., et al. *Using a novel resource to decrease proteomic biomarker identification time*. In: *Proceedings of the AMIA Annual Symposium Proceedings*. Washington, D.C. November 6, 2008. 1033.
167. Shane, S., *Prior knowledge and the discovery of entrepreneurial opportunities* Organization Science, 2000. **11**(4): p. 448-469.
168. Taria, L., C. Barala, and S. Kima, *Fuzzy c-means clustering with prior biological knowledge.* Journal of Biomedical Informatics, 2008.
169. Wei, P. and W. Pan, *Incorporating gene networks into statistical tests for genomic data via a spatially correlated mixture model.* Bioinformatics, 2008. **24**(3): p. 404-411.