Information learning approach

Zheng, Yun

2007

Zheng, Y. (2007). Information learning approach. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/2480

https://doi.org/10.32657/10356/2480

NANYANG TECHNOLOGICAL UNIVERSITY

INFORMATION LEARNING APPROACH

ZHENG YUN

2007

INFORMATION LEARNING APPROACH

ZHENG YUN

SCHOOL OF COMPUTER ENGINEERING

2007

INFORMATION LEARNING APPROACH

ZHENG YUN

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2007

Acknowledgments

FIRST AND FOREMOST, I would like to thank my supervisor, Kwoh Chee Keong, for his guidance and support throughout my candidature at Nanyang Technological University. Dr Kwoh was very helpful and influential to my research and development, with his solid knowledge in both statistics and mathematics.

I would like to thank Wong Limsoon from Institute of Infocomm Research in Singapore, now with National University of Singapore. Limsoon had helped me magnanimously despite of his busy schedule and never hesitated to give me his valuable guidance. With his comprehensive knowledge in both computer science and computational biology, Limsoon gave me valuable inter-discipline advice which is not available from others.

I would like to express my appreciation to George C. Tseng from University of Pittsburgh in the United States, who is prominent in Biostatistics. George shared with me many of his expertise for the formalization of the method proposed in this thesis.

I am also grateful to Li Jinyan from Institute of Infocomm Research in Singapore. Jinyan is an expert in classification and has given me many useful suggestions when I was preparing some papers in classification problems.

I owe a lot to Ng See-Kiong from Institute of Infocomm Research in Singapore for the applications used in this thesis. It was See-Kiong who helped me to find more applications for our method in the early stage of my research. Without this suggestion, the applications of classification and feature selection will never be included in this thesis.

I also thank Prasanna R Kolatkar from Genome Institute of Singapore. Prasanna is a structure biologist. In the early years of my candidature, Prasanna kindly shared his knowledge in molecular biology and genetics with me.

I also would like to thank Liu Huiqing from University of Georgia in the United States. Huiqing is a specialist in cancer classification based on biological data sets. When I was anxiously looking for data sets, Huiqing generously shared a biomedical data repository maintained by her. I also benefited a lot from Huiqing's expertise in feature selection.

I am grateful to the three reviewers of my thesis. They gave me many valuable suggestions and comments for the improvement of this thesis.

I also thank my friends, Zhao Ying, Chen Jinmiao, Du Zhihua, Zhou Juan, Yang Xiao and Zhu Zexuan at Bioinformatics Research Center of Nanyang Technological University, Zhang Guang Lan from Institute of Infocomm Research in Singapore, Li Ye from Bioinformatics Institute in Singapore. In my candidate, their friendships, collaboration, and encouragement were my source of inspiration for higher achievements. The time shared with them gave me many joyful memory and many happy experiences.

I am grateful to all other friends for their encouragements and friendship. The experiences as a Ph.D candidate let me understand the valuable friendship with them.

Finally and persistently, I appreciate my parents and my brother for everything they taught me. It is my parents who gave me all necessary foundations for my life and development. It is also my parents who taught me their priceless value principles: integrity and diligence. They also gave me constant support at any time, in any place. I cannot imagine that I can finish this thesis without them.

Abstract

COMPUTATIAL LEARNING THEORY is a mathematical field related to the analysis of machine learning algorithms. Based on different principles of inference and different definitions of probability, there have been several approaches to computational learning theory, like the Probably Approximately Correct (PAC) learning theory proposed by Leslie Valiant; the VC theory proposed by Vladimir Vapnik; the Bayesian inference arising from the work first done by Thomas Bayes; and the algorithmic learning theory from the work of E. Mark Gold.

Different approaches of computational learning theory have led to different learning algorithms. For example, boosting has been proposed with the PAC theory, support vector machines have been invented with the VC theory, and Bayesian networks have been proposed with the Bayesian inference.

In this thesis, we propose a new approach to computational learning theory, called Information Learning Approach (ILA). The ILA is based on information theory, incorporating ideas from graphical models. In our approach, learning is interpreted and regarded as a procedure to acquire information of the concept, like the class attribute in classification problems. Particularly, the ILA is based on a theorem, which says that if the mutual information between a vector $\mathbf{X}$ and a variable Y equals to the entropy of the variable Y, then Y is a function of $\mathbf{X}$. We propose a learning algorithm, called Discrete Function Learning (DFL) algorithm, to fulfill the learning task as a process of obtaining information of the concept. We prove that the learning process of the DFL algorithm can be done in polynomial time, if there are limited number of attributes which essentially describe the concept.

We demonstrate three distinct application areas of the *information learning approach*:

1. To find qualitative models of Gene Regulatory Networks (GRNs) from time-series microarray gene expression data sets. GRNs are the underlying mechanism which controls different expression patterns of the genes within a cell and the developmental processes of a species. We prove that the DFL algorithm can learn qualitative models of GRNs more efficiently than existing methods without loss of accuracy.

2. To solve classification problems with tissue-specific microarray gene expression data sets for cancer classification and other benchmark data sets. The DFL algorithm obtains comparable or more competitive prediction accuracies than existing classification methods with lower-complexity models in our experiments.

3. To find informative and discriminatory subset of features which can be used by other classification algorithms. The experimental results show that the DFL algorithm can find more informative and discriminatory feature subsets than current methods in shorter time.

Contents

Acknowledgments i

Abstract iii

List of Figures xii

List of Tables xv

1 Introduction **1**

1.1 Related Work 2

1.1.1 The Probably Approximately Correct Learning Theory 2

1.1.2 The Vapnik Chervonenkis Theory 4

1.1.3 The Bayesian Inference 5

1.1.4 The Algorithmic Learning Theory 7

1.1.5 Limitations of Current Approaches 8

1.2 Information Learning Approach 8

1.2.1 Our Approach 9

1.2.2 Other Learning Methods Based On Information Theory 12

1.2.3 Comparisons of Different Learning Approaches 13

1.3 Contributions 13

1.4 Outline 15

2 Information Learning Approach 18

2.1 Background Knowledge . . . 18

2.1.1 Fundamental Knowledge of Information Theory . . . 19

2.1.2 Background knowledge of Graphical Models . . . 23

2.2 Information Learning Approach . . . 24

2.2.1 Problem Definition . . . 24

2.2.2 The Information Learning Approach . . . 26

2.3 Relation to Markov Blanket . . . 28

2.3.1 Markov Blanket . . . 29

2.3.2 The Relation of Information Learning Approach and Markov Blanket 30

2.4 The Discrete Function Learning Algorithm . . . 30

2.4.1 Theoretical Motivation . . . 30

2.4.2 The Discrete Function Learning Algorithm . . . 35

2.4.3 Complexity Analysis . . . 41

2.4.4 Correctness Analysis . . . 42

2.5 The ϵ Value Method for Noisy Data Sets . . . 43

2.5.1 The ϵ Value Method . . . 44

2.5.2 The Relation with The Over-fitting Problem . . . 45

2.5.3 The Relation with The Time Complexity . . . 47

2.6 Selection of Parameters . . . 47

2.6.1 Selection of The Expected Cardinality K . . . 47

2.6.2 Selection of ϵ value . . . 48

2.6.3 Balance between K and ϵ . . . 51

2.7 Prediction Methods . . . 51

2.8 Implementation Issues . . . 53

2.8.1 The Computation of Mutual Information $I(\mathbf{U}; Y)$ . . . 53

2.8.2 Redundancy Matrix 54
2.9 Conclusions 55

3 Learning Qualitative Models of Gene Regulatory Networks 57
3.1 A Brief Introduction to Gene Regulatory Networks 58
3.1.1 Introduction to Microarray Technology 58
3.1.2 Analyzing Gene Expression Profiles with The DFL Algorithm . . . 60
3.1.3 What Are Gene Regulatory Networks 62
3.1.4 Why We Research on Gene Regulatory Networks 66
3.1.5 How to Obtain Gene Regulatory Networks 68
3.1.6 What Are the Challenges for Reconstructing Gene Regulatory Networks? 68
3.2 Methods 71
3.2.1 Qualitative Models of Gene Regulatory Networks 71
3.2.2 The DFL Algorithm for Learning Gene Regulatory Network Models 73
3.2.3 Data Quantity for Learning Qualitative Gene Regulatory Network Models 75
3.3 Learning Models of Generalized Logical Formalism 76
3.4 Learning Gene Regulatory Networks Related to Yeast Cell Cycle 77
3.4.1 Learning Gene Regulatory Networks with The DFL Algorithm . . . 78
3.4.2 Applying The ϵ Value Method to Yeast Cell Cycle Data 82
3.5 Conclusions 83

4 Learning Boolean Networks 85
4.1 Current Methods For Learning Boolean Networks 86
4.2 Learning Boolean Networks With The DFL Algorithm 88
4.2.1 Problem Definition 88

4.2.2 Obtaining Correct Truth Table From Noisy Data Sets 89
4.3 Evaluation Criterion for Learning Boolean Networks 90
4.4 The Analysis Of Some Special Boolean Networks 92
4.4.1 The Mutual Information in OR Boolean Networks 92
4.4.2 The Complexity Analysis for Bounded OR Boolean Networks . . . 95
4.4.3 The Complexity Analysis for Unbounded OR Boolean Networks . . 98
4.5 The Analysis of General Boolean Functions 98
4.5.1 The Cases Which Require More Computation Time 99
4.5.2 The Constant Functions 100
4.6 Results 100
4.6.1 Synthetic Data Sets of Boolean Networks 100
4.6.2 Experiments for Time Complexity 101
4.6.3 Experiments for Sensitivity 106
4.6.4 Comparisons of Time Complexity with Existing Methods 111
4.6.5 Comparisons of Robustness to Noisy Data Sets with Existing Methods 113
4.7 Related Models 114
4.7.1 Probabilistic Boolean Networks 114
4.7.2 Dynamic Bayesian Networks 115
4.8 Conclusions 116

5 Solving Classification Problems 119
5.1 Cancer Classification with Biological Data 120
5.1.1 A Brief Introduction to SELDI Technology 120
5.1.2 Cancer Classification with Biological Data 121
5.2 A Supervised Discretization Method 122

5.3 Results 124
5.3.1 Data Sets 124
5.3.2 Comparison with Other Classification Methods 125
5.3.3 Comparison of Model Complexity 129
5.3.4 Comparison of Efficiency 130
5.4 Biological Evaluation 131
5.5 Discussions and Conclusions 133

6 Performing Feature Selection 136
6.1 Introduction 137
6.2 Related Work 139
6.2.1 Categorization of Feature Selection Methods 139
6.2.2 Feature Selection Methods Based on Information Theory 140
6.2.3 Limitations of Current Feature Subset Selection Methods 143
6.3 Choosing Essential Attributes with The Information Learning Approach . . 144
6.4 Results 145
6.4.1 The DFL Algorithm as A Filter Feature Selection Method 145
6.4.2 Comparison with Other Feature Selection Methods 147
6.4.3 Comparison of Model Complexity 151
6.4.4 Comparison of Efficiency 153
6.5 Discussions 154
6.6 Conclusions 159

7 Conclusions 161
7.1 Discussions 161
7.1.1 Comparison of The ILA and PAC Theory 161
7.1.2 Comparison of The ILA and VC Theory 162

7.1.3 Comparison of ILA and Bayesian Inference 164
7.1.4 Comparison of ILA and Algorithmic Learning Theory 165
7.2 Contributions 166
7.3 Limitations and Future Work 168
7.4 Perspective 169

Appendix A

The Proofs of the Theorems in Chapter 2 171
A.1 Jensen's Inequality 171
A.2 Proofs of the Theorems in Section 2.1 172
A.3 Proofs of the Theorems in Section 2.2 173
A.4 Proofs of the Theorems in Section 2.4 177

Appendix B

The Proofs of the Theorems in Chapter 3 179

Appendix C

The Proofs of the Theorems in Chapter 4 180

Appendix D

The Detailed Settings 188

Appendix E

The Extended Main Steps of The DFL Algorithm 191

Appendix F

Notation 194

Appendix G

Abbreviations 196

Bibliography **198**

List of Figures

2.1 The relationship of entropy and mutual information. 20
2.2 The Asia Bayesian network. 25
2.3 The mutual information between X_i and the class attribute Y for the LED+17 data sets. 32
2.4 The advantage of using mutual information to choose the most discriminatory feature vectors. 33
2.5 The rules of the *noiseless* LED+17 data set learned by the DFL algorithm. . 34
2.6 Search procedures of the DFL algorithm when learning $Y = (A \cdot C) + (A \cdot D)$. 38
2.7 The $\Delta Tree$ when searching the EAs for $Y = (A \cdot C) + (A \cdot D)$. 39
2.8 The exhaustive searching procedures of the DFL algorithm when learning $Y = (A \cdot C) + (A \cdot D)$. 40
2.9 The Venn diagram of $H(\mathbf{X})$,$H(Y)$ and $I(\mathbf{X}, Y)$, when $Y = f(\mathbf{X})$. 44
2.10 The performance of the DFL algorithm for different ϵ values. 46
2.11 The manual binary search of minimum ϵ value. 49
2.12 The noisy rules in the one-dimensional space. 52
3.1 Overview of cDNA microarray technology. 59
3.2 A schematic view of using the Discrete Function Learning algorithm to analyze microarray gene expression profiles. 61
3.3 Three states of the *lac* genes. 64

3.4 Gross anatomy of a minimal gene regulatory network (GRN) embedded in a regulatory network. 66

3.5 A simple GLF model of GRN. 76

3.6 The learned GRN model related to yeast cell cycle. 78

3.7 The learned GRN model for yeast cell cycle with the ϵ function method. . . 82

3.8 The combined GRN models related to yeast cell cycle. 83

4.1 The $I(X_j; X_i')$ for OR BLNs with 10 variables, where $X_i' = X_1 + X_2 + X_3$. 91

4.2 Mutual information in OR function $X_i' = X_{i1} + \ldots + X_{ik}$. 94

4.3 The run time, t (vertical axes, shown in seconds), of the DFL algorithm for inferring the bounded BLNs. 102

4.4 The efficiency of the DFL algorithm for the unbounded OR data sets. 103

4.5 The histograms of the number of subsets checked for learning one Boolean function, m, and run time of the DFL algorithm for RANDOM data sets, when $n = 100$, $k = 3$ and $N = 200$. 105

4.6 The sensitivity of the DFL algorithm vs sample size N. 107

4.7 The run time, t (vertical axis, shown in seconds), of the DFL algorithm for small OR data sets, where $n = 100$ and $k = 3$. 108

4.8 The setting and performance of the DFL algorithm for noisy data sets, whose $k = 3$, $n = 100$ and $N = 1000$. 109

4.9 Comparison of the run times of the DFL algorithm and the REVEAL algorithm. 112

5.1 Overview of Surface-Enhanced Laser Desorption/Ionization (SELDI) type mass spectrometry technology. 120

5.2 The comparison of training time of different classification algorithms. . . . 131

5.3 The comparisons of the expression values of the genes chosen by the DFL algorithm. 132

5.4 The philosophy of the DFL algorithm and other classification algorithms. . 134
6.1 The comparison of training times of different classification algorithms on all features and on the features chosen by the DFL algorithm. 148
6.2 The comparison of accuracies for different feature subset selection methods. 150
6.3 The number of features chosen by different feature selection methods. . . . 152
6.4 The run times of different feature selection methods. 154
6.5 The $I(X_i; Y)$ in the data sets generated with $Y = \neg(X_{21} \bigoplus X_{29} \bigoplus X_{60})$. . 157
7.1 A schematic view of the unstableness of the classifiers obtained by the support vector machine algorithm. 163

List of Tables

1.1 The comparison of different approaches to computational learning theory. . 13

2.1 The DFL algorithm. 35

2.2 The subroutine of the DFL algorithm. 36

2.3 The training data set $\mathbf{T}$ of the example to learn $Y = (A \cdot C) + (A \cdot D)$. . . 37

2.4 The learned classifier f of the example to learn $Y = (A \cdot C) + (A \cdot D)$. . . 39

3.1 The DFL algorithm for learning qualitative models of GRNs. 74

3.2 The correlation coefficient matrix of the GLF example in Figure 3.5. 77

3.3 The literature evidences for the GRN model in Figure 3.6 and Figure 3.7. . 79

3.4 The prediction measures. 80

3.5 The accuracy, sensitivity and precision of the DFL algorithm and the K2 algorithm. 81

4.1 The summary of complexities of different algorithms for learning BLNs. . . 87

4.2 The truth table of $B^{'} = A + C$. 93

4.3 The obtained Boolean rules from one noisy RANDOM-h data set with 1000 samples and 10% noise. In the original BLN, $X_i^{'} = \neg X_1 \cdot X_2 + X_1 \cdot \neg X_2 \cdot X_3$.111

4.4 The success ratio (%) of the GREEDY1 algorithm [9] for noisy data sets of general BLNs, where the indegree $k = 3$ and $n = 1000$. 113

5.1 The benchmark data sets used in the experiments for comparison. 125

5.2 The comparison of accuracies for the DFL algorithm and other well-known classification methods. 126

5.3 The summary of prediction performances of different algorithms in Table 5.2. 127

5.4 The comparison of the DFL algorithm and other methods in literature. . . . 128

5.5 The classifier for the ALL data set learned with the DFL algorithm. 132

6.1 The accuracies of the well-known classification methods on the data sets filtered with the features chosen by the DFL algorithm. 147

6.2 The improvement of performances of different learning algorithms when applied to the features chosen by the DFL algorithm. 148

6.3 The comparison summary of accuracies obtained by different feature selection methods. 151

6.4 The comparison summary of the number of features chosen by different feature selection methods. 153

C.1 The truth table of $X_i^{'} = X_1 + X_2 + X_3$. 182

C.2 The tuple $(X_{ij}, X_i^{'})$ and $(X_{ij}^{*}, X_i^{'})$, where k is two. 186

D.1 The settings of the DFL algorithm. 189

D.2 The features chosen by the DFL algorithm. 190

E.1 The extended version of the DFL algorithm. 192

E.2 The extended version of the subroutine of the DFL algorithm. 193

Chapter 1

Introduction

WITH more and more data has been accumulated in databases, it becomes a more and more urgent challenge to extract useful knowledge from these large amount of data.

Machine learning is such a field to find meaningful and useful knowledge from databases by developing computer programs which can obtain knowledge from data sets. Machine learning algorithms take a training data set, formulate hypotheses or models, and make predictions about the future with the hypotheses or models. In statistics, the analysis of machine learning algorithms is categorized as computational learning theory. There have been several approaches to computational learning theory: the Probably Approximately Correct (PAC) learning theory proposed by Leslie Valiant [166]; the VC theory proposed by Vladimir Vapnik [168]; the Bayesian inference [131] arising from the work first done by Thomas Bayes and the algorithmic learning theory from the work of E. M. Gold [80].

In this thesis, we propose a new approach, called Information Learning Approach (ILA), to computational learning theory to find useful and meaningful knowledge from databases. The ILA is based on information theory, with some ideas from graphical models. In our approach, learning is explained and regarded as a procedure to acquire information of the concept, such as the class attribute in classification problems.

We also propose the Discrete Function Learning (DFL) algorithm to efficiently implement the ILA. The DFL algorithm learns a concept by finding a small subset of informative features from all variables that describe the concept. Then, it constructs models with these critical features, and uses these models to predict the future events.

The ILA has several advantages. First, there is solid theoretical motivation and foundation for the ILA. Second, the DFL algorithm is efficient and easily understandable. Third, the ILA is robust to noise in the data sets.

1.1 Related Work

In this section, we will briefly review the existing approaches to computational learning theory and analyze their limitations.

1.1.1 The Probably Approximately Correct Learning Theory

The Probably Approximately Correct (PAC) learning theory is first proposed by Leslie Valiant [166]. Here, we will briefly describe the PAC learning by following the definition in [91]. The intention of the PAC approach is that successful learning should obtain a hypothesis, with high probability, which approximate the concept well [91]. In the basic model of the PAC learning, the samples are assumed to be obtained from n-dimensional Boolean space $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$. The instance space of $\mathbf{V}$ is $\mathbb{V} = \{\{0,1\}^n : n \in \mathbf{N}\}$. The concept C and the hypothesis H are described by subsets of $\mathbf{V}$. The notion of approximation is defined by a probability distribution D defined on $\mathbf{V}$, $\forall \mathbf{v} \in \mathbb{V}$, $p(\mathbf{v}) > 0$. The error of a hypothesis h with respect to a fix target concept c, denoted with $e(h)$, is defined by

$$e(h) = \sum_{x \in h \Psi c} D(x),$$

where Ψ is the symmetric difference. Thus, $e(h)$ is the probability that h and c will disagree on an instance drawn randomly according to D. The hypothesis h is a good approximation of the target concept c if $e(h)$ is small.

For each $n \geq 1$, let C_n be a set of target concepts over the instance space $\mathbb{V}$, and let $\mathbf{C} = \{C_n : n \in \mathbf{N}\}$. Let H_n for $n \geq 1$, and $\mathbf{H} = \{H_n : n \in \mathbf{N}\}$. The PAC theory is defined as follows.

Definition 1.1.1 (PAC Learnability) *The concept class* $\mathbf{C}$ *is PAC learnable by the hypothesis space* $\mathbf{H}$ *if there exists a polynomial time learning algorithm* A *and a polynomial* $p(\cdot,\cdot,\cdot)$ *such that* $\forall n \geq 1$*, all target concepts* $c \in C_n$*, all probability distributions* D *on the instance space* $\mathbb{V}$*, and all* ϵ *and* δ*, where* $0 < \epsilon, \delta < 1$*, if the algorithm* A *is given at least* $p(n, 1/\epsilon, 1/\delta)$ *independent random samples of* c *drawn according to* D*, then with probability at least* $1 - \delta$*,* A *will return a hypothesis* $h \in H_n$ *with error* $e(h) \leq \epsilon$*. The smallest such polynomial* p *is the sample complexity of the learning algorithm* A*.*

The sample complexity of the PAC theory is at most $\frac{1}{\epsilon}(ln|H_n| + ln\frac{1}{\delta})$ [168]. One important characteristic of this definition is that the learning algorithm A must process the samples in polynomial time, and must produce a good approximation to the target concept with high probability using only a reasonable number of training samples randomly drawn according to D [91].

Valiant [166] proved that k-CNF is PAC learnable, where k-CNF denotes the conjunctive normal form where each clause is the sum of at most k literals. Later, k-DNF (disjunctive normal form) and k-decision lists were also proved to be PAC learnable [142, 165].

The PAC theory inspired the boosting algorithm. Boosting occurs in stages, by incrementally adding to the current learned function. At every stage, a weak learner (i.e., one that can have an accuracy as bad as slightly greater than chance) is trained with the data. The output of the weak learner is then added to the learned function, with some strength (proportional to how accurate the weak learner is). Then, the data is reweighted: examples that the current learned function gets wrong are "boosted" in importance, so that future weak learners will attempt to fix the errors.

There are two major criticisms levelled at the PAC learning. First, the worse case emphasis in the model makes it unusable in practice [36]. Second, the notions of target concepts and noise-free training data are too restrictive in practice [26].

1.1.2 The Vapnik Chervonenkis Theory

Vapnik Chervonenkis theory (also known as VC theory) was developed during 1960-1990 by Vladimir Vapnik and Alexey Chervonenkis, and formalized in [168]. The VC theory explicitly takes into account the sample size and provides quantitative description of the trade-off between the model complexity and the available information (i.e., finite training data) [41]. This theory consists of four parts [168]:

1. The general qualitative theory that includes the necessary and sufficient conditions for consistency of learning process;

2. The general quantitative theory that includes bounds on the rate of convergence (the rate of generalization) of these learning process;

3. Principles for estimating functions from a small collection of data that are based on the developed theory;

4. Methods of function estimation and their application to solving real-life problems that are based on these principles.

The last part of the VC theory introduced a well-known algorithm, the Support Vector Machines (SVMs). The SVMs choose some samples as support vectors, and approximate the classification function with these support vectors, such that the distance from the closest examples (the margin) to the classification boundary is maximized. The classification boundary can be approximated with various type of functions, i.e., kernels. Different kernels produce different boundaries, e.g., the basic linear kernels will produce boundaries as hyperplanes.

The use of the maximum-margin boundary is motivated by Vapnik Chervonenkis theory, which provides a probabilistic test error bound which is minimized when the margin is maximized.

There are some limitations in the Support Vector Machines. Perhaps, the biggest limitation of the support vector approach lies in the choice of the kernel function. Once the kernel is fixed, SVM classifiers have only one user-chosen parameter (the error penalty), but the kernel is a very big rug under which to sweep parameters. Some work has been done on limiting kernels using prior knowledge [37, 149], but the best choice of kernel for a give problem is still a research issue [51, 117]. The second limitation is speed and size, both in training and testing. Training for very large data sets (millions of support vectors) is an unsolved problem. The third problem is discrete data, although with suitable rescaling excellent results have nevertheless been obtained [99]. Finally although some work has been done on training multi-class SVMs in one step, the optimal design for multi-class SVMs classifiers is a further area for research.

1.1.3 The Bayesian Inference

The Bayesian inference is a kind of statistical inference in which the probabilities are not interpreted as frequencies but rather as degrees of belief. The name is coming from the

Bayes theorem, proposed by Thomas Bayes, in the following.

Theorem 1.1.1 (Bayes Theorem)

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}, \tag{1.1}$$

where H stands for the hypothesis, E stands for the evidence, or the training samples, $P(H|E)$ is called the posterior probability of H given E, $P(E|H)$ is called the likelihood of E given H, $P(H)$ is called the prior probability of H, $P(E)$ is called the marginal probability of E.

The Bayesian inference of a concept C is to estimate the probability of $P(C = c|\mathbf{V} = \mathbf{v})$. However, the conditional probability $p(c|\mathbf{v})$ is very hard to estimate, since it is in the multi-dimensional space defined by $\mathbf{V}$.

Pearl [131] proposed the Bayesian networks, which will be introduced in Section 2.1.2. There have been some inference algorithms for Bayesian networks [101, 132]. The inference of Bayesian networks has been proved to be NP-hard [47, 52]. Before the inference can be performed, Bayesian networks are first needed to be learned from data, which is also NP-hard [42, 43].

There are some criticisms levelled at the Bayesian inference. One major criticism is the prior probability $P(H)$. In some situations, the $P(H)$ is very hard to estimate with prior knowledge, since the prior knowledge may be very limited in these cases. Another criticism is that different people may give different prior probabilities, since different people hold different degrees of belief for the hypothesis.

1.1.4 The Algorithmic Learning Theory

The algorithmic learning theory, or *inductive inference* [17], is introduced by E. M. Gold [80], in which Gold analyzed the learnability of language. The objective is to develop a program which can produce another program by which any given sentences can be determined to be "grammatical" or "ungrammatical".

In the framework of algorithmic learning theory, the tester gives the learner an example sentence at each step, and the learner responds with a hypothesis, which is the program capable of telling the grammatical correctness. The algorithmic learning theory requires the ergodicity of the sentences, i.e., that every possible sentence will be called by the tester, without specific requirement of the order of these sentences. It is also required that at each step the hypothesis is consistent with the sentences provided so far.

A particular learner is said to be able to "learn a language in the limit" if there is a certain number of steps beyond which the hypothesis given by the learner does not improve any more. At this point, the language has been learned by the learner, since all possible sentences have been called by the tester, and the hypothesis is consistent with all sentences.

Gold showed that any language defined by a Turing-machine can be learned in the limit by another Turing-complete machine using enumeration. The learner reaches this goal by testing all possible Turing machines in turn until it has found one consistent with all the provided sentences so far, which is the hypothesis for the current step. Eventually, the learner can produce the correct hypothesis, which will not change again.

The major shortcoming of the algorithmic learning theory lies in the unrestricted time and space complexity, which may be limited in practice. Another shortcoming is that the input data has to be noiseless, since the enumeration method may fail when the input sentences are incorrect or noisy.

The Gold paradigm has significantly been developed to allow change of minds [118, 152] and noise [16, 39, 147], as also demonstrated by many works from Sanjay Jain [97], Stephan Frank and many other researchers.

Some people also put the PAC theory to the same category of algorithmic learning theory.

1.1.5 Limitations of Current Approaches

In this section, we summarize the limitations of existing approaches to computational learning theory. The PAC learning theory only makes approximations to the original models, as indicated by its name. The VC theory tries to maximize the margins between the classes, but essentially it is still trying to approximate the original models. Bayesian inference is more robust to noise, and can find a better explanation for the concept. However, Bayesian inference is much computationally expensive, i.e., NP-hard, and still does not explicitly provide the original models but implicitly provides the probability of a hypothesis. The algorithmic learning theory searches the hypotheses by enumeration, which is computationally intractable. The algorithmic learning theory does not consider the time and space, i.e., computer memory, limit, which can occur in practice.

1.2 Information Learning Approach

In this section, we first talk about our approach and then discuss existing learning methods that based on information theory. Finally, we compare different approaches to computational learning theory.

1.2.1 Our Approach

In comparison with other approaches to the computational learning theory, our approach interprets and regards learning as a procedure to acquire information about the concept. Thus, we name our approach as Information Learning Approach (ILA). The underlying philosophy of this interpretation lies in that people learn a new concept and always try to understand it by gradually refining the acquired knowledge with more information about it. In ILA, the learner obtains information of the considering concept from variables, which describe the concept. After the knowledge is accumulated bit by bit, the uncertainty of the concept becomes less and less. The learning of this concept will continue until the acquired knowledge is sufficient to fully determine the concept.

First, we restate a theorem about the relationship between the mutual information $I(\mathbf{X}; Y)$ and the number of attributes in $\mathbf{X}$.

Theorem 1.2.1 ([122], p. 26) $I(\{\mathbf{X}, \mathbf{Z}\}; Y) \geq I(\mathbf{X}; Y)$*, with equality if and only if* $p(y|\mathbf{x}) = p(y|\mathbf{x}, \mathbf{z})$ *for all* $(\mathbf{x}, y, \mathbf{z})$ *with* $p(\mathbf{x}, y, \mathbf{z}) > 0$.

Proof of Theorem 1.2.1 can be found in [122]. In Theorem 1.2.1, it can be seen that $\{\mathbf{X}, \mathbf{Z}\}$ will contain more or equal information about Y as $\mathbf{X}$ does. To put it another way, the more variables, the more information is provided about another variable.

Theorem 1.2.1 reflects the learning process to a certain extent. Let us discuss the problem with an example. A computer in a laboratory is lost one day. Then, a policeman is sent to investigate the issue. He first needs to collect information about the computer. George may tell the policeman that the computer is an IBM PC. Alice may tell the policeman that the computer is black, and so on so forth. When more descriptions about the computer are obtained, the concept, i.e., the computer, becomes more and more distinct and specific, until finally the computer can fully be determined with the descriptions. Such a process is actually a procedure to obtain information about the concept. In the ILA, the descriptions provided by the people become descriptive variables in the data sets, the lost computer becomes the concept under consideration, and the knowledge contained in the descriptions of the people becomes the mutual information between the descriptive variables in the data sets and the concept. Information, or knowledge, is used to eliminate uncertainty, i.e., the entropy of the concept. The more information, the more specific and deterministic the concept is, i.e., the less the uncertainty of the concept becomes.

Then, to measure which subset of variables is complete and optimal, we restate the following theorem, which is the theoretical foundation of our approach.

Theorem 1.2.2 ([50], p. 43) *If the mutual information between $\mathbf{X}$ and Y is equal to the entropy of Y, i.e., $I(\mathbf{X}; Y) = H(Y)$, then Y is a function of $\mathbf{X}$.*

The entropy $H(Y)$ represents the diversity of the variable Y. The mutual information $I(\mathbf{X}; Y)$ represents the dependence or relation between vector $\mathbf{X}$ and Y. From this point of view, Theorem 1.2.2 actually says that the dependence between vector $\mathbf{X}$ and Y is very strong, such that there is no more diversity for Y if $\mathbf{X}$ has been known. In other words, the value of $\mathbf{X}$ can fully and completely determine the value of Y. If the concept is represented with the variable Y, then the learning process is becoming a process to find a subset of the descriptive features $\mathbf{X}$, which satisfies the criterion of Theorem 1.2.2. The features in $\mathbf{X}$ are called *Essential Attributes*, or EAs for short. For the above example, there must exist some essential attributes which will be of primary importance and can fully determine the lost computer, like the brand and the series number. If two persons have told the policeman these two properties of the lost computer, it will be unnecessary to talk to other people. Since the policeman can correctly identify the lost computer with the brand and the series number of it, other people who do not talk to the policeman will not provide any additional necessary information for the lost computer. For another example, humankind may have many properties or characters, like straight walking and having large brains, which are different from other species. But the most fundamental difference between the human being, *Homo sapiens*, and other species is the DNA content within our cells.

For a domain with n variables, $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$, there are 2^n subsets of $\mathbf{V}$. Obviously, the task to find the $\mathbf{X}$ which satisfies the criterion of Theorem 1.2.2 is NP-hard. Thus, we propose the Discrete Function Learning (DFL) algorithm to efficiently find the target subset $\mathbf{X}$. The DFL algorithm uses one adjustable parameter K, called expected cardinality of the EAs, to control the number of attributes in the target subset $\mathbf{X}$. The complexity of the DFL algorithm becomes polynomial after the introduction of K. Unlike other approaches to computational learning theory, what the DFL algorithm does is to directly learn the original functions based on Theorem 1.2.2, and not to perform approximations. After the learning process, the DFL algorithm determines the models which are given by the truth tables of the original functions.

When data sets are noisy, the equality between $I(\mathbf{X}; Y)$ and $H(Y)$ is not fulfilled. In these cases, we have to relax the requirement of Theorem 1.2.2 to obtain a best estimated result. Therefore, we introduce a method called ϵ value to deal with noisy training data sets in practice. In the ϵ value method, we attribute the missing part of the $H(Y)$, which is not captured by $\mathbf{X}$, to the noise in the data sets. Specifically, we let the missing part of $H(Y)$ be smaller than or equal to $\epsilon \times H(Y)$.

Next, we consider the prediction task. People make predictions based on their knowledge, or the information obtained in living process and stored as memories in their brains. Consider the example about the lost computer. After the properties of the lost computer are obtained, the policeman will use these properties to predict whether they have found the lost computer. Definitely, there would be many ways to use these properties. But the most accurate and convenient way is to check the essential properties which distinctly specify the lost computer, like the brand and the series number. If other properties, like the brand and the color, are used to perform the prediction, it is possible to find incorrect targets, since there likely are many computers with the same brand and color as the lost one. However, the brand and the series number is unique for every computer. For the example of human, it is the DNA of humankind, that completely and accurately differentiates us, *Homo sapiens*, from other species.

Formally, the prediction task is performed with the weighted 1-Nearest-Neighbor (1NN) [5] algorithm based on the Hamming distance [88].

1.2.2 Other Learning Methods Based On Information Theory

There have been many learning methods based on information theory. The C4.5 algorithm uses a variant of mutual information, called *information gain*, to build decision trees [140]. Some feature selection methods use the ranking of information gain [87, 115, 174], as to be discussed in Section 6.2.1. There are also some feature selection methods based on mutual information, such as the work by Dumais *et al.* [65], Yang and Pedersen [176], Vidal-Naquet and Ullman [169], Fleuret [68], Chow and Huang [45] and Peng *et al.* [133]. Generally, because these existing methods perform learning with mutual information or variants of it, thus both these methods and our approach belong to the category of *Information Learning Approach*. However, as shown in Section 1.2.1, our approach explicitly uses Theorem 1.2.1 and 1.2.2 as its theoretical foundation, which is different from other learning methods based on information theory. A detailed comparison between our approach and existing feature selection methods based on information theory is given in Section 6.5. In this thesis, we will use ILA to stand for our approach.

Table 1.1: The comparison of different approaches to computational learning theory. The columns Learning Power, Complexity, Approximate and Noisy represent the learning ability for the target functions, the time complexity of the learning algorithms, whether this approach approximates the target functions and whether this approach can deal with noisy data sets.

Approach	Learning Power	Complexity	Approximate	Noisy
PAC Theory	Restricted	Polynomial	Yes	No
VC Theory	Unrestricted	Polynomial	Yes	Yes
Bayesian inference	Unrestricted	NP-hard	Yes	Yes
Algorithmic Learning	Unrestricted	NP-hard	No/Yes	Yes
Information Learning	Restricted/ Unrestricted	Polynomial/ NP-hard	No/Yes	Yes

1.2.3 Comparisons of Different Learning Approaches

Finally, we compare different approaches to computational learning theory in Table 1.1. From Table 1.1, it can be seen that the PAC theory, the VC theory and the Bayesian inference are all approximation methods. In comparison, the ILA performs exact learning when $\epsilon = 0$ and estimates the original model when $\epsilon > 0$. Furthermore, the PAC cannot deal with noisy data sets, which restricts its applications in practice. The complexity of the Bayesian inference and algorithmic learning theory is NP-hard, which makes it very difficult to apply them to high-dimensional data sets. In comparison, the ILA can be both polynomial and NP-hard, when the target functions are restricted and unrestricted respectively.

1.3 Contributions

In this thesis, we made the following major contributions.

1. We propose a new approach, named as Information Learning Approach, to the computational learning theory. In our approach, learning is interpreted as a procedure to

obtain information of the concept. There have been some existing learning methods based on information theory, as discussed in Section 1.2.2. Our method, as well as the existing methods, can be categorized into *Information Learning Approach* in general. However, our approach is different from these existing methods in theoretical foundation.

2. We propose a new learning algorithm, the DFL algorithm, based on the ILA. We show that the DFL algorithm can be used to learn functions; to find qualitative models of gene regulatory networks; to solve classification problems; and to find discriminatory feature subsets for other classification algorithms. The searching schema of the DFL algorithm extends the classical greedy searching method [49] and has been demonstrated to be indispensable for solving some problems.

3. An open problem is partially solved with our approach. We prove that some Boolean functions can be learned with the DFL algorithm in $O(k \cdot (N + logn) \cdot n)$ time. Before this work, it is still an open problem to learn Boolean functions with fewer than or equal to k inputs in $o(N \cdot n^k)$ [1] time.

4. We discuss the completeness of features in machine learning. We prove that complete feature subsets for classification problems can be obtained with the DFL algorithm. In many benchmark data sets, the DFL algorithm finds informative and discriminatory feature subsets for other classification methods.

5. We discuss the *curse of dimensionality* with the ILA. We show that the DFL algorithm can obtain optimal and complete feature subsets, when training data sets are large enough. This is helpful for overcoming the *curse of dimensionality*, since the

[1]The o-notation is used to denote an upper bound that is not asymptotically tight [49]. $o(g(n))$ is formally defined as the set, $o(g(n)) = \{f(n)$: for any positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$.

reduction of dimension achieved by the DFL algorithm does not deteriorate the prediction performances, but actually improves them in most data sets used in this thesis.

6. We propose to use a new measure, the structure sensitivity, as the criterion to evaluate the performances of inference algorithms for learning qualitative models of Gene Regulatory Networks (GRNs).

7. We have developed a new machine learning software, the Discrete Function Learner (DFLearner) [193]. The DFLearner implements the DFL algorithm for performing classification, choosing discriminatory feature subsets and learning qualitative models of GRNs.

Other contributions include performing cancer classification based on microRNA [14, 15, 22, 181] gene expression profiles [190, 192], exploring essential attributes for detecting microRNA precursors [183], classification of RNA splicing sites [185, 193], classification of DNA sequences as promoter or non-promoter sequences of genes [185, 193], improved MDL (minimum description length) score for learning Bayesian networks [186], learning Boolean networks from noisy data sets with Karnaugh-maps [187], leukemia subtype identification with pair-wise classification [188] and finding important peptides for the binding of Human Leukocyte Antigen (HLA) alleles belonging to HLA-B7 supertype [182].

1.4 Outline

The outline of this thesis is as follows. In Chapter 2, we formally present the ILA and the DFL algorithm. The ILA is based on information theory [151], incorporating some ideas from graphical models [131]. This chapter aims at providing a framework of the ILA and the DFL algorithm, while minor modifications for different application are introduced in the following chapters. This chapter is based on our work [184, 189, 191].

In Chapter 3, we will apply the DFL algorithm to learning qualitative models of GRNs. GRNs is the underlying mechanism which control different expression patterns of the genes within a cell and the developmental processes of a species. The reasons why genes are expressed when and where they are in the spatial domains of the developing organism are revealed in network "architecture," that is, in the total aggregate pattern of regulatory linkages [54]. Thus, it is of primary importance to decipher the architecture of GRNs. We use the DFL algorithm to learn qualitative models of GRNs from gene expression data sets, which are assumed to be the products of the GRNs. This chapter is based on our work [184, 191, 193].

In Chapter 4, we will consider two open problems about learning Boolean functions under the context of learning Boolean networks as models of GRNs. We prove that the bounded OR/AND Boolean functions can be learned with the DFL algorithm in $O(k \cdot (N + logn) \cdot n)$ time. The experiment results soundly support our analysis about the complexity of the DFL algorithm for learning Boolean functions. We also show that the DFL algorithm can correctly find the original Boolean functions from noisy data sets. We also propose to use structure sensitivity as a criterion to evaluate the performance of the algorithm for inferring Boolean networks. This chapter is based on our work [184, 191].

In Chapter 5, we apply the DFL algorithm to solving classification problems. To solve classification problems, the weighted 1-Nearest-Neighbor algorithm is used to perform predictions. Twenty four benchmark data sets are used to validate our approach. Twenty data sets are classic machine learning data sets, and the remaining four are high-dimensional biological data sets. Our approach obtains comparable or more competitive prediction performances than those from well-known methods, while with lower-complexity models. This chapter is based on our work [183, 185, 188, 189, 192].

In Chapter 6, we apply the DFL algorithm to performing feature subset selection. In supervised learning, there are always irrelevant and redundant features in the data sets, which will deteriorate the performances of inference algorithms, in terms of both accuracies and efficiencies. When the data sets are high-dimensional, the problems introduced by irrelevant and redundant features become more serious, and sometimes, incur the *curse of dimensionality*. Thus, feature selection is critical for improving the performances of the inference algorithms, and for further alleviating the *curse of dimensionality*. The DFL algorithm can automatically eliminate the irrelevant and redundant features by choosing the feature subsets which provide all or most information about the class attribute. This chapter is based on our work [183, 189, 192].

Finally in Chapter 7, we discuss the relationships of the ILA and the DFL algorithm to the existing learning theories. We summarize the contributions of this thesis. Then, we discuss the limitations of our approach and some future directions. We also propose a conjecture about the complexity of the DFL algorithm for learning multi-value discrete functions. Lastly, we give a perspective about the applications of the ILA in the future and suggest the role of the ILA in helping us to understand learning.

This thesis has seven appendixes. The proofs for theorems in Chapter 2, 3 and 4 are given in Appendix A, B and C respectively. The references of the theorems that have been proposed are given in the names of them. The new theorems proposed in this thesis have no reference in their names. The detailed settings of the DFL algorithm for the data sets used in Chapter 5 and Chapter 6 are shown in Appendix D. An extended version of the DFL algorithm is given in Appendix E. The common notation of this thesis is listed in Appendix F. The abbreviations of this thesis are listed in Appendix G. The softwares, data sets, and supplementary results of this thesis are provides at the supplementary website[2] of this thesis.

[2]The supplements of this thesis are available at http://www.ntu.edu.sg/home5/pg04325488/thesis.htm.

Chapter 2

Information Learning Approach

In this chapter, we will present the Information Learning Approach (ILA) and the Discrete Function Learning (DFL) algorithm. The ILA is based on information theory [151], incorporating some ideas from graphical models [131].

This chapter is organized as follows. Section 2.1 will first describe necessary background knowledge of information theory and graphical models. Second, The ILA is proposed in Section 2.2. Third, The relation between ILA and Markov Blanket is discussed in Section 2.3. Fourth, the DFL algorithm is proposed in Section 2.4. Fifth, in Section 2.5, we introduce the ϵ value method for noisy data sets. Sixth, Section 2.6 discusses the selection of the parameters of the DFL algorithm. Section 2.7 introduces the weighted 1-Nearest-Neighbor algorithm for prediction. Next, two implementation issues are discussed in Section 2.8. Finally, we will summarize this chapter in the last section.

2.1 Background Knowledge

In this section, we introduce the background knowledge of information theory and some concepts in graphical models [131].

2.1.1 Fundamental Knowledge of Information Theory

We will first introduce some notation. We use capital letters to represent discrete random variables, such as X and Y; lower case letters to represent an instance of the random variables, such as x and y; bold capital letters, like $\mathbf{X}$, to represent a vector; and lower case bold letters, like $\mathbf{x}$, to represent an instance of $\mathbf{X}$. The cardinality of $\mathbf{X}$ is represented with $|\mathbf{X}|$. In the remainder parts of this paper, we denote the attributes except the class attribute as a set of discrete random variables $\mathbf{V} = \{X_1, \ldots, X_n\}$, the class attribute as variable Y. The entropy of X is represented with $H(X)$, and the mutual information between X and Y is represented with $I(X;Y)$. The entropy and mutual information estimated from data sets are empirical values $\hat{H}(\cdot)$ and $\hat{I}(\cdot;\cdot)$.

The entropy of a random variable X is defined in terms of probability of observing a particular value x of X as

$$H(X) = -\sum_x P(X=x) \log P(X=x).$$

Later, we will use $P(X)$ to denote the distribution of X and $p(x)$ to denote $P(X=x)$. For two variables, the joint entropy is defined in terms of the probabilities of all possible instances of the tuple (X,Y) as

$$H(X,Y) = -\sum_x \sum_y p(x,y) \log p(x,y).$$

Once we observed $X = x$, the uncertainty in Y is the entropy of the posterior distribution,

$$H(Y|X=x) = -\sum_y p(y|x) \log p(y|x). \tag{2.1}$$

The average value of Equation 2.1 over all possible values of X is the conditional entropy

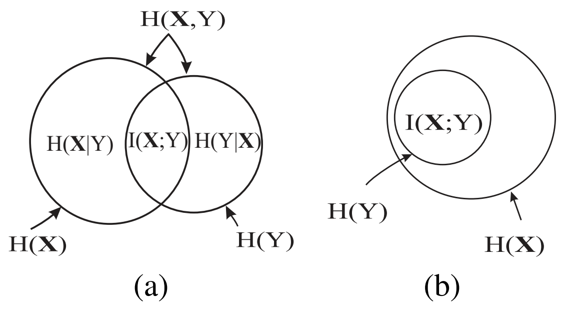


Figure 2.1: The relationship of entropy and mutual information. The circles represents the entropy of variables. The intersection between the circles stands for the mutual information between the variables. (a) The normal case. (b) When $Y = f(\mathbf{X})$.

of Y given X,

$$H(Y|X) = \sum_x p(x)H(Y|X = x).$$

If X becomes a set of variables $\mathbf{X} = \{X_{(1)}, X_{(2)}, \ldots, X_{(k)}\} \subseteq \mathbf{V}$, the conditional entropy of Y given $\mathbf{X}$ is defined as

$$H(Y|\mathbf{X}) = -\sum_{\mathbf{x}} \sum_{y} p(\mathbf{x}, y) \log p(y|\mathbf{x}). \tag{2.2}$$

Obviously, there are two conditional entropies which capture the relationships between $H(X)$ and $H(Y)$, $H(X|Y)$ and $H(Y|X)$. As illustrated in Figure 2.1 (a), these are related with Equation 2.3 [151]:

$$H(X, Y) = H(Y|X) + H(X) = H(X|Y) + H(Y). \tag{2.3}$$

In other words, the uncertainty of X and the remaining uncertainty of Y given knowledge of X, i.e., the information contained in Y that is not shared with X, sum to the entropy of the combination of X and Y. We can now find an expression for the shared or "mutual information", $I(X;Y)$, also referred to as "rate of transmission" between an input-output pair [151]:

$$I(X;Y) = H(Y) - H(Y|X) = H(X) - H(X|Y). \tag{2.4}$$

The shared information between X and Y corresponds to the remaining information of X if we remove the information of X that is not shared with Y. In other words, mutual information is the measure of the amount of information that one random variable contains about another random variable. From Equation 2.3 and Equation 2.4, the mutual information can also be represented as

$$I(X;Y) = H(X) + H(Y) - H(X,Y). \tag{2.5}$$

Similar to Equation 2.2, the mutual information between a vector $\mathbf{X}$ and Y is defined as

$$\begin{aligned} I(\mathbf{X};Y) &= H(Y) - H(Y|\mathbf{X}) = H(\mathbf{X}) - H(\mathbf{X}|Y) \qquad (2.6) \\ &= H(\mathbf{X}) + H(Y) - H(\mathbf{X},Y) = \sum_{\mathbf{x}} \sum_{y} p(\mathbf{x},y) \log \frac{p(\mathbf{x},y)}{p(\mathbf{x})p(y)}. \end{aligned}$$

From Figure 2.1 and Equation 2.6, it is clear that $I(\mathbf{X};Y) = I(Y;\mathbf{X})$.

The conditional entropy and entropy are related with Theorem 2.1.1, for which the proof is available in [50].

Theorem 2.1.1 ([50], p. 27) $H(\mathbf{X}|Y) \leq H(\mathbf{X})$ *with equality if and only if X and Y are independent.*

The conditional mutual information of random variable Y and Z given $\mathbf{X}$ is defined by

$$I(Y;Z|\mathbf{X}) = H(Y|\mathbf{X}) - H(Y|\mathbf{X},Z) = \sum_{\mathbf{x}} \sum_{y} \sum_{z} p(\mathbf{x},y,z) \log \frac{p(y,z|\mathbf{x})}{p(y|\mathbf{x})p(z|\mathbf{x})}. \tag{2.7}$$

From Equation 2.7, it is obvious that $I(Y;Z|\mathbf{X}) = I(Z;Y|\mathbf{X})$.

The chain rule for mutual information is give by Theorem 2.1.2, for which the proof is available in [50].

Theorem 2.1.2 ([50], p. 22) $I(X_1, X_2, \ldots, X_n; Y) = \sum_{i=1}^{n} I(X_i; Y|X_{i-1}, X_{i-2}, \ldots, X_1)$.

Theorem 2.1.3 ([50], p. 27) *For any discrete random vectors Y and $\mathbf{Z}$, $I(Y;\mathbf{Z}) \geq 0$. Moreover, $I(Y;\mathbf{Z}) = 0$ if and only if Y and $\mathbf{Z}$ are independent.*

Proof of Theorem 2.1.3 can also be found in [50]. Immediately from Theorem 2.1.3, the following corollary is also correct.

Corollary 2.1.1 ([50], p. 27) $I(Y;\mathbf{Z}|\mathbf{X}) \geq 0$*, with equality if and only if Y and $\mathbf{Z}$ are conditional independent given $\mathbf{X}$.*

We will review *conditional independence*, introduced by Pearl [131], in Section 2.1.2.

Let us recall the *law of large numbers* [1] which states that if an event of probability p is observed repeatedly during independent repetitions, the ratio of the observed frequency of that event to the total number of repetitions converges towards p as the number of repetitions becomes arbitrarily large [1–3]. Formally, the *law of large numbers* is given in Theorem 2.1.4.

[1]The law of large numbers was first proved by the Swiss mathematician Jakob Bernoulli in 1713 [1]. There is also a more general version of the law of large numbers for averages, proved more than a century later by the Russian mathematician Pafnuty Chebyshev [2]. There are two versions of the Law of Large Numbers, one called the "weak" law and the other the "strong" law. In essence the two laws do not describe different actual laws but instead refer to different ways of describing the convergence of the sample mean with the population mean [1]. The weak law of large numbers states that if $x_1, x_2, \ldots, x_n$, is an infinite sequence of random variables, where all the random variables have the same expected value μ and variance σ^2; and are uncorrelated (i.e., the correlation between any two of them is zero), then the sample average $\overline{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$, for any positive number ϵ, no matter how small, we have $\lim_{n\to\infty} p(|\overline{x}_n - \mu| < \epsilon) = 1$ [1, 70]. On the same condition, the strong law of large numbers states that $p(\lim_{n\to\infty} \overline{x}_n = \mu) = 1$ [1].

Theorem 2.1.4 (Law of Larger Numbers) *Let* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ *be a infinite sequence of random vector* $\mathbf{X}$*,* $\mathbf{x}_i$*s are independently drawn from the same distribution* $P(\mathbf{X})$*, then*

$$\lim_{N\to\infty} \widehat{p}(\mathbf{x}) = \frac{N_{\mathbf{x}}}{N} = p(\mathbf{x}),$$

where $N_{\mathbf{x}}$ *is the number of instances in which* $\mathbf{X} = \mathbf{x}$*.*

Note the Theorem 2.1.4 is correct regardless of sample distribution. From Theorem 2.1.4, it is known that if there are enough samples, $p(\mathbf{x})$ can be correctly estimated. Then, if the variables in $\mathbf{X}$ and in $\mathbf{V}\setminus\mathbf{X}$ are independent (see supplementary Figure S1), from Corollary 2.1.1, we can have Theorem 2.1.5.

Theorem 2.1.5 *Let* $\mathbf{V} = \{X_1, \ldots, X_n\}$*,* $\forall \mathbf{Z} \in \mathbf{V} \setminus \mathbf{X}$*,* $\mathbf{X}$ *and* $\mathbf{Z}$ *are independent. Given enough samples of* $\mathbf{V}$*. If* Y *and* $\mathbf{Z}$ *are conditional independent given* $\mathbf{X}$*, then the empirical mutual information* $\hat{I}(Y;\mathbf{Z}) = 0$*.*

In a function, we have the following theorem. The relation between $I(\mathbf{X};Y)$ and $H(Y)$ is shown in Figure 2.1 (b).

Theorem 2.1.6 ([83], p. 37) *If* $Y = f(\mathbf{X})$*, then* $I(\mathbf{X};Y) = H(Y)$*.*

2.1.2 Background knowledge of Graphical Models

Conditional Independence (see [131], p. 83) is a concept used in graphical models, especially Bayesian networks [131].

Definition 2.1.1 (Conditional Independence) *Let* $\mathbf{V} = \{X_1, \ldots, X_n\}$ *and* $P(\cdot)$ *be a joint probability function over the variables in* $\mathbf{V}$*.* $\forall \mathbf{X}, \mathbf{Y}$*, and* $\mathbf{Z} \subseteq \mathbf{V}$*, the sets* $\mathbf{Y}$ *and* $\mathbf{Z}$ *are said to be* conditional independent given $\mathbf{X}$ *if*

$$P(\mathbf{Y}|\mathbf{X},\mathbf{Z}) = P(\mathbf{Y}|\mathbf{X}). \tag{2.8}$$

In other words, learning the value of $\mathbf{Z}$ does not provide additional information about $\mathbf{Y}$, once we know $\mathbf{X}$.

A Bayesian network for $\mathbf{V}$ is the tuple $B(G, \Theta)$. G is a Directed Acyclic Graph (DAG) whose nodes are in one-to-one correspondence to variables in $\mathbf{V}$, whose edges encode the conditional dependence between variables [131]. In particular, X_i is independent of its non-descendants given its parents $\mathbf{Pa}_i$ in G [131]. The second component Θ is a set of parameters which quantify the network. In particular, $\Theta = \bigcup_i \Theta_i$, where Θ_i is the Conditional Probability Table (CPT) of node X_i. The Bayesian network B encodes the joint probability over $\mathbf{V}$ by the following equation

$$P_B(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \mathbf{Pa}_i, \Theta_i) \tag{2.9}$$

where $\mathbf{Pa}_i$ is the set of parents of node X_i in G. From Equation 2.9, it can be seen that the joint probability of $\mathbf{V}$ is decomposed into the product of marginal distributions. An example of Bayesian networks is given in Figure 2.2.

2.2 Information Learning Approach

In this section, we will first define the learning problem. Then, we will introduce the ILA.

2.2.1 Problem Definition

Let $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$ be a domain with n discrete variables and let Y be a function of $\mathbf{X} \subseteq \mathbf{V}$. We denote the cardinality of $\mathbf{X}$ with k, i.e., $|\mathbf{X}| = k$ and $k \leq n$. Let the training samples be generated in such a way that variables in $\mathbf{V}$ are assigned according to a distribution $P(\mathbf{V})$ and y is generated with x. We will consider the learning problem defined as follows.

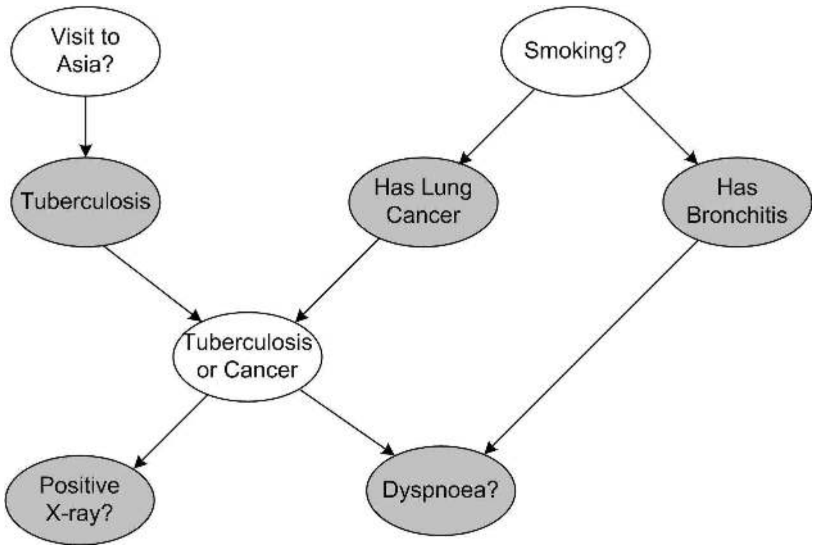


Figure 2.2: The Asia Bayesian network, a fictitious expert system representing the diagnosis of a patient, having just come back from a trip to Asia and showing dyspnoea [107]. The *Markov Blanket* of node *Tuberculosis or Cancer* is the set {*Tuberculosis*, *Has Lung Cancer*, *Has Bronchitis*, *Positive X-ray?*, *Dyspnoea?*}(the gray nodes in the figure).

Definition 2.2.1 (The Learning Problem) *Given a training data sets* $\mathbf{T} = \{(\mathbf{v}_i, y_i) : i = 1, \ldots, N\}$*, find a function* $Y = f(\mathbf{U})$ *where* $\mathbf{U} \subseteq \mathbf{V}$*, such that* f *can produce the same output value as those in* $\mathbf{T}$ *as frequently as possible. Formally,*

$$f = \arg\max_{g} Accuracy(g) = \frac{N_C(g)}{N}$$

where $N_C(g)$ *is the number of samples whose* $g(\mathbf{u}_i) = y_i$*.*

Note that $\mathbf{U}$ may be different from the input $\mathbf{X}$ of the original function, which indicates the failure or partial failure of the learning process (we will discuss this issue in detail in Chapter 4). It is often the case that there are some irrelevant and redundant features in the domain $\mathbf{V}$, since $\mathbf{X} \subseteq \mathbf{V}$. The training data sets may also include some noise, i.e., some y_i in the pair $(\mathbf{v}_i, y_i)$ is not the value produced with $Y = f(\mathbf{X})$.

2.2.2 The Information Learning Approach

We restate a theorem about the relationship between the mutual information $I(\mathbf{X}; Y)$ and the number of attributes in $\mathbf{X}$.

Theorem 2.2.1 ([122], p. 26) $I(\{\mathbf{X}, \mathbf{Z}\}; Y) \geq I(\mathbf{X}; Y)$*, with equality if and only if* $p(y|\mathbf{x}) = p(y|\mathbf{x}, \mathbf{z})$ *for all* $(\mathbf{x}, y, \mathbf{z})$ *with* $p(\mathbf{x}, y, \mathbf{z}) > 0$.

In Theorem 2.2.1, it can be seen that $\{\mathbf{X}, \mathbf{Z}\}$ will contain more or equal information about Y as $\mathbf{X}$ does. Intuitively, it can be illustrated in Figure 2.4, $H(\mathbf{X}, \mathbf{Z})$ will definitely share no less information with $H(Y)$ than $H(\mathbf{X})$ does alone, since $H(\mathbf{X}, \mathbf{Z})$ can provide at least the part of information about Y already provided by $H(\mathbf{X})$ alone. To put it another way, the more variables, the more information is provided about another variable.

From Theorem 2.2.1, it can be deduced that individual variables cannot provide more information about the class attribute than vectors, i.e., a collective of variables. As to be demonstrated in Figure 2.4, it is obvious that choosing the top variables will not make sure that we find the optimal subset of features which contains maximum mutual information with the class attribute. Therefore, it is better to find the optimal subset of variables by considering the collective effect of variables as vectors.

To measure which subset of variables is optimal, we restate the following theorem, which is the theoretical foundation of our algorithm.

Theorem 2.2.2 ([50], p. 43) *If the mutual information between* $\mathbf{X}$ *and* Y *is equal to the entropy of* Y*, i.e.,* $I(\mathbf{X}; Y) = H(Y)$*, then* Y *is a function of* $\mathbf{X}$.

The entropy $H(Y)$ represents the diversity, or randomness, of the variable Y. The mutual information $I(\mathbf{X};Y)$ represents the dependence between $\mathbf{X}$ and Y. From this point of view, Theorem 2.2.2 actually says that the dependence between vector $\mathbf{X}$ and Y is very strong, such that there is no more diversity for Y if $\mathbf{X}$ has been known. In other words, the value of $\mathbf{X}$ can fully determine the value of Y. We will name the subset of features $\mathbf{X}$ which satisfies the criterion of Theorem 2.2.2 as the *Essential Attributes*, or EAs for short.

More intuitively as shown in Figure 2.1 (b), if the mutual information between a set of variables $\mathbf{X}$ and another variable Y, $I(\mathbf{X};Y)$, is equal to the entropy of Y, $H(Y)$, then Y is fully determined by $\mathbf{X}$, i.e., $\mathbf{X}$ gives all the information needed to decide the state of Y. That is to say, Y is a function of $\mathbf{X}$.

Next, we discuss the probabilistic relationship between $\mathbf{X}$, Y and another vector $\mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$.

Theorem 2.2.3 *If* $I(\mathbf{X};Y) = H(Y)$, $\mathbf{X} = \{X_{(1)}, \ldots, X_{(k)}\}, \forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$, *$Y$ and $\mathbf{Z}$ are conditional independent given* $\mathbf{X}$.

From Theorem 2.2.3 and Corollary 2.1.1, we have the Corollary 2.2.1.

Corollary 2.2.1 *If* $I(\mathbf{X};Y) = H(Y)$, $\mathbf{X} = \{X_{(1)}, \ldots, X_{(k)}\}$, $\forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$, *then* $I(Y;\mathbf{Z}|\mathbf{X}) = 0$.

Theorem 2.2.3 is actually saying that if we can find a subset of features $\mathbf{X}$ which satisfies $I(\mathbf{X};Y) = H(Y)$, the remaining variables in $\mathbf{V}$ will be prevented from giving additional information about Y, once we know $\mathbf{X}$. Immediately from Theorem 2.1.5 and Corollary 2.2.1, we have Theorem 2.2.4.

Theorem 2.2.4 *Let* $\mathbf{V} = \{X_1, \ldots, X_n\}$, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{X}$, *$\mathbf{X}$ and $\mathbf{Z}$ are independent. Given enough samples of* $\mathbf{V}$. *If* $I(\mathbf{X};Y) = H(Y)$, *then* $\forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$, *the empirical mutual information* $\hat{I}(Y;\mathbf{Z}) = 0$.

From Theorem 2.2.4, it is known that if the sample size is large enough, the variables in $\mathbf{V}\backslash\mathbf{X}$ will share no mutual information with the variable Y under consideration. As to be shown in Figure 2.3, $\hat{I}(Y;\mathbf{Z})$ may be larger than zero when sample size is small, and tends to be zero when sample size is large. This suggests that it is more advisable to use large training data sets to determine the correct classification functions and optimal feature subsets, as the negative effects from irrelevant features, $\hat{I}(Y;\mathbf{Z})$, are minimized when sample size is large.

Based on Theorem 2.2.2, it is known that if $I(\mathbf{X};Y) = H(Y)$, then the subset $\mathbf{X}$ can fully determine the value of Y for the training data sets. And from Theorem 2.2.4, it is known that if $I(\mathbf{X};Y) = H(Y)$, then $\forall\mathbf{Z} \subseteq \mathbf{V}\backslash\mathbf{X}$, $\mathbf{Z}$ provides no information of Y. Hence, the $\mathbf{X}$ which satisfies $I(\mathbf{X};Y)$ is the complete set of features needed to determine the value of Y for the training data set.

From Theorem 2.2.2, the learning problem in Definition 2.2.1 is converted to finding a subset of attributes $\mathbf{U} \subseteq \mathbf{V}$ whose mutual information with Y is equal to the entropy of Y, where the $\mathbf{U}$ is the EAs which we are trying to find from the data sets. For n discrete variables, there are totally 2^n subsets. Clearly, it is NP-hard to examine all possible subsets exhaustively. However, since there usually are irrelevant and redundant features in data sets, it is reasonable to reduce the searching space by considering those subsets with limited number of features. Hence, the problem can be solved in polynomial time. To efficiently find the $\mathbf{U}$, we will propose the DFL algorithm in Section 2.4.

2.3 Relation to Markov Blanket

In this section, we will first briefly introduce the *Markov Blanket*. Then, we will discuss the relation between the ILA and *Markov Blanket*.

2.3.1 Markov Blanket

Markov Blanket [131] is defined as follows.

Definition 2.3.1 (Markov Blanket) *Let $\mathbf{U}$ be some set of features(variables) which does not contain X_i. We say that $\mathbf{U}$ is a* Markov Blanket *for X_i if X_i is conditional independent of $\mathbf{R} = \mathbf{V} \setminus \{\mathbf{U} \cup \{X_i\}\}$[2] given $\mathbf{U}$, i.e.,*

$$p(x_i|\mathbf{r}, \mathbf{u}) = p(x_i|\mathbf{u}), \forall p(\mathbf{r}, \mathbf{u}) > 0. \tag{2.10}$$

A set is called a Markov boundary of X_i, if it is a minimum Markov Blanket *of X_i, i.e., none of its proper subsets satisfy Equation 2.10.(see [131], p. 97).*

In classification problems, the data sets can be assumed to be generated by a joint distribution of $\mathbf{V}$ and Y. Thus, if the $\mathbf{V}$ and Y are regarded as nodes in a Bayesian network, the learning algorithms try to find the conditional probability of Y given $\mathbf{V}$, $P(Y|\mathbf{V})$. In Bayesian networks, the union of parents and children of Y, and parents of children (spouses) of Y is equivalent to the *Markov Blanket* [131, 177]. As shown in Figure 2.2, the *Markov Blanket* of the node *Tuberculosis or Cancer* is the set {*Tuberculosis*, *Has Lung Cancer*, *Has Bronchitis*, *Positive X-ray?*, *Dyspnoea?*}(the gray nodes in Figure 2.2).

From the definition of *Markov Blanket*, it is known that if we can find a *Markov Blanket* $\mathbf{U}$ for the class attribute Y, then all other variables in $\mathbf{V}$ will be statistically independent of Y given $\mathbf{U}$. This means that all the information that may influence the value of Y is stored in values of $\mathbf{U}$ [177]. In other words, *Markov Blanket* $\mathbf{U}$ has prevented other nodes from affecting the value of Y. *Markov Blanket* $\mathbf{U}$ also corresponds to strongly relevant features [163], as defined by Kohavi and John [103]. Therefore, if we can find a *Markov*

[2] *In [131], "-" is used to denote the set minus(difference) operation. To be consistent to other parts of this thesis, we will use "$\setminus$" to denote the set minus operation. Particularly, $\mathbf{A} \setminus \mathbf{B}$ is defined by $\mathbf{A} \setminus \mathbf{B} = \{X : X \in \mathbf{A}$ and $X \notin \mathbf{B}\}$.*

Blanket $\mathbf{U}$ of Y as the candidate feature subsets, $\mathbf{U}$ should be the theoretical optimal subset of features to predict the value of Y, as discussed in [104, 163].

2.3.2 The Relation of Information Learning Approach and Markov Blanket

From Theorem 2.2.3, it is known that if $I(\mathbf{X};Y) = H(Y)$, then $\forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$, Y and $\mathbf{Z}$ are conditional independent given $\mathbf{X}$. From the concept of *Markov Blanket*, it is known that if $I(\mathbf{X};Y) = H(Y)$, then $\mathbf{X}$ is a *Markov Blanket* of Y. Formally, we have

Theorem 2.3.1 *If* $I(\mathbf{X};Y) = H(Y)$*, then* $\mathbf{X}$ *is a* Markov Blanket *of* Y.

As to be introduced in Section 2.5, $I(\mathbf{X};Y) = H(Y)$ can be satisfied only when the data sets are noiseless. However, with the introduction of ϵ method in Section 2.5, the set that carries most information of Y, $H(Y)$, is still a good estimation of the true *Markov Blanket* of Y. In addition, our method has competitive expected computational costs to other methods for finding *Markov Blankets*, such as in [10, 104, 163, 164].

2.4 The Discrete Function Learning Algorithm

In this section, we will first discuss the theoretical motivation of the DFL algorithm, then introduce the DFL algorithm. Next, we analyze the complexity and correctness of the DFL algorithm.

2.4.1 Theoretical Motivation

From Theorem 2.1.3 and Theorem 2.2.4, the irrelevant features tend to share zero or very small mutual information with the class attribute in the presence of noise. Therefore, the irrelevant features can be eliminated by choosing those features with relatively large mutual information with the class attribute in modelling process.

When choosing candidate features, our approach maximizes the mutual information between the feature subsets and the class attribute. Suppose that $\mathbf{U}_{s-1}$ has already been selected at the step $s-1$, and the DFL algorithm is trying to add a new feature $X_i \in \mathbf{V} \setminus \mathbf{U}_{s-1}$ to $\mathbf{U}_{s-1}$. Specifically, our method uses $X_{(1)} = \arg\max_i I(X_i;Y)$ and Equation 2.11 as criterion to add new features to $\mathbf{U}$.

$$X_{(s)} = \arg\max_i I(\mathbf{U}_{s-1}, X_i; Y), \tag{2.11}$$

where $\forall s$, $1 < s \leq k$, $\mathbf{U}_1 = \{X_{(1)}\}$, and $\mathbf{U}_s = \mathbf{U}_{s-1} \cup \{X_{(s)}\}$. From Equation 2.11, it is obvious that the irrelevant features have lost the opportunity of being chosen as EAs of the classifiers after the first EA, $X_{(1)}$, is chosen, since $I(X_i;Y)$ is very small if X_i is an irrelevant feature.

For instance, the mutual information between X_i and the class attribute Y for the LED+17 data sets from the UCI machine learning repository [31] is shown in Figure 2.3. In the LED+17 data sets, there are 7 relevant features, each representing a Light Emitting Diode (LED), and 17 irrelevant features with random values. The 7 relevant features are boolean, with "1" representing that the LED is turned on and "0" representing that the LED is turned off. With different instances of the 7 relevant features, the class attribute can take 10 values, each for one decimal digit. The Golden Rule in Figure 2.3 is obtained from a data set with 10 samples, each for one decimal digit. From Figure 2.3, it is shown that the relevant features have significantly larger $I(X_i;Y)$ than irrelevant ones. The noise makes $I(X_i;Y)$ of relevant features much smaller than the Golden Rule, but still larger than the values of irrelevant ones. Especially, when sample size is large, $I(X_i;Y)$ for irrelevant features is almost zero, which is due to Theorem 2.2.4. In addition, the curves for large

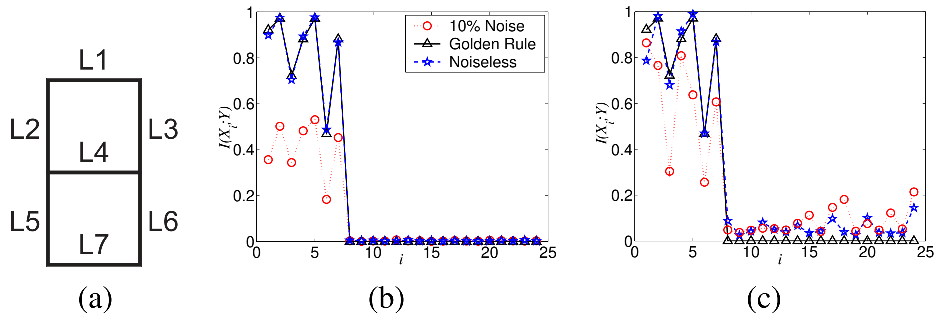


Figure 2.3: The mutual information between X_i and the class attribute Y for the LED+17 data sets. The horizontal axis represents the index i of the features X_is. The curves marked with triangles, pentagrams and circles represents the values for the Golden Rule, the noiseless data sets and data sets with 10% noise. (a) The 7 relevant attribute of the LED+17 data sets. (b) $I(X_i;Y)$ when the sample size is 2000. The one with 10% noise is used in the experiments in Section 6.4. (c) $I(X_i;Y)$ when the sample size is 100.

data sets is quite similar to the Golden Rule.

Next, we illustrate how to eliminate the redundant features. From Theorem 2.1.2, we have

$$I(\mathbf{U}_{s-1}, X_i; Y) = I(\mathbf{U}_{s-1}; Y) + I(X_i; Y|\mathbf{U}_{s-1}). \tag{2.12}$$

In Equation 2.12, note that $I(\mathbf{U}_{s-1}; Y)$ does not change when trying different $X_i \in \mathbf{V} \setminus \mathbf{U}_{s-1}$. Hence, the maximization of $I(\mathbf{U}_{s-1}, X_i; Y)$ in our method is actually maximizing $I(X_i; Y|\mathbf{U}_{s-1})$, as shown by the shaded region in Figure 2.4, which is the conditional mutual information of X_i and Y given the already selected features $\mathbf{U}_{s-1}$, i.e., the information of Y not captured by $\mathbf{U}_{s-1}$ but carried by X_i. As shown in Figure 2.4 (b), if the new feature B is a redundant feature, i.e., $I(\mathbf{U}_{s-1}; B)$ is large, then the additional information of Y carried by X_i, $I(B; Y|\mathbf{U}_{s-1})$, will be small. Consequently, B is unlikely to be chosen as an *Essential Attribute* (EA). Hence, the redundant features are automatically eliminated by

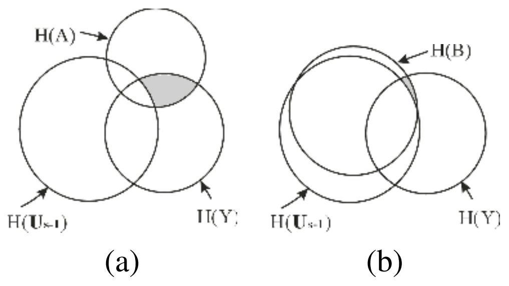


Figure 2.4: The advantage of using mutual information to choose the most discriminatory feature vectors. The notations are the same as those in Figure 2.1. $\mathbf{U}_{s-1}$ is the features already chosen at step $s-1$. The shaded regions represent $I(X_i; Y|\mathbf{U}_{s-1})$, where $X_i \in \mathbf{V} \setminus \mathbf{U}_{s-1}$. (a) When $X_i = A$. A shares less mutual information with Y than B does. However, the vector $\{\mathbf{U}_{s-1}, A\}$ shares larger mutual information with Y than the vector $\{\mathbf{U}_{s-1}, B\}$ does. (b) When $X_i = B$. B shares larger mutual information with Y than A does. But B and $\mathbf{U}_{s-1}$ have a large mutual information, which means that $\mathbf{U}_{s-1}$ has contained most of the information of Y carried by B or the additional information of Y carried by B, $I(B; Y|\mathbf{U}_{s-1})$, is small.

maximizing $I(\mathbf{U}_{s-1}, X_i; Y)$.

Figure 2.4 also illustrates one short-coming of the top-ranking methods based on mutual information and other measures. The feature A in Figure 2.4 (a) does not share more mutual information with Y than B in part (b) does. If the top-ranking method is used, then B in part (b) will be chosen. However, if features are evaluated as vectors, then $\{\mathbf{U}_{s-1}, A\}$ in part (a) is a better subset than $\{\mathbf{U}_{s-1}, B\}$ in part (b).

Based on Theorem 2.3.1, it is known that if $I(\mathbf{X}; Y) = H(Y)$, then $\mathbf{X}$ is a *Markov Blanket* of Y. Hence, in the feature selection process of our method, the $I(\mathbf{U}; Y)$ is evaluated with respect to $H(Y)$. It is critical to compare $I(\mathbf{U}; Y)$ with $H(Y)$. Since by performing this comparison, the optimal subset of features can be automatically determined, without the need of specifying a predefined number of features or a threshold value of the mutual

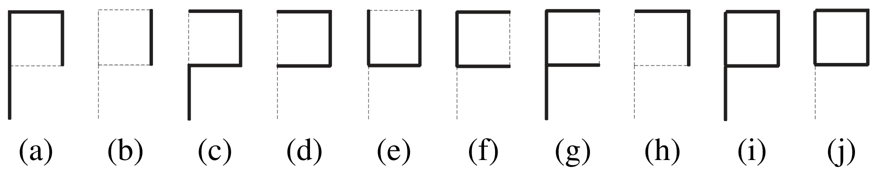


Figure 2.5: The rules of the *noiseless* LED+17 data set learned by the DFL algorithm. The positions of the attributes are the same as those in Figure 2.3 (a). Part (a) to (j) corresponds to rules for 0 to 9 respectively. Wide solid and narrow dashed lines mean that the corresponding attributes take the value of 1 and 0 respectively.

information.

We also use the example of LED+17 to illustrate the removing of redundant features. For the *noiseless* LED+17 data sets, the DFL algorithm finds that only 5 features, L1 to L5, is sufficient to build classification models, with the learned rules shown in Figure 2.5. For example, in Figure 2.5 (a), L1, L2, L3 and L5 are turned on and L4 is turned off, which means that the rule for digit "0" is (L1,L2,L3,L4,L5,Y) = (1,1,1,0,1,0). Each rule in Figure 2.5 uniquely corresponds to one digit from 0 and 9, which means these rules can correctly classify the noiseless LED+17 data sets. However, the generation function of the LED+17 data sets has 7 relevant features. This difference means that the 7 relevant features contain redundant information for the class attribute, which is almost impossible or very hard to detect explicitly. Fortunately, by evaluating $I(\mathbf{U};Y)$ with respect to $H(Y)$, this redundancy is automatically detected.

In summary, the irrelevant and redundant features can be automatically removed, if the new candidate feature X_i is evaluated with respect to the selected features as a vector $\mathbf{U}_{s-1}$ by maximizing $I(\mathbf{U}_{s-1}, X_i; Y)$. Furthermore, the optimal subset of features can be determined by evaluating the $I(\mathbf{U};Y)$ with respect to $H(Y)$.

Table 2.1: The DFL algorithm.

```
   Algorithm: DFL($\mathbf{V}, K, \mathbf{T}$)
   Input: a list $\mathbf{V}$ with $n$ variables, indegree $K$,
   $\mathbf{T} = \{(\mathbf{v}_i, y_i) : i = 1, \cdots, N\}$. $\mathbf{T}$ is global.
   Output: $f$
   Begin:
1    $L \leftarrow$ all single element subsets of $\mathbf{V}$;
2    $\Delta Tree.FirstNode \leftarrow L$;
3    calculate $H(Y)$;          //from $\mathbf{T}$
4    $D \leftarrow 1$;          //initial depth
5*   $f = Sub(Y, \Delta Tree, H(Y), D, K)$;
6   return $f$;
   End
```

* $Sub()$ is a subroutine listed in Table 2.2.

2.4.2 The Discrete Function Learning Algorithm

The main steps of the DFL algorithm are listed in Table 2.1. The DFL algorithm has two parameters, the expected cardinality K and the ϵ value. The ϵ value will be elaborated in Section 2.5.

The K is the expected maximum number of attributes in the classifier. The DFL algorithm uses the K to prevent the exhaustive searching of all subsets of attributes by checking those subsets with fewer than or equal to K attributes. When trying to find the EAs from all combinations whose cardinalities are not larger than K, the DFL algorithm will examine the mutual information between the combination of variables under consideration, $\mathbf{U}$, and the class attribute, Y. If $I(\mathbf{U}; Y) = H(Y)$, then the DFL algorithm will terminate its searching process, and obtain the classifiers by deleting the non-essential attributes and duplicate instances of the EAs in the training data sets. Meanwhile, the counts of different instances of $(\mathbf{U}, Y)$ are stored in the classifiers and will be used in the prediction process. In the algorithm, we use the following definitions.

Definition 2.4.1 (δ Superset) *Let $\mathbf{X}$ be a subset of $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$, then $\delta_i(\mathbf{X})$ of*

Table 2.2: The subroutine of the DFL algorithm.

```
     **Algorithm:** $Sub(Y, \Delta Tree, H, D, K)$
     **Input:** variable $Y$, $\Delta Tree$, entropy $H(Y)$
     current depth $D$, maximum indegree $K$
     **Output:** function table for $Y$, $Y = f(\mathbf{X})$
     **Begin:**
1    $L \leftarrow \Delta Tree.DthNode$;
2    **for** every element $\mathbf{X} \in L$ {
3        calculate $I(\mathbf{X}; Y)$;     //from $\mathbf{T}$
4        **if**($I(\mathbf{X}; Y) == H$) {   //from Theorem 2.2.2
5            extract $Y = f(\mathbf{X})$ from $\mathbf{T}$;
6            **return** $Y = f(\mathbf{X})$ ;
         }
     }
7    sort $L$ according to $I$;
8    **for** every element $\mathbf{X} \in L$ {
9        **if**($D < K$){
10           $D \leftarrow D + 1$;
11           $\Delta Tree.DthNode \leftarrow \Delta_1(\mathbf{X})$;
12           **return** $Sub(Y, \Delta Tree, H, D, K)$;
         }
     }
13   **return** "Fail(Y)";     //fail to find function for $Y$
     **End**
```

$\mathbf{X}$ *is a superset of* $\mathbf{X}$ *so that* $\mathbf{X} \subset \delta_i(\mathbf{X})$ *and* $|\delta_i(\mathbf{X})| = |\mathbf{X}| + i$.

Definition 2.4.2 (Δ Supersets) *Let* $\mathbf{X}$ *be a subset of* $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$*, then* $\Delta_i(\mathbf{X})$ *of* $\mathbf{X}$ *is the collective of all* $\delta_i(\mathbf{X})$ *and* $\Delta_i(\mathbf{X}) = \bigcup \delta_i(\mathbf{X})$.

Definition 2.4.3 (Searching Layer $\mathcal{L}$ of V) *Let* $\mathbf{X} \subseteq \mathbf{V}$*, then the* i*th layer* $\mathcal{L}_i$ *of all subsets of* $\mathbf{V}$ *is,* $\forall|\mathbf{X}| = i$*,* $\mathcal{L}_i = \bigcup \mathbf{X}$.

Definition 2.4.4 (Searching Space) *The searching space of functions with a bounded indegree* K *is* $\mathcal{S}_K = \bigcup_{i=1}^{K} \mathcal{L}_i$.

From Definition 2.4.3, it is known that there are $\binom{n}{i}$ subsets of $\mathbf{V}$ in $\mathcal{L}_i$. And there are $\sum_{i=1}^{K} \binom{n}{i} \approx n^K$ subsets of $\mathbf{V}$ in $\mathcal{S}_K$.

Table 2.3: The training data set $\mathbf{T}$ of the example to learn $Y = (A \cdot C) + (A \cdot D)$.

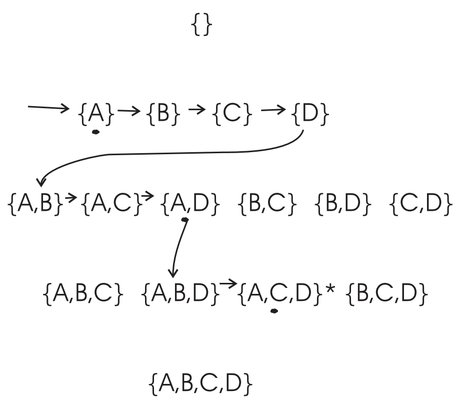
ABCD	*Y*	*ABCD*	*Y*	*ABCD*	*Y*	*ABCD*	*Y*
0000	0	0100	0	1000	0	1100	0
0001	0	0101	0	1001	1	1101	1
0010	0	0110	0	1010	1	1110	1
0011	0	0111	0	1011	1	1111	1

To clarify the search process of the DFL algorithm, let us consider an example, as shown in Figure 2.6. In this example, the set of attributes is $\mathbf{V} = \{A, B, C, D\}$ and the class attribute is determined with $Y = (A \cdot C) + (A \cdot D)$, where "$\cdot$" and "+" are logic AND and OR operation respectively. The expected cardinality K is set to 4 for this example. The training data set $\mathbf{T}$ of this example is shown in Table 2.3.

The search procedure of the DFL algorithm for this example is shown in Figure 2.6. In the learning process, the DFL algorithm uses a data structure called $\Delta Tree$ to store the Δ supersets in the searching process. For instance, the $\Delta Tree$ when the DFL algorithm is learning the Y is shown in Figure 2.7.

As shown in Figure 2.6 and 2.7, the DFL algorithm searches the first layer $\mathcal{L}_1$, then it sorts all subsets according to their mutual information with Y on $\mathcal{L}_1$. From Theorem 2.2.4, A, C or D has larger mutual information with Y than B has. Consequently, the DFL algorithm finds that $\{A\}$ shares the largest mutual information with Y among subsets on $\mathcal{L}_1$.

Next, the $\Delta_1(A)$s are added to the second layer of $\Delta Tree$, as shown in Figure 2.7. Similarly to $\mathcal{L}_1$, the DFL algorithm finds that $\{A, D\}$ shares the largest mutual information with Y on $\mathcal{L}_2$. Then, the DFL algorithm searches through $\Delta_2(A)$, . . ., $\Delta_{K-1}(A)$, however it always decides the search order of Δ_{i+1} (A) based on the calculation results of $\Delta_i(A)$. Finally, the DFL algorithm finds that the subset $\{A, C, D\}$ satisfies the requirement of Theorem 2.2.2, i.e., $I(\{A, C, D\}; Y) = H(Y)$, and will construct the function f for Y



{}
{A} {B} {C} {D}
{A,B} {A,C} {A,D} {B,C} {B,D} {C,D}
{A,B,C} {A,B,D} {A,C,D}* {B,C,D}
{A,B,C,D}

Figure 2.6: The search procedures of the DFL algorithm when learning $Y = (A \cdot C) + (A \cdot D)$. $\{A, C, D\}^*$ is the target combination. The combinations with a black dot under them are the subsets which share the largest mutual information with Y on their layers. Firstly, the DFL algorithm searches the first layer, then finds that $\{A\}$, with a black dot under it, shares the largest mutual information with Y among subsets on the first layer. Then, it continues to search $\Delta_1(A)$ on the second layer. Similarly, these calculations continue until the target combination $\{A, C, D\}$ is found on the third layer.

with these three attributes.

To determine the truth table, firstly, the B is deleted from training data set since it is a non-essential attribute. Then, the duplicate rows of $(\{A, C, D\}, Y)$ are removed from the training data set to obtain the final function f as the truth table of $(A \cdot C) + (A \cdot D)$ along with the counts for each instance of $(\{A, C, D\}, Y)$, as shown in Table 2.4. *This is the reason for which we name our algorithm as the Discrete Function Learning algorithm.*

If the DFL algorithm still does not find the target subset, which satisfies the requirement of Theorem 2.2.2, in Kth layer $\mathcal{L}_K$, the DFL algorithm will return to the first layer. Now, the first node on the $\mathcal{L}_1$ and all its $\Delta_1, \ldots, \Delta_{K-1}$ supersets have already been checked. In the following, the DFL algorithm continues to calculate the second node on the first layer

Table 2.4: The learned classifier f of the example to learn $Y = (A \cdot C) + (A \cdot D)$.

ACD	*Y*	Count	*ACD*	*Y*	Count
000	0	2	100	0	2
001	0	2	101	1	2
010	0	2	110	1	2
011	0	2	111	1	2

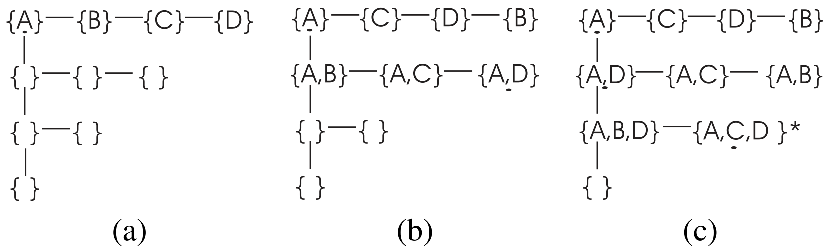
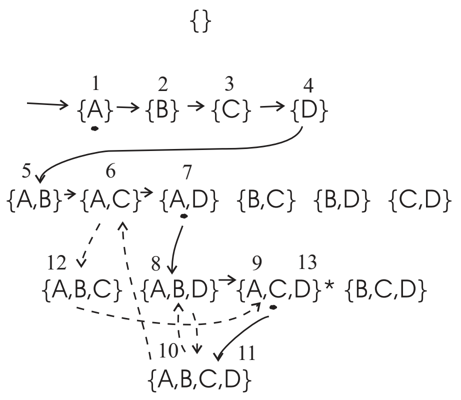


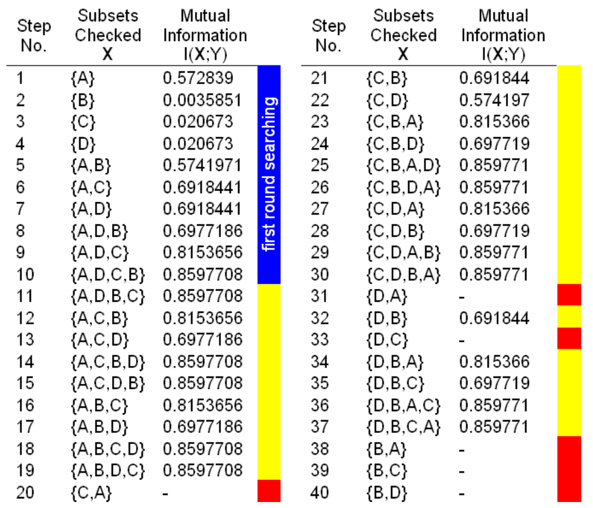
Figure 2.7: The $\Delta Tree$ when searching the EAs for $Y = (A \cdot C) + (A \cdot D)$. (a) after searching the first layer of Figure 2.6 but before the sort step in line 7 of Table 2.2. (b) when searching the second layer of Figure 2.6. The $\{A\}$, $\{C\}$ and $\{D\}$ which are included in the EAs of Y are listed before $\{B\}$ after the sort step in line 7 of Table 2.2. (c) when searching the third layer of Figure 2.6, $\{A, C, D\}^*$ is the target combination. Similar to part (b), the $\{A, C\}$ and $\{A, D\}$ are listed before $\{A, B\}$. When checking the combination $\{A, C, D\}$, the DFL algorithm finds that $\{A, C, D\}$ is the complete EAs for Y since $\{A, C, D\}$ satisfies the criterion of Theorem 2.2.2.

(and all its $\Delta_1, \ldots, \Delta_{K-1}$ supersets), the third one, and so on, until it reaches the end of $\mathcal{L}_1$.

We use the example in Figure 2.8 to illustrate the searching steps beyond the first round searching of the DFL algorithm. Note that the DFL algorithm is the same as the classical greedy forward selection algorithm [49] and uses the mutual information $I(\mathbf{U}; Y)$ as the greedy measure before it returns to the $(K-1)$th layer from Kth layer for the first time.



(a)



Step No.	Subsets Checked X	Mutual Information I(X;Y)	
1	{A}	0.572839	🟦 first round searching
2	{B}	0.0035851	🟦
3	{C}	0.020673	🟦
4	{D}	0.020673	🟦
5	{A,B}	0.5741971	🟦
6	{A,C}	0.6918441	🟦
7	{A,D}	0.6918441	🟦
8	{A,D,B}	0.6977186	🟦
9	{A,D,C}	0.8153656	🟦
10	{A,D,C,B}	0.8597708	🟦
11	{A,D,B,C}	0.8597708	🟨
12	{A,C,B}	0.8153656	🟨
13	{A,C,D}	0.6977186	🟨
14	{A,C,B,D}	0.8597708	🟨
15	{A,C,D,B}	0.8597708	🟨
16	{A,B,C}	0.8153656	🟨
17	{A,B,D}	0.6977186	🟨
18	{A,B,C,D}	0.8597708	🟨
19	{A,B,D,C}	0.8597708	🟨
20	{C,A}	-	🟥
21	{C,B}	0.691844	🟨
22	{C,D}	0.574197	🟨
23	{C,B,A}	0.815366	🟨
24	{C,B,D}	0.697719	🟨
25	{C,B,A,D}	0.859771	🟨
26	{C,B,D,A}	0.859771	🟨
27	{C,D,A}	0.815366	🟨
28	{C,D,B}	0.697719	🟨
29	{C,D,A,B}	0.859771	🟨
30	{C,D,B,A}	0.859771	🟨
31	{D,A}	-	🟥
32	{D,B}	0.691844	🟨
33	{D,C}	-	🟥
34	{D,B,A}	0.815366	🟨
35	{D,B,C}	0.697719	🟨
36	{D,B,A,C}	0.859771	🟨
37	{D,B,C,A}	0.859771	🟨
38	{B,A}	-	🟥
39	{B,C}	-	🟥
40	{B,D}	-	🟥

(b)

Figure 2.8: The exhaustive searching procedures of the DFL algorithm when learning $Y = (A \cdot C) + (A \cdot D)$. $\{A, C, D\}^*$ is the target combination. (a) The exhaustive searching after the first round searching. The numbers beside the subsets are the steps of the DFL algorithm in part (b). The solid edges represent the searching path in the *first round searching*, marked as blue region in part (b). The dashed edges represent the searching path beyond the first round searching (only partly shown for the sake of legibility), marked as yellow regions in the table below. (b) The exhaustive searching steps. Blue, yellow and red regions correspond to *first round searching*, exhaustive searching and the subsets, as well as their supersets, not checked after deploying the redundancy matrix to be introduced in Section 2.8.2.

We name the searching steps before this first return as the *first round searching* of the DFL algorithm. As shown in Figure 2.8 (a) and (b), this first return happens after step 10.

To produce the exhaustive searching, we add one noisy sample (1100,1) to the training data set in Table 2.3. Then, we keep the same settings of $K = 4$ and $\epsilon = 0$. As shown in Figure 2.8 (b), the mutual information $I(\mathbf{X}; Y)$ of all subsets is not equal to $H(Y) = 0.977$. Therefore, the DFL algorithm will exhaustively check all subsets and finally report "Fail to identify the model for Y (the classifier) when ϵ = 0".

In Figure 2.8 (a), the first round searching is show in solid edges and the subsets checked in each step are shown in the blue region of Figure 2.8 (b). In Figure 2.8 (a), the dashed edges represent the searching path beyond the first round searching (only partly shown for the sake of legibility), marked as yellow regions in Figure 2.8 (b). The red regions are the subsets, as well as their supersets, that will not be checked after deploying the redundancy matrix to be introduced in Section 2.8.2.

2.4.3 Complexity Analysis

First, we analyze the worst-case complexity of the DFL algorithm. As to be discussed in Section 2.8.1, the complexity to compute the mutual information $I(\mathbf{X};Y)$ is $O(N)$. For the example in Figure 2.6, $\{A,B\}$ will be visited twice from $\{A\}$ and $\{B\}$ in the worst case. $\{A,B,C\}$ will be visited from $\{A,B\}$, $\{A,C\}$ and $\{B,C\}$. Thus, $\{A,B,C\}$ will be checked for $3 \times 2 = 3!$ times in the worst case. In general, for a subset with K features, it will be checked for $K!$ times in the worst case. Hence, it takes $O((\binom{n}{1} + \binom{n}{2}2! + \ldots + \binom{n}{K}K!) \times N) = O(N \cdot n^K)$ to examine all subsets in $\mathcal{S}_K$. Another computation intensive step is the sort step in line 7 of Table 2.2. In $\mathcal{L}_1$, there is only one sort operation, which takes $O(n \log n)$ time. In $\mathcal{L}_2$, there would be n sort operations, which takes $O(n^2 \log n)$ time. Similarly, in $\mathcal{L}_K$, the sort operation will be executed for n^{K-1} times, which takes $O(n^K \log n)$ time. Therefore, the total complexity of the DFL algorithm is $O((N + \log n) \cdot n^K)$ in the worst case.

As described in Section 2.2.1, we use k to denote the actual cardinality of the EAs. After the EAs with k attributes are found in the subsets of cardinalities $\leq K$, the DFL algorithm will stop its searching. In our example, the K is 4, while the k is automatically determined as 3, since there are only 3 EAs for the example.

Then, we analyze the expected complexity of the DFL algorithm. Contributing to sort step in the line 7 of the subroutine, the algorithm makes the best choice on current layer of subsets. Since there are $(n-1)$ Δ_1 supersets for a given single element subset, $(n-2)$ Δ_1 supersets for a given two element subset, and so on. The DFL algorithm only considers $\sum_{i=0}^{k-1}(n-i) \approx k \cdot n$ subsets on the average. Thus, the expected time complexity of the DFL algorithm is approximately $O(k \cdot n \cdot (N + \log n))$, where $\log n$ is for sort step in line 7 of Table 2.2.

Next, we consider the space complexity of the DFL algorithm. To store the information needed in the search processes, the DFL algorithm uses two data structures. The first one is a linked list, which stores the value list of every variable. Therefore, the space complexity of the first data structure is $O(Nn)$. The second one is the $\Delta Tree$, which is a linked list of length K, and each node in the first dimension is itself a linked list. The $\Delta Tree$ for the example in Figure 2.6 is shown in Figure 2.7. The first node of this data structure is used to store the single element subsets. If the DFL algorithm is processing $\{X_i\}$ and its Δ supersets, the second node to the Kth node are used to store Δ_1 to Δ_{K-1} [3] supersets of $\{X_i\}$. If there are n variables, there would be $\sum_{i=0}^{K-1}(n-i) \approx Kn$ subsets in the $\Delta Tree$. To store the $\Delta Tree$, the space complexity would be $O(Kn)$, since only the indexes of the variables are stored for each subsets. Therefore, the total space complexity of the DFL algorithm is $O((K+N) \cdot n)$.

2.4.4 Correctness Analysis

The DFL algorithm is correct. Formally, we have

Theorem 2.4.1 *Let* $\mathbf{V} = \{X_1, \ldots, X_n\}$. *The DFL algorithm can find a consistent function* $Y = f(\mathbf{U})$ *of maximum indegree* K *with* $O((N + logn) \cdot n^K)$ *time in the worse case from*

[3]Except Δ_1 supersets, only a part of other $\Delta_i (i = 2, \ldots, K-1)$ supersets is stored in $\Delta Tree$.

$\mathbf{T} = \{(\mathbf{v}_i, y_i) : i = 1, 2, \ldots, N\}$.

The word "consistent" means that the function $Y = f(\mathbf{U})$ is consistent with the learning samples, i.e., $\forall \mathbf{u}_i, f(\mathbf{u}_i) = y_i$. Clearly, the original function is a consistent function. Note that Theorem 2.4.1 does not specify the requirement of the sample size. In Theorem 2.4.1, the $\mathbf{U}$ is not necessarily the input $\mathbf{X}$ in the original generation function $Y = f(\mathbf{X})$. We will discuss this issue in detail in Section 4.3.

However, if there are enough samples, the DFL algorithm will correctly find the original function in polynomial time. Formally, we have

Theorem 2.4.2 *Let* $\mathbf{V} = \{X_1, \ldots, X_n\}$*,* $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{X}$*,* $\mathbf{X}$ *and* $\mathbf{Z}$ *are independent. Given enough samples of* $\mathbf{V}$*. The DFL algorithm can find the original generation function* $Y = f(\mathbf{X})$ *of maximum indegree* K *with* $O((N + logn) \cdot n^K)$ *time in the worse case from* $\mathbf{T} = \{(\mathbf{v}_i, y_i) : i = 1, 2, \ldots, N\}$*.*

This is due the fact that when sample size is large enough, $I(\mathbf{Z}; Y) = 0, \forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$, based on Theorem 2.2.4. Then, the DFL algorithm will not choose the variables in $\mathbf{V} \backslash \mathbf{X}$ as EAs, given enough samples.

From Theorem 2.4.1 and 2.4.2, it can be seen that the DFL algorithm will converge to the original generation function, when the sample size increases. In other words, the DFL algorithm will produce a hypothesis, which will not change any more, after a certain number of samples are given.

2.5 The ϵ Value Method for Noisy Data Sets

In this section, we will first introduce the ϵ value method. Then, we discuss the relationship between the ϵ value method and the over-fitting problem. Finally, we show that the ϵ value method is useful to reduce the run time of the DFL algorithm.

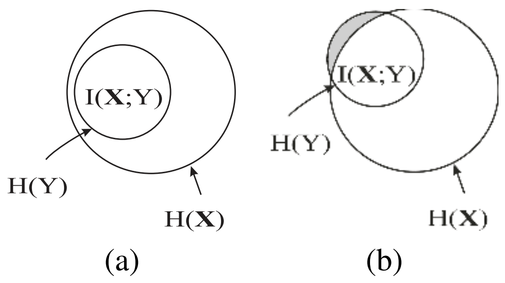


Figure 2.9: The Venn diagram of $H(\mathbf{X})$,$H(Y)$ and $I(\mathbf{X},Y)$, when $Y = f(\mathbf{X})$. (a) The noiseless case, where the mutual information between $\mathbf{X}$ and Y is the entropy of Y. (b) The noisy case, where the entropy of Y is not equal to the mutual information between $\mathbf{X}$ and Y strictly. The shaded region is resulted from the noises. The ϵ value method means that if the area of the shaded region is smaller than or equal to $\epsilon \times H(Y)$, then the DFL algorithm will stop the searching process, and build the function for Y with $\mathbf{X}$.

2.5.1 The ϵ Value Method

In Theorem 2.2.2, the exact functional relation demands the strict equality between the entropy of Y, $H(Y)$ and the mutual information of $\mathbf{X}$ and Y, $I(\mathbf{X};Y)$. However, this equality is often ruined by the noisy data, like microarray gene expression data. The noise changes the distribution of $\mathbf{X}$ or Y, therefore $H(\mathbf{X})$, $H(\mathbf{X},Y)$ and $H(Y)$ are changed due to the noise. From Equation 2.6, $I(\mathbf{X};Y)$ is changed as a consequence. In these cases, we have to relax the requirement to obtain a best estimated result. As shown in Figure 2.9, by defining a significant factor ϵ, if the difference between $I(\mathbf{X};Y)$ and $H(Y)$ is less than $\epsilon \times H(Y)$, then the DFL algorithm will stop the searching process, and build the classifier for Y with $\mathbf{X}$ at the significant level ϵ.

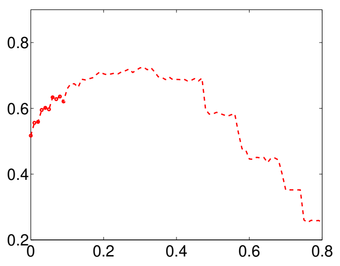
Because the $H(Y)$ may be quite different for various classification problems, it is not appropriate to use an absolute value, like ϵ, to stop the searching process or not. Therefore, a percentage of $H(Y)$ is used as the criterion to decide whether to stop the searching process or not. Such normalized form of $H(Y)$ is in accordance with the fundamental of statistics, like the z-value and t-value in classical statistics.

The main idea of the ϵ value method is to find a subset of attributes which captures not all the diversity of the Y, $H(Y)$, but the major part of it, i.e. $(1-\epsilon)\times H(Y)$, then to build functions with these attributes. The attributes in vectors, which have strong relations with Y, are expected to be selected as input variables of Y, i.e., the EAs of the models, in the ϵ value method.

2.5.2 The Relation with The Over-fitting Problem

The ϵ value method can help to avoid over-fitting of the training data sets. For a given noisy data set, the missing part of $H(Y)$ is determined, so there exists a threshold value of ϵ with which the DFL algorithm can find the correct input variables $\mathbf{X}$ of the generation function $Y = f(\mathbf{X})$. From Theorem 2.2.1, it is known that more variables tend to contain more information about the class attribute Y. On the other hand, from Figure 2.9, it can be seen that some part of $H(Y)$ is not captured by the input variables $\mathbf{X}$ due to the noise. Therefore, it is likely to include more than necessary number of feature as EAs, if we continue to add variables after the threshold value of ϵ. The unnecessary input variables often incur complex models and risks of over-fitting the training data sets. By introducing the ϵ value method, the DFL algorithm will stop the searching procedure when the missing part of $H(Y)$ is smaller than or equal to $\epsilon\times H(Y)$, and avoids the inclusion of unnecessary input variables.

An example is given in Figure 2.10, which is generated with the LED+17 data set [31] with 3000 samples. The LED+17 data set has 23 Boolean features, 7 relevant and 16 irrelevant. We randomly choose 2000 samples as the training data set and the remaining 1000 as testing data set. This LED+17 data set will be used later in Section 5.3. From Figure 2.10



0.8
0.6
0.4
0.2
0
0.2
0.4
0.6
0.8

2.5.3 The Relation with The Time Complexity

The ϵ value method is very helpful to avoid the exhaustive searching when dealing with noisy data sets. Because there is not subset that satisfies Theorem 2.2.2 in all subsets of $\mathbf{V}$ when the data sets are noisy. After introducing proper ϵ value, the DFL algorithm will just check the n subsets with one variable, and $n-1$ subsets with two variables, and so on. Thus, the DFL algorithm maintains its expected complexity of $O((k \cdot n \cdot (N + \log n))$. For example, as shown in Figure 2.8 (b), since the data set is noisy, the $I(\mathbf{X}; Y) = H(Y)$ cannot be satisfied with ϵ of 0. Thus, the DFL algorithm will exhaustively search all subsets of $\mathbf{V}$ and report "Fail to identify the model for Y (the classifier) when ϵ = 0". But when the ϵ value increases to 0.17, the DFL algorithm can correctly find the three input variables $\{A, C, D\}$ in the 9th step in Figure 2.8 (b), since $H(Y) - I(\{A, C, D\}; Y) = 0.977 - 0.815 = 0.162 < 0.17 \times H(Y) = 0.17 \times 0.977 = 0.166$. Thus, the complex exhaustive searching is avoided by introducing $\epsilon = 0.17$. For another example, in Figure 2.10 (c), it is shown that if proper ϵ value is chosen, the DFL algorithm can be significantly faster while achieves better prediction performance, in Figure 2.10 (a).

2.6 Selection of Parameters

We first discuss how to select the two parameters, the expected cardinality of the EAs K and the ϵ value, of the DFL algorithm in this section. Then, the balance between these two parameters is also discussed.

2.6.1 Selection of The Expected Cardinality K

We discuss the selection of the expected cardinality K in this section. Generally, if a data set has a large number of features, like several thousands, then K can be assigned to a small constant, like 20, since the models with large number of features will be very difficult to understand. If the number of features is small, then K can be directly specified to the number of features n.

Another usage of K is to control model complexity. If the number of features is more important than accuracy, then a predefined K can be set. Thus, the learned model will have fewer than or equal to K features.

The expected cardinality K can also be used to incorporate the prior knowledge about the number of relevant features. If we have the prior knowledge about the number of relevant features, then the K can be specified as the predetermined value.

For biological data sets, the introduction of K has its biological foundation. In learning qualitative models of GRNs, each gene is estimated on the average to interact with four to eight other genes [19]. In the cancer classification problems, only a small set of genes of the human genome are responsible for the tumor cell developmental pathway [126]. That is to say, it is sufficient to consider the combinations whose cardinalities are bounded by a small integer, K, in solving problems based on biological data sets.

2.6.2 Selection of ϵ value

For a given noisy data set, the missing part of $H(Y)$, as demonstrated in Figure 2.9, is determined, i.e., there exists a specific minimum ϵ value, ϵ_m, with which the DFL algorithm can find the original model. If the ϵ value is smaller than the ϵ_m, the DFL algorithm will not find the original model. Here, we will introduce two methods to efficiently find ϵ_m.

In the first method, the ϵ_m can be found automatically by a restricted learning process. To efficiently find the ϵ_m, we restrict the maximum number of the subsets to be checked to $K \times n$. A pre-defined scope of ϵ is specified in prior. If the DFL algorithm cannot find the model for a noisy data set with the specified minimum ϵ value, then the ϵ will be increased

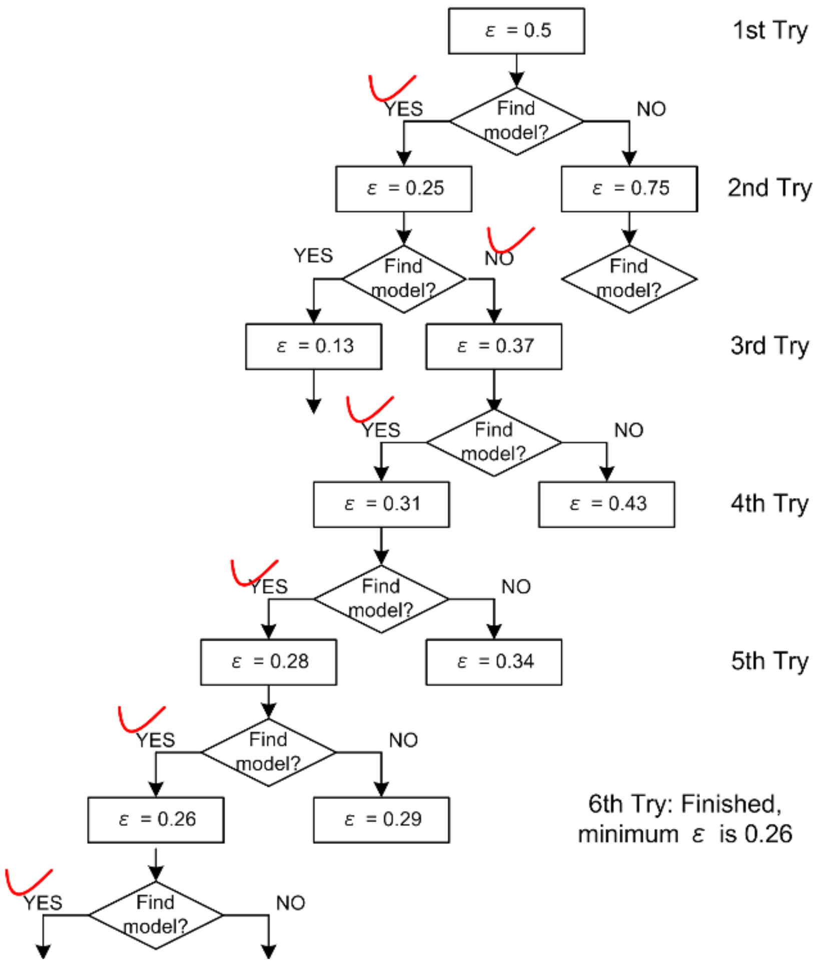


Figure 2.11: The manual binary search of minimum ϵ value. This figure is generated with the LED training data set in Table 5.1, with 2000 samples. The ticks indicate whether the DFL algorithm can find a model after a ϵ value is specified in each try. K is set to $n = 7$.

with a step of 0.01. The restricted learning will be performed, until the DFL algorithm finds a model with a threshold value of ϵ, i.e., the ϵ_m. Since only $K \times n$ subsets are checked, the time to find ϵ_m will be $O(K \cdot n)$.

In the second method, the ϵ_m can also be found with a manual binary search method. Since $\epsilon \in [0, 1)$, ϵ is specified to 0.5 in the first try. If the DFL algorithm finds a model with ϵ value of 0.5, then ϵ is specified to 0.25 in the second try. Otherwise, if the DFL algorithm cannot find a model with a long time, like 10 minutes, then the DFL algorithm can be stopped and ϵ is specified to 0.75 in the second try. The selection process is carried out until the ϵ_m value is found so that the DFL algorithm can find a model with it but cannot when $\epsilon = \epsilon_m - 0.01$. This selection process is also efficient. Since $\epsilon \in [0, 1)$, only 5 to 6 tries are needed to find the ϵ_m on the average.

As shown in Figure 2.11, we use the LED data set [31] with 10 percent noise to show the manual binary search procedure. There are 3000 samples in this data set, 2000 as training and 1000 as testing. This LED data set will also be used later in Section 5.3. For this example, in the first try, DFL algorithm finds a model for the training data set with ϵ of 0.5. Then, the DFL algorithm cannot find a model with the ϵ of 0.25 in the second try. Similarly, from the third to sixth tries, the DFL algorithm finds models with the specified ϵ values, 0.37, 0.31, 0.28 and 0.26. Since we have known in the second try that the DFL algorithm cannot find a model with ϵ of 0.25. Hence, 0.26 is the minimum ϵ value for this data set.

The restricted learning process can also be used to find optimal model in solving classification problems. To get optimal model, we change the ϵ value from 0 to the upper limit of the searching scope, like 0.8, with a step of 0.01. For each ϵ value, we train a model with the DFL algorithm, then validate its performance with cross validation or the testing data sets. The optimal model is the one which produces the best prediction performances. As demonstrated in Figure 2.10 (a), the optimal ϵ value, $\epsilon_{op} = 0.31$, is chosen from the training data set with a 10-fold cross validation. Then, this model also reaches its best performance on the testing data set, as demonstrated with the curve for the testing data set in Figure 2.10 (a).

2.6.3 Balance between K and ϵ

When the data sets are noisy and ϵ is set to a small value, then it is more advisable to choose a large K. Because it is possible that there are no good feature subsets with only a few features that can satisfy the small ϵ. For the example in Figure 2.11, if the K is set to 5 and ϵ to 0.26, then the DFL cannot find the model. But when K is set to 7, the DFL algorithm can find the model when $\epsilon = 0.26$. Similarly, if ϵ is large, then K can be adjusted to a relatively small value.

2.7 Prediction Methods

After the DFL algorithm obtains the classifiers as function tables of the pairs $(\mathbf{u}, y)$, or called as rules, the most reasonable way to use such function tables is to check the input values $\mathbf{u}$, and find the corresponding output values y. This is due to the fact that the DFL algorithm is based on Theorem 2.2.2. As demonstrated in Section 2.4.2, the learned model of the DFL algorithm is actually the generation function as a truth table or an estimation of it in the ϵ value method. Like the way in which people use truth tables, it is advisable to use a classification model as a truth table, or the estimation of it, with the 1NN algorithm. Therefore, we perform predictions in the space defined by the EAs, called the *EA space*, with the 1-Nearest-Neighbor algorithm [5] based on the Hamming distance defined as follows.

Definition 2.7.1 *Let $1(a, b)$ be an indicator function, which is 0 if and only if $a = b$, otherwise is 1. The Hamming distance between two arrays $\mathbf{A} = [a_1, \ldots, a_n]$ and $\mathbf{B} = [b_1, \ldots, b_n]$ is $Dist(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{n} 1(a_i, b_i)$.*

Note that the Hamming distance [88] is dedicated to binary arrays, however, we do not differentiate between binary or non-binary cases in this thesis. In the generation process

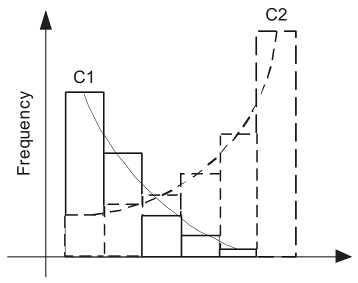


Figure 2.12: The noisy rules in the one-dimensional space. The rules below the two characters C1 and C2 are the genuine rules. Other rules are resulted from the noise in the data set. The vertical axis represents the frequencies of the rules in the training data sets. The rules are arranged according to their distance to the genuine rules. The solid and dashed curves are the distributions of rules for two class C1 and C2. In the real data sets, the frequencies of rules are represented by the histograms of solid and dashed lines.

of data samples, some features may change their values due to all kinds of reasons, like noise. We use the Hamming distance as a criterion to decide the class value of a new sample, since we believe that the rule with minimum Hamming distance to the EA values of a sample contains the maximum information of the sample. Thus, the class value of this rule is the best prediction for the sample.

In the prediction process, if a new sample is of same distance to several rules, we choose the rule with the biggest count value. The reason can be interpreted with the example shown in Figure 2.12. In Figure 2.12, it can be seen that a new sample in the region covered by two types of histograms can be of either classes. However, it is statistically more reasonable to believe the sample has the class value of a rule with higher frequency in the training data set. Hence, we call the 1NN algorithm used in our method as the weighted 1NN algorithm.

It is clear that the complexity of the weighted 1NN algorithm is mainly related to three parameters, the number of rules r in learned classifier f, the number of samples M in the testing data set and the actual cardinality of the EAs k. To compute the Hamming distance, it is sufficient to scan the two sequences with $O(k)$ time. Each sample in the testing data set is evaluated with respect to the r rules in the classifier, which takes $O(k \times r)$ steps. Thus, the total complexity of the weighted 1NN algorithm is $O(k \cdot r \cdot M)$.

Although there exists the probability that some instances of the EAs in the testing data set are not covered by the training data set, the 1NN algorithm still gives the most reasonable predictions for such samples.

For convenience, we will express the proposed classification method as the DFL algorithm hereafter when it does not result in misunderstanding.

2.8 Implementation Issues

In this section, we talk about two important issues in the implementation.

2.8.1 The Computation of Mutual Information $I(\mathbf{U}; Y)$

We use Equation 2.6 to compute $I(\mathbf{U}; Y)$. The $H(Y)$ does not change in the searching process of the DFL algorithm. To compute $H(\mathbf{U})$ and $H(\mathbf{U}, Y)$, we need to estimate the joint distribution of $\mathbf{U}$ and $(\mathbf{U}, Y)$, which can be estimated from the input table $\mathbf{T}$. The DFL algorithm will construct a matrix containing the values of $\mathbf{U}$. Then, it scans the matrix and finds the frequencies of different instances of $\mathbf{U}$, which are stored in a frequency table with a linked list. The size of the frequency table grows exponentially with the number of variables in $\mathbf{U}$, but will not exceed N. Next, the DFL algorithm will obtain the estimation of $H(\mathbf{U})$ with Equation 2.1.1. For each instance of $\mathbf{U}$ in $\mathbf{T}$, we need to update its frequency in the frequency table, which takes $O(min(||\mathbf{U}||, N)$ steps. The total complexity to compute $H(\mathbf{U})$ is $O(N \cdot min(||\mathbf{U}||, N))$. The computation of $H(\mathbf{U}; Y)$ is similar to that of $H(\mathbf{U})$.

Hence, if $\mathbf{U}$ only contains a few variables, it will need approximate $O(N)$ steps to compute $I(\mathbf{U};Y)$, since $||\mathbf{U}||$ is small. While $|\mathbf{U}|$ is large, the computation of $I(\mathbf{U};Y)$ tends to take $O(N^2)$ steps in the worst case.

However, the complexity for computing $I(\mathbf{U};Y)$ can be improved by storing the frequencies of different instances of $\mathbf{U}$ and $\{\mathbf{U},Y\}$ in a hash table. For each instance of $\mathbf{U}$ in $\mathbf{T}$, it only takes $O(1)$ time to update its frequency in the hash table [49]. Hence, the total complexity to compute $H(\mathbf{U})$ is $O(N)$. The computation of $H(\mathbf{U};Y)$ is similar to that of $H(\mathbf{U})$. Therefore, it will only need approximate $O(N)$ steps to compute $I(\mathbf{U};Y)$. An important issue to notice is the proper setting of the initial capacity of the hash table, since too large value brings waste but too small value may incur the dynamic increasing the capacity and reorganizing of the hash table, which is time-consuming.

In summary, if $|\mathbf{U}|$ and N are large at the same time and there are enough memory space available, it is more advisable to use hash tables for calculating $I(\mathbf{U};Y)$. While $|\mathbf{U}|$ or N is small and memory space is limited, it is better to use linked lists or arrays to compute $I(\mathbf{U};Y)$.

2.8.2 Redundancy Matrix

The subroutine in Table 2.2 is recursive, which will introduce some redundant computation when the DFL algorithm exhaustively searches the searching space $\mathcal{S}_K$. For instance, the A,B is a common Δ_1 supersets of $\{A\}$ and $\{B\}$. Hence, if the DFL algorithm will check $\{A,B\}$ twice in the worst case. Generally, for a subset with K variables, since the permutation number of variables in the subset is $K!$, hence this subset will be checked for $K!$ times. Thus, the worst time complexity of the DFL algorithm is $O((n + \binom{n}{2}\cdot 2! + \ldots + \binom{n}{K}\cdot K!)N + n^K \log n) = O((N + \log n)\cdot n^K)$.

However, this redundant computation can be relieved by storing the information of whether a subset has been checked with a Boolean type matrix. Let us consider the subsets with 2 variables. We introduce an n by n matrix called *redundancy matrix*, $\mathrm{boolean}\ \mathbb{R}(n \times n)$. After a subset $\{X_i, X_j\}$ and its supersets have been checked, $\mathbb{R}[i][j]$ is assigned as true. Later, when the DFL algorithm is trying $\{X_j, X_i\}$, it will first check whether $\mathbb{R}[i][j]$ or $\mathbb{R}[j][i]$ is true. If yes, it will examine next subset. By doing so, the original worst time complexity becomes $O((n+\frac{1}{2}[\binom{n}{2}\cdot 2!+\ldots+\binom{n}{K}\cdot K!])N+n^K \log n) = O((N+\log n)\cdot n^K)$. Although, this alleviated worst time complexity is in the same order as the original one, but it saves about half of the run time. The space complexity of $\mathbb{R}$ is $O(n^2)$. However, the type of $\mathbb{R}[\cdot][\cdot]$ is $\mathrm{boolean}$, so $\mathbb{R}$ will cost very limited memory space. In addition, if run time is more critical and the memory space is sufficient, higher dimensional matrices can be introduced to further reduce the run time of the DFL algorithm.

For instance, as shown in Figure 2.8, after introducing the redundancy matrix, the exhaustive searching of the DFL algorithm will take $n+1/2*[\binom{n}{2}2!+\binom{n}{3}3!+...+\binom{n}{K}K!] = 4+1/2*[\binom{4}{2}*2+\binom{4}{3}*3!+\binom{4}{1}4!] = 4+1/2*(6*2+4*6+1*24) = 4+30 = 34$ steps, which is in the order of $O(n^4) = O(4^4)$ but much smaller than 4^4. As shown in Figure 2.8 (b), there are totally 40 steps. But six of them marked as red regions, as well as their supersets, are not computed by checking the redundancy matrix.

To clearly show the implementation of the *redundancy matrix* $\mathbb{R}$, an extended version of the main steps of the DFL algorithm is provided at Table E.1.

2.9 Conclusions

In this chapter, we propose the ILA as a new approach to the computational learning theory. The ILA is based on information theory, incorporating some ideas from graphical models. In ILA, learning is interpreted and regarded as a procedure to acquire information about the considering concept.

We propose the DFL algorithm to implement the ILA. The DFL algorithm is flexible in learning functions with restricted or unrestricted number of inputs by introducing a parameter called expected cardinality K to control the cardinality of the EAs. We also discuss the selection of K and its underlying biological reasons for biological data sets.

We also propose the ϵ value to deal with noisy data sets. In the ϵ value method, the strong feature collectives which provide major part of the diversity, i.e., the entropy, of the concept $H(Y)$, are chosen to estimate the original models. We discuss relation between the ϵ value method and the over-fitting problem. We display that the ϵ value method is useful to reduce the run time of the DFL algorithm. We also discuss the selection of ϵ value in dealing with noisy data sets.

The searching method of the DFL algorithm is different from the classical greedy algorithm [49]. The merit of our searching method lies in that it is greedy in the first round of searching as shown in Figure 2.6. But the DFL algorithm still guarantees to check all the subsets in $\mathcal{S}_K$, as shown in Figure 2.8. Although it takes more time in its worst complexity, the DFL algorithm can still find the correct models in polynomial time as proved in Theorem 2.4.2. This searching scheme can also be used for solving many other combinatorial optimization problems.

To fulfill the prediction tasks in classification problems, we propose to use the weighted 1NN algorithm to perform predictions after the DFL algorithm obtains the classification model as a truth table or an estimation of it.

Two important issues in implementation of the DFL algorithm, to compute $I(\mathbf{U};Y)$ and to introduce the redundancy matrix $\mathbb{R}$, are also discussed.

Chapter 3

Learning Qualitative Models of Gene Regulatory Networks

ONE of the fundamental biological problems of 21st century is to understand how cellular phenomena arise from the interactions of genes and proteins. This chapter is dedicated to this problem from a reverse engineering approach.

Currently, still a little is known about the complex networks which are responsible for different expression patterns within cells. However, we have more and more data about gene regulatory networks with the availability of high-throughput technologies, such as DNA microarray [145, 146], which introduces an ambitious challenge to identify the original Gene Regulatory Networks (GRNs) from these data with a reverse engineering approach. In this chapter, we will apply the DFL algorithm to learning multi-value qualitative models of GRNs. In next chapter, we will use the DFL algorithm to learn Boolean networks, which is a special type of qualitative models of GRNs.

This chapter is organized as follows. First, in Section 3.1, we will briefly introduce GRNs, the motivation of researching on GRNs, and the approaches to deciphering GRNs.

Then, in Section 3.2, we describe the qualitative models of GRNs. In Section 3.3, we will learn a qualitative model of GRN with the DFL algorithm. In Section 3.4, we present the results of applying the DFL algorithm to the yeast cell cycle gene expression data sets. When validated with published literature, many of the regulatory relations found by the DFL algorithm are biologically significant. Section 3.5 presents a summary of this chapter.

3.1 A Brief Introduction to Gene Regulatory Networks

First, we briefly introduce the microarray technology [145,146]. Next, we show how to use the DFL algorithm to analyze the microarray gene expression data sets. Then, we discuss three basic questions in the introduction to the GRNs: *what are GRNs? why we research on GRNs? how to obtain GRNs?* and *what are the challenges?*

3.1.1 Introduction to Microarray Technology

The microarray technology was first introduced by Schena *et al.* [146] to measure the expression (concentration) level of messenger RNA in the cells. As shown in Figure 3.1, in an array experiment, many gene-specific polynucleotides derived from the $3'$ end of RNA transcripts are individually arrayed on a single matrix. This matrix is then simultaneously probed with fluorescently tagged cDNA representations of total RNA pools from test and reference cells, allowing one to determine the relative amount of transcript present in the pool by the type of fluorescent signals generated. Relative message abundance is inherently based on a direct comparison between a ‘test’ cell state and a ‘reference’ cell state; an internal control is thus provided for each measurement.

Microarray gene expression profiles have widely used to detect a specific biological pathway or process, like the cell cycle of yeast by Spellman *et al.* [158], to investigate

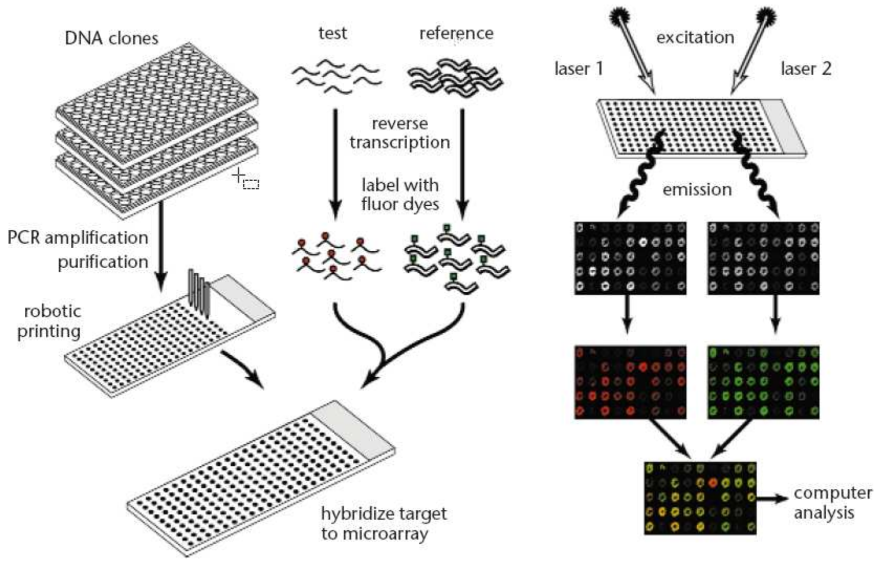


Figure 3.1: Overview of cDNA microarray technology. Templates for genes of interest are obtained and amplified by Polymerase Chain Reaction (PCR). Following purification and quality control, aliquots ($\sim$5 nl) are printed on coated glass microscope slides using a computer-controlled, high-speed robot. Total RNA from both the test and reference sample is fluorescently labelled with either Cye3- or Cye5-dUTP using a single round of reverse transcription. The fluorescent targets are pooled and allowed to hybridize under stringent conditions to the clones on the array. Laser excitation of the incorporated targets yields an emission with a characteristic spectra, which is measured using a scanning confocal laser microscope. Monochrome images from the scanner are imported into software in which the images are pseudo-colored and merged. Information about the clones, including gene name, clone identifier, intensity values, intensity ratios, normalization constant and confidence intervals, is attached to each target. Data from a single hybridization experiment is viewed as a normalized ratio (that is, Cye3/Cye5) in which significant deviations from 1 (no change) are indicative of increased ($>$1) or decreased ($<$1) levels of gene expression relative to the reference sample. In addition, data from multiple experiments can be examined using any number of data mining tools. (Courtesy of Duggan *et al.* [64])

differences between the tumor cell lines and non-tumor cell lines, like the colon tumor detection by Alon *et al.* [12], and to investigate differences between the tumor subtypes, like leukemia subtype detection by Golub *et al.* [81]. The first application often consists of some time-series experiments which are used to detect the change of gene expression levels in the biological pathway. These expression profiles are often named as time specific gene expression profiles. In Section 3.4, we will use time specific gene expression profiles of yeast cell cycle to infer GRN models. In comparison, the experiments in the second and third application are often tissue or spatial specific in the sense that the samples are from different tissue, like a tumor or a normal tissue (used as control). In Chapter 5, we will investigate the classification problems based on spatial specific gene expression profiles.

3.1.2 Analyzing Gene Expression Profiles with The DFL Algorithm

Let us see a schematic view of using the DFL algorithm to analyze microarray gene expression profiles in Figure 5.3, where the heat-maps are prepared with the GeneCluster 2.0 [141]. The DFL algorithm has also been used to perform cancer classification based on proteomic profiles [188] and microRNA gene expression profiles [190, 192]. These applications are similar to the application of cancer classification based on microarray gene expression profiles, as shown in Figure 5.3 (b), although not shown here.

The application for learning qualitative models of GRNs is shown in Figure 5.3 (a), where the gene expression profiles is related to yeast cell cycle [44]. In Figure 5.3 (a), it is shown that the data is discretized with unsupervised discretization methods [62], then rearranged to transition tables. Finally, the DFL algorithm is applied to transition pairs to obtain the qualitative models of GRNs. The network shown in Figure 5.3 (a) is a combined result for several sets of parameters, which will be discussed in Section 3.4.

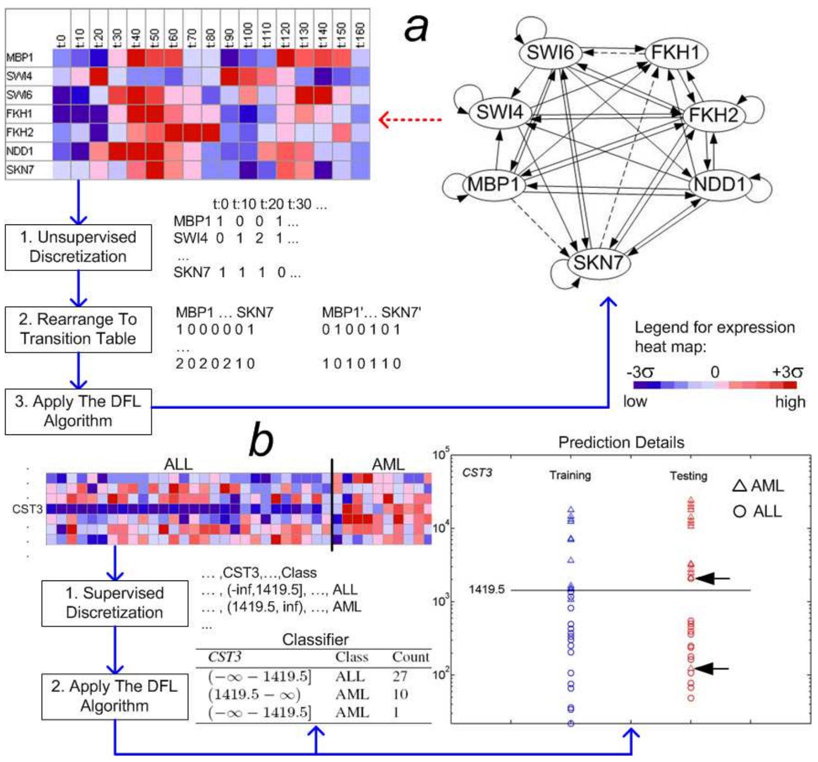

a
b
1. Unsupervised Discretization
2. Rearrange To Transition Table
3. Apply The DFL Algorithm
Legend for expression heat map:
low
high
ALL
AML
CST3
1. Supervised Discretization
2. Apply The DFL Algorithm
Prediction Details
Training
Testing
AML
ALL
1419.5

Figure 3.2: A schematic view of using the Discrete Function Learning algorithm to analyze microarray gene expression profiles. The blue arrows are the processes to use the DFL algorithm. (a) The DFL algorithm is used to learn qualitative models of GRNs from time-series gene expression profiles. The red dashed arrows means that the time-series gene expression data sets are assumed to be generated by the GRN model. The solid and dashed edges in the GRN model represent verified and putative regulatory relations respectively [184]. (b) The The DFL algorithm is used to perform tumor classification based on tissue-specific gene expression profiles. The black line in the prediction details is the cutting point of the *CST3* gene determined in the discretization process. In the testing data set, the two samples pointed by arrows are the incorrect predictions made by the classifier [189].

The application of classification is demonstrated in Figure 5.3 (b), where the gene expression profile is the leukemia subtype data [81]. The features X_is are the expression levels of genes. In Figure 5.3 (b), it is shown that the data is first discretized with supervised discretization methods [67], which has been implemented in the *Weka* software [69]. The DFL algorithm obtains the classifier shown in the central bottom part of Figure 5.3 (b) with $K = 20$ and $\epsilon = 0.29$. The ϵ value is automatically found with the restricted learning method introduced in Section 2.6.2. In prediction process, the obtained classifier makes 2 prediction errors in the independent testing data set. More details of this example are available in Section 5.4 and [189].

The application of feature selection is also demonstrated in Figure 5.3 (b). The DFL algorithm chooses the *CST3* gene to build the classifier, since the *CST3* gene captures the major diversity of the class attribute. We apply the C4.5 algorithm [140], the Naive Bayes (NB) algorithm [106], the 1NN and k-Nearest-Neighbors (kNN) algorithm [5] and the Support Vector Machines (SVM) algorithm [137] implemented in the *Weka* software to the disrectized data sets with all features and with only the *CST3* gene. The C4.5, NB, 1NN, kNN ($k = 5$) and SVM algorithms make 3, 5, 8, 6 and 6 prediction errors respectively on the data sets with all features [189], but only 2 errors on the data sets with the *CST3* gene, which demonstrates the discriminatory power of the feature subset chosen by the DFL algorithm. The details of this example are available in Section 6.4.

3.1.3 What Are Gene Regulatory Networks

To clarify GRNs, we should first of all know what genes are, i.e., the components of GRNs. In this section, we first discuss some basic biological background knowledge about genes. Then, we introduce how these genes are regulated to produce their functional products. Finally, we describe GRNs which are the interaction networks of genes.

Genes: The Basic Functional Component of Gene Regulatory Networks

The genome is universally defined as the total DNA content of a haploid cell or half the DNA content of a diploid cell. In cells, the DNA molecules are often organized as chromosomes. A gene is defined in molecular terms as a complete chromosomal segment responsible for making a functional product [157]. Thus, from this point, the genome also could be seen as the entire collection of genes encoded by a particular organism.

Gene Regulations

The expressions of genes are regulated by many factors. To start the transcription, the RNA polymerase II needs to be bound to the upstream regions of genes. The Transcription Factors (TFs) control the binding of RNA polymerase II. Some TFs are called transcriptional activators, who impose specificity on polymerase by binding the polymerase to the DNA. Other TFs, called repressors, work in a variety of ways, the simplest of which is to block the binding of polymerase to the gene. The polymerase can work on a wide array of genes (each a potential substrate), and the choice of substrate is regulated by the TFs. In the bacterium *Escherichia coli*, for example, the RNA polymerase transcribes some 3000 genes: certain genes are transcribed only at certain times, often as determined by signals received by the cell from its surroundings. In eukaryotes, similar RNA polymerases face the same regulatory task but on a larger scale. This example of substrate choice is called *regulation of gene expression* or, more casually, *gene regulation* [139]. Let see an example of *gene regulation* as shown in Figure 3.3.

In the bacterium *Escherichia coli*, the enzyme β-galactosidase, the product of the *lacZ* gene, cleaves lactose, the first step in metabolism of that sugar, as shown in Figure 3.3. The gene is transcribed if, and only if, lactose is present in the medium. But that physiological signal is almost overridden by the simultaneous presence of glucose, a more efficient energy

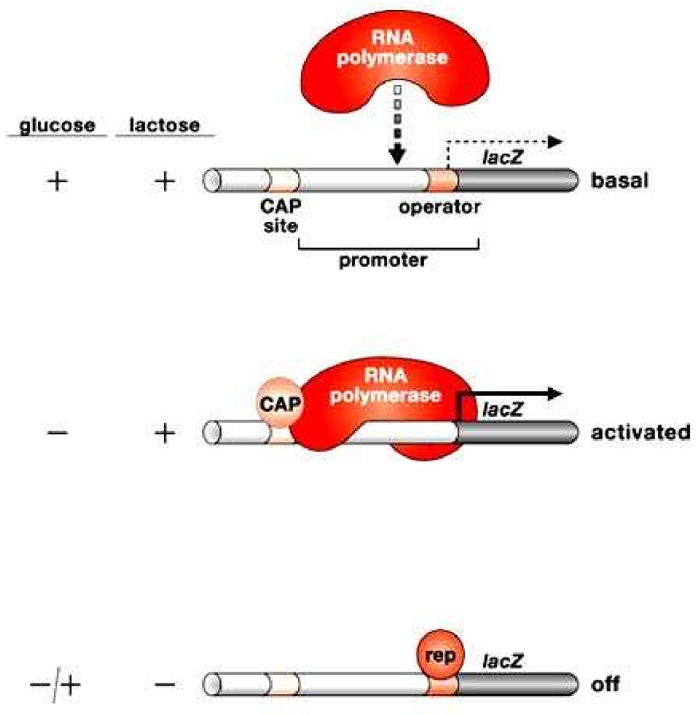


Figure 3.3: Three states of the *lac* genes. When bound to the operator, repressor (rep) excludes polymerase whether or not activate adenylyl Cylclase Associated Protein (CAP) is present. In the upper graph, We have a constitutively active enzyme (RNA polymerase) that, alone, works with a certain frequency. The activator "CAP" increases this frequency by recruiting the enzyme to the gene (middle), and the repressor "rep" decreases the frequency by excluding the enzyme (bottom). (Courtesy of Ptashne and Gann, [139])

source that lactose. Only after having exhausted the supply of glucose does the bacterium fully turn on expression of *lacZ*.

The effects of these two signals (lactose and glucose) are mediated by two DNA-binding regulatory proteins, each of which senses one of the sugars. Lac repressor "rep" binds the operator only in the absence of lactose, and adenylyl Cylclase Associated Protein (CAP), an activator, binds DNA only in the absence of glucose.

Gene Regulatory Networks

The TFs are essentially themselves proteins, i.e., the products of expression of other genes. In this way a product of one gene can influence the expression of other genes, thus it forms a network of gene regulations, gene regulatory networks [53], within a cell.

Every regulatory module contained in the genome receives multiple disparate inputs and processes them in ways that can be mathematically represented as combinations of logic functions (e.g., “and” functions, “switch functions”, “or” functions) [54]. Then, different regulatory modules are waved into complex GRNs, which give specific outputs, i.e., different gene expression patterns (like in developmental process), depending on their inputs, i.e., current status of the cell. Hence, the architecture of GRN is fundamental for both explaining and predicting developmental phenomenology [55, 109].

GRNs are the on-off switches and rheostats of a cell operating at the gene level. Gene regulatory networks govern which genes are expressed in a cell at any given time, how much product is made from each one, and the cell’s response to diverse environmental cues and intracellular signals [71].

As shown in Figure 3.4, a regulatory network can be viewed as a cellular input-output device. At minimum, a gene regulatory network typically contains the following components: (1) an input signal reception and transduction system that mediates intra and extracellular cues (left box; often, more than one signal impinges on a given target gene); (2) a “core component” complex composed of transacting regulatory proteins and cognate *cis*-acting DNA sequences (circle; functionally similar components may be associated with multiple target genes, resulting in similar gene-expression patterns); and (3) primary molecular outputs from target genes, which are RNA and protein (box to right of circle). The net effects are changes in cell phenotype and function (right box). Direct and indirect

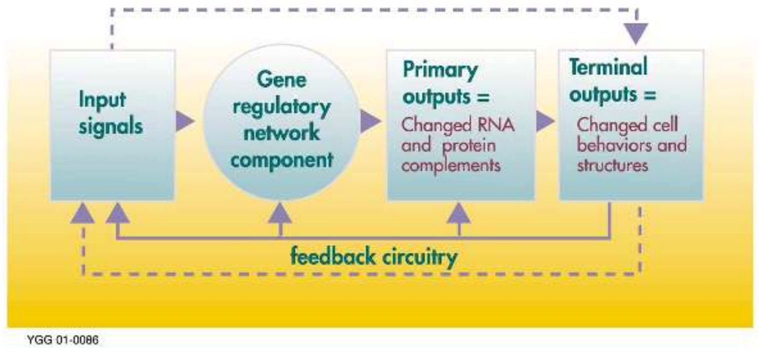

Input signals
Gene regulatory network component
Primary outputs = Changed RNA and protein complements
Terminal outputs = Changed cell behaviors and structures
feedback circuitry
YGG 01-0086

Figure 3.4: Gross anatomy of a minimal gene regulatory network (GRN) embedded in a regulatory network. (Courtesy of U.S. Department of Energy Genomes to Life Program, http://www.doegenomestolife.org.)

feedbacks typically are important. More realistic networks often feature multiple tiers of regulation, with first-tier gene products regulating expression of another group of genes, and so on. Beyond GRN boundaries are signaling responses and feedbacks, such as those that drive bacterial chemotaxis, which do not involve any regulation of gene expression but instead act directly on proteins and protein machine assemblies (dashed arrows). Some regulatory networks have no embedded GRN component.

3.1.4 Why We Research on Gene Regulatory Networks

GRNs are critical for us to understand many fundamental phenomena of biological processes within a cell.

First, GRNs are critical for understanding the developmental process from a zygote to a mature human body, as well as those of other species. One of the most fundamental challenges of 21st century biological research is to decipher the complex GRNs responsible for embryonic development [85]. GRNs regulate the expression of thousands of genes in any given developmental process [54]. They are essentially hardwired genomic regulatory codes, the role of which is to specify the sets of genes that must be expressed in specific spatial and temporal patterns [54].

Second, GRNs help to explain the discrepancy between the large difference between species whose genome sizes are however differing less severely. One such basic question has emerged from the *Human Genome Project*: How can a multicellular organism as complex as a human, with all its cell and tissue types and functions, use only 2 or 3 times as many as genes (about 30,000) as the simple worm and 5 to 10 times as many as single-cell microbe? Much of the answer may be in the regulatory network architecture and complexity [71]. Given the remarkable commonality of protein families in the animal kingdom, we must conclude that the morphological differences between animal species arise primarily through differential regulation of genes and their products, and that the information for this differential regulation must be encoded in the inherited DNA [32].

Third, a global gene network is assumed to be useful for disease diagnosis, disease treatment, and drug discovery as well as contributing tremendously to the fundamental understanding of biological processes. The gene may be expressed with an interesting temporal or spatial pattern during the development of a cell or organism, which suggests that the gene's TFs play an important developmental role. Alternatively, aberrant regulation of the gene may contribute to a particular disease, or the gene may be specifically expressed in a cell type associated with a disease. In these instances, the TFs may contribute to an understanding of disease pathogenesis or may provide targets for therapeutic intervention [38].

For these important roles of GRNs, the Genomes to Life program of the US Department of Energy has placed characterizing GRNs as its second goal [72]. After the *Human*

Genome Project, the next step, even a more difficult one, is to annotate the human genome and to find the complex the GRNs.

3.1.5 How to Obtain Gene Regulatory Networks

There are mainly two ways to obtain the Gene Regulatory Networks. One possible way, as demonstrated by Lee *et al.* [108], Simon *et al.* [156], Yuh *et al.* [179, 180] and Davidson *et al.* [56], is the classical experimental method. The regulatory network architecture is determined by experimental perturbation followed by measured of the effects on function of individual genes. However, the GRN architecture can be authenticated only by experimental molecular biology in which the functional meaning of given regulatory sequences is directly determined [54].

The other way is the reverse engineering method, which reconstructs the original GRNs from the data of the GRNs, i.e., the gene expression data and other biological data. If the expression of gene A is regulated by proteins B and C, then A's expression level is a function of the joint activity levels of B and C [73]. By mapping the output of each gene to the inputs of other genes, it is possible to reverse engineer development circuits and even whole gene network [93]. This kind of methods always employ *Machine Learning* methods to learn the underlying models from a vast amount of data. For instance, Friedman *et al.* [75] reconstructed a GRN of yeast *Saccharomyces cerevisiae*, which is related to the cell cycle, under the framework of Bayesian networks.

3.1.6 What Are the Challenges for Reconstructing Gene Regulatory Networks?

The goal of reconstructing Gene Regulatory Networks is a daunting task even with the technological developments nowadays. Here, we will talk about two major challenges, the complexity of biological systems and the scarceness of data sets, when constructing GRNs with the reverse engineering approach.

The Biological Systems Are Complex

Biological systems, through evolution, have achieved levels of intricacy and subtlety that dwarf the complexities of the 21st century's most sophisticated engineering feats [71].

First, the number of genes are large. Computational models of GRNs are required for analysis, for experimental manipulation and, most fundamentally, for comprehension of how GRNs work. The number of genes within a genome is very large, e.g., there are approximate 6000 genes within a yeast genome. It is beyond the human capacity to analyze the 6000 component simultaneously, which has made it necessary to investigate GRNs with the help of computational models.

Second, genes are combinatorially controlled by the TFs. In the combinatorial control of transcription initiation processes, different combinations of ubiquitous and cell-type-specific regulatory proteins are used to turn genes on and off in different regulatory contexts [34]. For example, the cis-regulatory region of the *Endo16* gene of sea urchin embryo consists of 2300 base pairs and 55 protein binding sites [179]. In different context, different combinations of regulators bind to the cis-regulatory region, thus, give rise to different expression levels of the *Endo16* gene.

The third kind of complexities of biological systems are coming from the temporal and spatial expression patterns in the development process. Different genes are expressed in different developmental stages. For example, the *EMF1* gene of Arabidopsis thaliana is a flowering repressing element, which is only active in the early stage of flower development [175]. In high metazoa, the expression of genes are also spatially different, for instance, the expression of the *Endo16* gene in sea urchin embryo [179]. Another example of spatial patterns of gene expression is that gap genes of Drosophila (such as *Gt*, *Hb*, *Kr*, etc.) are expressed at different level along the anterior-posterior axis of the embryo [53, 139]. In some cases, the temporal and spatial patterns are shown simultaneously. For example, Davidson *et al.* [56] summarized a comprehensive GRN, which have both temporal and spacial properties, for the specification of endoderm and mesoderm in the sea urchin embryo.

The Data Sets Are Scarce

Second, the data sets needed to reconstruct GRNs are not sufficient but scarce.

We first talk about the gene expression data sets. As discussed in the prior section, the number of genes are large, thus, many experiments are needed to capture the different expression patterns of these large number of genes. To correctly infer the regulation of a single gene, we need to observe the expression of the gene under many different combinations of expression levels of its regulatory inputs [60]. The spatial and temporal expression of different genes makes it necessary to obtain expression data sets of them under many of these spatial and temporal conditions. The integration and exchange of expression data from different experiments and different technological platforms are difficult, since it lacks standards in the microarray fields [33]. Ball et al. [20] reported one of the endeavors to set up a standard for the microarray data. But there is still much work to overcome the difficulty of exchanging microarray data sets of different platforms.

Beside gene expression data sets, other data sets, such as the *cis*-regulatory motif [93, 109], genome-wide location data sets [21, 90] and literature, are also available. How to combine different kinds of data sets to construct more reliable models is another challenge for reconstructing GRNs.

How We Handle These Challenges in Our Method?

To simulate gene networks with the limited data sets, we use several strategies in our method. First, we simplify the expression values as discrete variables, which makes large-scale simulation become feasible. Second, we only use the microarray gene expression data sets, although the gene expression level is controlled by many other factors. Third, we only consider the expression profiles of a specific biological process. In next section, we will describe how to use our method to learn qualitative models of GRNs, which is built from microarray gene expression profiles based on these three simplifications.

3.2 Methods

In this section, we will first briefly introduce the qualitative models of GRNs. Next, we formally define the problem of learning GRN models from state transition pairs and introduce the modified version of the DFL algorithm for solving this problem.

3.2.1 Qualitative Models of Gene Regulatory Networks

Due to the fact that very little data is available about the quantitative values of the concentrations of messenger RNA molecules and the strength of interactions between proteins and DNA, the traditional methods to simulate dynamic systems, like ordinary differential equations, cannot be applied to biological system easily. Therefore, qualitative models, like Generalized Logical Formalism (GLF) [160, 161] and Piecewise Linear Differential Equation (PLDE) [79, 125], are introduced to meet this problem. In qualitative models, the architecture of the GRNs are represented with regulatory function related to each gene, and the strength of the regulation impressed by the TFs is represented by discrete values. The purpose of using qualitative models is to understand the GRNs at the precision of enough expression levels of different genes that are needed to interpret the biological processes, as shown in [124, 143, 144].

However, it is not easy to build qualitative models of GRNs. Currently, almost all GLF and PLDE models are built by collecting evidence from literature, as shown in [13, 57, 58, 78, 124, 143, 144]. The manual extraction of knowledge from literature clearly hinders the applicability of these models on a large scale. With the recent development of microarray technology [59, 158], the expression levels of thousands of genes can simultaneously be obtained at discrete time points. It is a worthy effort to make use of these data to accelerate the building of qualitative models of GRNs.

Our aim is to learn qualitative models of GRNs from discretized microarray gene expression data. The qualitative models of GRNs are a set of discrete functions which depict the regulatory relations between genes under consideration. In our method, the expression data are assumed to be the products of these functions. Then, we use a reverse engineering method based on information theory to find these functions from gene expression data.

In qualitative models of GRNs, the genes are represented by a set of discrete variables, $\mathbf{V} = \{X_1, \ldots, X_n\}$. In GRNs, the expression level of a gene X at time step $t + 1$ is controlled by the expression levels of its regulatory genes, which encode the TFs of the gene X, at time step t. Hence, in qualitative models of GRNs, the genes at the same time step are assumed to be independent of each other, which is a standard assumption in learning GRNs under the qualitative models, as assumed in [6, 7, 9, 105, 113, 184]. In other words, the messenger RNAs of different genes produced at the same time cannot interact with each other. Formally, $\forall 1 \leq i, j \leq n$, $X_i(t)$, $X_j(t)$ are independent. And the regulatory relationships between the genes are expressed by discrete functions related to each variables. Formally, a GRN $G(\mathbf{V}, \mathbf{F})$ with indegree k (the number of inputs) consists of a set $\mathbf{V} = \{X_1, \ldots, X_n\}$ of nodes representing genes and a set $\mathbf{F} = \{f_1, \ldots, f_n\}$ of discrete functions, where a discrete function $f_i(X_{i1}, \ldots, X_{ik})$ with inputs from specified nodes X_{i1}, . . ., X_{ik} at time step t is assigned to the node X_i at time step $t+1$ [9, 13, 57, 58, 78, 105, 113, 124, 143, 184], as shown in the following equation

$$X_i(t+1) = f_i(X_{i1}(t), \ldots, X_{ik}(t)), \tag{3.1}$$

where $1 \leq i \leq n$. We call the inputs of f_i as the parent nodes of $X_i(t+1)$, and let $\mathbf{Pa}(X_i(t+1)) = \{X_{i1}(t), \ldots, X_{ik}(t)\}$.

The state of the GRN is expressed by the state vector of its nodes. We use $\mathbf{v}(t) = \{x_1, \ldots, x_n\}$ to represent the state of the GRN at time t, and $\mathbf{v}(t+1) = \{x_1^{'}, \ldots, x_n^{'}\}$ to represent the state of the GRN at time $t+1$. $\{x_1^{'}, \ldots, x_n^{'}\}$ is calculated from $\{x_1, \ldots, x_n\}$ with Equation 3.1. A state transition pair is $(\mathbf{v}(t), \mathbf{v}(t+1))$.

When f_is in Equation 3.1 are Boolean functions, the G is a Boolean Network (BLN) model [6, 102, 184]. When using BLNs to model GRNs, genes are represented with binary variables with two values ON (1) and OFF (0), which means the genes are turned on or turned off respectively. In qualitative models like the Generalized Logical Formalism [160, 161] and Piecewise Linear Differential Equation [79, 125], the f_is are multi-value discrete functions.

3.2.2 The DFL Algorithm for Learning Gene Regulatory Network Models

The problem of inferring the qualitative model of the GRN from input-output transition pairs (time series of gene expression) is defined as follows.

Definition 3.2.1 (Inference of Qualitative Models) *Let* $\mathbf{V} = \{X_1, \ldots, X_n\}$. *Given a transition table* $\mathbf{T} = \{(\mathbf{v}_j, \mathbf{v}_j^{'}) : j = 1, 2, \ldots, N\}$, *find a set of functions* $\mathbf{F} = \{f_1, f_2, \cdots, f_n\}$,

Table 3.1: The DFL algorithm for learning qualitative models of GRNs.

```
     Algorithm: DFL($\mathbf{V}, K, \mathbf{T}$)
     Input: $\mathbf{V}$ with $n$ genes, indegree $K$,
     $\mathbf{T} = \{(\mathbf{v}_j, \mathbf{v}_j^{'}) : j = 1, 2, \ldots, N\}$.
     Output: $\mathbf{F} = \{f_1, f_2, \cdots, f_n\}$
     Begin:
1     $L \leftarrow$ all single element subsets of $\mathbf{V}$;
2     $\Delta Tree.FirstNode \leftarrow L$;
3    for every gene $Y \in \mathbf{V}$ {
4        calculate $H(Y')$;        //from $\mathbf{T}$
5        $D \leftarrow 1$;        //initial depth
6*       $\mathbf{F}$.add($Sub(Y, \Delta Tree, H(Y'), D, K)$);
     }
7    return $\mathbf{F}$;
     End
```

* The $Sub()$ is a subroutine listed in Table 2.2.

so that $X_i(t+1)$ is calculated from f_i as follows

$$X_i(t+1) = f_i(X_{i1}(t), \ldots, X_{ik}(t)).$$

From Equation 3.1, it is obvious that the qualitative models of GRNs can be learned with the DFL algorithm by using it to learn the regulatory function f_i for each gene sequentially, since the expression levels of the genes at the same time step are assumed to be independent.

As discussed in Section 2.6.1, in GRNs, each gene is estimated on the average to interact with four to eight other genes [19]. That is to say, it is sufficient to consider the combinations whose cardinalities are bounded by a small integer, K. The main steps of the DFL algorithm for learning qualitative models of GRNs are given in Table 3.1, where the subroutine is the same as those in Table 2.2. In learning GRN models, it is not needed to perform the prediction step introduced in Section 2.7.

Since there are n genes in the $\mathbf{V}$, the complexity of the DFL algorithm for learning GRN models will be n times the original complexity in Theorem 2.4.1. We will comprehensively analyze the complexity of the DFL algorithm for learning Boolean networks in next chapter.

3.2.3 Data Quantity for Learning Qualitative Gene Regulatory Network Models

We discuss how much data is necessary to successfully infer $\mathbf{F}$ in this section. Akutsu *et al.* [6] proved that $\Omega(2^k + k \log_2 n)$ [1] transition pairs are the theoretic lower bound to infer the BLNs, where n is the number of genes and k is the maximum indegree of these genes.

Theorem 3.2.1 (Akutsu *et al.* [6]) $\Omega(2^k + k \log_2 n)$ *transition pairs are necessary in the worst case to identify the Boolean network of maximum indegree* $\leq k$*.*

To meet the requirement of multi-state discrete functions in GLF and PLDE models, we introduce the following theorem in [184], which is a generalization of Theorem 3.2.1. The proof of the theorem is also given in Appendix B.

Theorem 3.2.2 (Zheng & Kwoh [184]) $\Omega(b^k + k \log_b n)$ *transition pairs are necessary in the worst case to identify the qualitative GRN models of maximum indegree* $\leq k$ *and the maximum number of discrete level for variables* $\leq b$*.*

In the DFL algorithm, we can introduce a coefficient c to determine the actual size of synthetic data sets as follows,

$$N = c \times (b^k + k \log_b n). \tag{3.2}$$

[1] Just as O-notation provides an asymptotic upper bound on a function, Ω-notation provides an asymptotic lower bound [49]. For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions, $\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$.

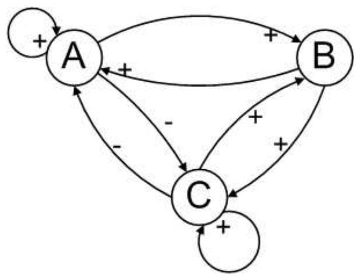


Figure 3.5: A simple GLF model of GRN. The genes are represented by circles. The directed edges represents the regulatory relations between genes. The "+" and "-" beside an edge represents this regulatory relation is activation and repression respectively.

That is to say, the parameter j in Definition 3.2.1 goes from 1 to N. In practice, the c is often bigger than 1, since the sample distribution of the data is not strictly following a predefined distribution. In addition, the large sample size is also a guarantee of convergence of the DFL algorithm, as discussed in Section 2.4.4.

3.3 Learning Models of Generalized Logical Formalism

We implement the DFL algorithm with the Java language version 1.4.1. The experiments in this chapter and those in later chapters are conducted on an HP *AlphaServer* SC computer, with one EV68 1GHz CPU and 1GB memory, running *Tru64* Unix operating system. The implementation software and the data sets, and those used in later chapters, are available at the supplementary website of this thesis.

In this section, we use the DFL algorithm to find a GLF model discussed in [159] and shown in Figure 3.5. In [159], Thieffry and Thomas provided the transition table of the model, which consists of 18 lines. We use this transition table as the input table

Table 3.2: The correlation coefficient matrix of the GLF example in Figure 3.5.

	A'	B'	C'
A	0.2	0.6	-0.7
B	0.3	0	0.3
C	-0.7	0.6	0.2

$\mathbf{T}$ of the DFL algorithm. The DFL algorithm can correctly find that $A' = f_1(A, B, C)$, $B' = f_2(A, C)$, and $C' = f_3(A, B, C)$. The learned f_is are truth tables, since we still lack the tools to simplify the multi-value discrete functions like the Karnaugh maps for Boolean functions. In addition, the activation or repression relations in the graph can be obtained by analyzing the correlation coefficient between genes [21]. The correlation coefficient matrix of the example in Figure 3.5 is listed in Table 3.2.

In the correlation coefficient matrix, positive, negative and zero values indicate activation, repression and no direct interaction respectively. In our example, the 0.2 in the first column of the first line in Table 3.2 means A gives activation to A', and so on. We see that the activation and repression relations in Figure 3.5 are correctly identified with the correlation coefficient matrix in Table 3.2. Hence, the GLF model has correctly been rebuilt from its output data by the DFL algorithm.

3.4 Learning Gene Regulatory Networks Related to Yeast Cell Cycle

In this section, we apply the DFL algorithm to the gene expression data of yeast *Saccharomyces cerevisiae* cell cycle from Cho *et al.* [44], which covers approximately two full cell cycles [44]. In [108], Lee *et al.* reported a GRN related to cell cycle of yeast. The GRN consists of 11 well-known yeast cell cycle regulators, which are Mbp1, Swi4, Swi6,

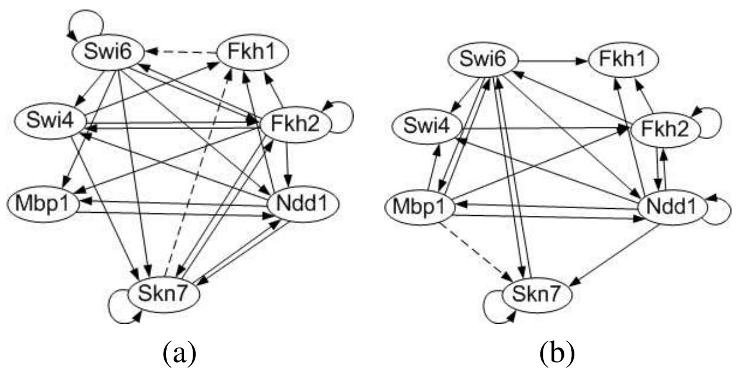


Figure 3.6: The learned GRN model related to yeast cell cycle. (a) The number of discrete levels for gene expression value is 3 and the indegree of the GRN is set to 5. (b) Idem, where the base for gene expression value is 4. The regulators are represented by ovals. The directed edge from Gene A to Gene B means that Gene A is a regulator of Gene B. The solid edges represent regulatory relations that have been verified by other approaches. The dashed edges represent regulatory relations that have not been verified.

Mcm1, Fkh1, Fkh2, Ndd1, Swi5, Ace2, Skn7 and Stb1. The Mcm1, Swi5, Ace2 and Stb1 are relatively loosely related to other genes. Thus, we only consider the remaining 7 genes.

3.4.1 Learning Gene Regulatory Networks with The DFL Algorithm

We discretize the data set in [44] to three and four levels, then rearrange these expression values to state-transition pairs such that the expression values at current time step are the product of expression values at the prior time step. As mentioned in Section 3.1.1, the equal-width binning discretization algorithm, which is an unsupervised discretization method and has been implemented in the DFLearner software, is used here. Finally, we apply the DFL algorithm to the obtained transition table. The learned models are shown in Figure 3.6.

Table 3.3: The literature evidences for the GRN model in Figure 3.6 and Figure 3.7.

Gene	Regulator (Protein)							Evidence
	Mbp1	Swi4	Swi6	Fkh1	Fkh2	Ndd1	Skn7	
MBP1	*3	*	*34		*34	*34		[108], [156]
SWI4	*34	*3	*34		*3	*34	*	[108], [156]
SWI6	* 4	*	*34	3	*34	*	* 4	[108], [156]
FKH1	* 4	*3	*4	*	*34	*34	3	[108], [156]
FKH2	* 4	*34	*3	*3	*34	* 4	*3	[108], [156]
NDD1	*34	*	*34	*	*34	* 4	*3	[108], [156]
SKN7	34	*3	*34		*3	*34	*34	[108]

“*” means regulatory relations. For example, “*” in the first cell of first line means that Mbp1 gives MBP1 gene autoregulation [108]. “3” and “4” represent the regulatory relations found with the DFL algorithm when the bases for expression values are 3 and 4 respectively.

The DFL algorithm is automatic and requires no prior knowledge of the regulatory relations between the genes under consideration. The DFL algorithm is also quite efficient, only needs less than 0.2 seconds for all experiments done.

The literature evidence for regulatory relations represented in Figure 3.6 is shown in Table 3.3. For instance, Swi4 transcription is regulated in late G1 by both SBF(Swi4/Swi6) and MBF(Mbp1/Swi6) [156]. In Figure 3.6, these regulatory relations are identified in (a) and (b) respectively.

From Table 3.3, we obtain the accuracy, sensitivity and precision of the DFL algorithm, and tabulate them in Table 3.5, where the accuracy, sensitivity and precision are defined in Equation 3.3, 3.4 and 3.5. The TP, FP, TN and FN in these equations are defined in Table 3.4.

In Table 3.5, we see that approximate 83 percent of the regulatory relations which have literature evidences are found with the DFL algorithm, when we combine the results from both Figure 3.6 and Figure 3.7. It is also shown that the precision of the DFL algorithm is quite high no matter what the bases for expression values are. That means, it is quite probable that the regulatory relations found with the DFL algorithm are biologically meaningful.

Table 3.4: The prediction measures.

	predicted as positive	predicted as negative
positive	TP	FN
negative	FP	TN

In Table 3.5, it is shown that over 90 percent of the regulatory relations found by the DFL algorithm are biologically significant.

$$\begin{aligned} Accuracy &= \frac{NO.of Correct Predictions}{NO.of Predictions} \\ &= \frac{TP + TN}{TP + FP + TN + FN} \end{aligned} \tag{3.3}$$

$$\begin{aligned} Sensitivity &= \frac{NO.of Correct Positive Predictions}{NO.of Positives} \\ &= \frac{TP}{TP + FN} \end{aligned} \tag{3.4}$$

$$\begin{aligned} Precision &= \frac{NO.of Correct Positive Predictions}{NO.of Positive Predictions} \\ &= \frac{TP}{TP + FP} \end{aligned} \tag{3.5}$$

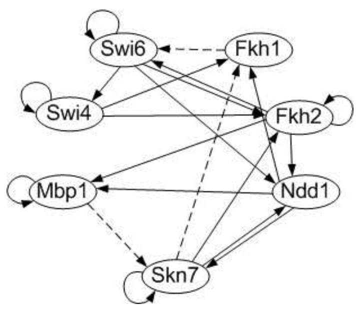
To do a comparison with another commonly used model, Bayesian networks, we apply the K2 algorithm [48] to the same data sets. Then, we calculate the accuracy, sensitivity and precision of the K2 algorithm with respect to literature evidences, and list them in Table 3.5

Table 3.5: The accuracy, sensitivity and precision of the DFL algorithm and the K2 algorithm.

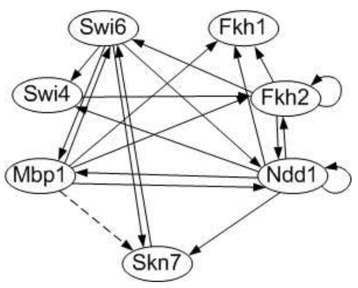
	Accur.	Sensi.	Preci.
DFL ($b = 3$)	65	67	90
DFL ($b = 4$)	63	60	96
DFL (Combined)	80	83	92
K2 ($b = 3$)	27	17	88
K2 ($b = 4$)	22	12	83
K2 (Combined)	33	24	91

also. In Table 3.5, it is shown that the measures of the K2 algorithm are substantially lower than those of the DFL algorithm. Another important thing is that the autoregulations cannot be represented by Bayesian networks due to fact that the structures of them are directed acyclic graphs [131]. Therefore, the autoregulations are predeterminately missed no matter what algorithm for learning Bayesian networks is used. However, the autoregulations are very common in GRNs as shown in Table 3.3, in which the diagonal line from upper-left corner to lower-right corner is fully occupied with autoregulation evidences.

Some regulatory relations which have literature evidence are not found by the DFL algorithm, as shown in Table 3.3. This is also shown by the sensitivity value in Table 3.5. There are mainly two reasons for this discrepancy. First, the size of the data set is too small. Second, there is noise in the gene expression data. It is reasonable to expect that the model obtained from the DFL algorithm will become more reasonable when the input data is larger and more precise. Further, there are also some putative regulatory relations (represented by dashed edges) to be verified yet. When we calculate the measures in Table 3.5, we count these relations as false positives.



(a)



(b)

Figure 3.7: The learned GRN model for yeast cell cycle with the ϵ function method. (a) The base for gene expression value is 3, the indegree of the GRN is 5, and the ϵ is 0.2. (b) The base for gene expression value is 4, the indegree of the GRN is 5, and the ϵ is 0.15. The legends are the same as those of Figure 3.6.

3.4.2 Applying The ϵ Value Method to Yeast Cell Cycle Data

We apply the ϵ value method to the same data set and the results in shown in Figure 3.7. As shown in Figure 3.7 (a) and (b), some regulatory relations that are not found in 3.6 (a) are identified with the ϵ value method. For example, the autoregulation of Mph1 and Swi4 are successfully found in Figure 3.7 (a). In Figure 3.7 (b), the regulation of Fkh1 by Mbp1 is identified. In addition, the regulation of Fkh2 by Fkh1 is identified in experiments with $b = 3$ and $\epsilon = 0.25$ (not shown in Figure 3.7).

However, some regulatory relations also disappear when we apply the ϵ value method. For instance, the regulation of Fkh1 by Fkh2 disappears in Figure 3.7 (a). Generally, the GRN model tends to become scarcer (contain fewer edges) when the value of ϵ increases. This is due to the fact that fewer genes can satisfy the requirement of the ϵ value method when ϵ increases.

Finally, we give unified models in Figure 3.8, in which (a) combines results in both Figure 3.6 and Figure 3.7, and (b) is a combined Bayesian network model learned by the

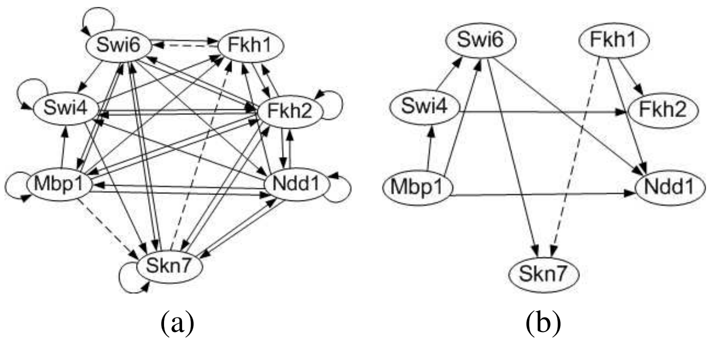


Figure 3.8: The combined GRN models. (a) Combined model of Figure 3.6 and Figure 3.7. (b) Combined Bayesian network structure learned with the K2 algorithm where the base for expression value is set to 3 and 4 respectively. The legends are the same as those of those of Figure 3.6.

K2 algorithm when the base for expression value is 3 and 4. It is shown in Figure 3.8 that the model found with the DFL algorithm is more significant than that learned with the K2 algorithm. As we mentioned before, there are no autoregulations found in Figure 3.8 (b), but 6 out of 7 autoregulations are found in Figure 3.8 (a).

3.5 Conclusions

The GRNs are critical for understanding the developmental process. Qualitative models provide structures of GRNs with easily understandable rules between the regulators and the genes. In this chapter, we apply the DFL algorithm to learning qualitative models of GRNs. The DFL algorithm can correctly find the original structure of a GLF model. While incorporating the analysis the correlation coefficient matrix of the genes, the regulation relations of the GLF model are correctly identified as activations or repressions. We also show that the DFL algorithm finds biologically significant GRNs related to yeast cell cycle from limited time-series microarray gene expression profiles.

Chapter 4

Learning Boolean Networks

CURRENT methods for learning Boolean networks (BLNs) are inefficient, with their complexity of at least $O(N \cdot n^{k+1})$. Particularly, it is still an open problem to Boolean learn functions with bounded number of inputs k in $o(N \cdot n^k)$ time [7]. This high-order complexity of $O(N \cdot n^{k+1})$ also makes it impractical to use them for learning Gene Regulatory Network (GRN) models from real data on a large scale.

In this chapter, we will use the DFL algorithm to learn BLNs. We prove that each bounded OR/AND Boolean function can be learned with $O(k \cdot (N + \log n) \cdot n)$ time complexity, hence OR/AND BLNs can be inferred with $O(k \cdot (N + \log n) \cdot n^2)$ time complexity. For general Boolean functions, we will show that the DFL algorithm still maintains its complexity of $O(k \cdot (N + \log n) \cdot n)$ in most cases, which is supported and verified with comprehensive experimental results.

We will also discuss the learning problem of unbounded Boolean functions. We show that the complexity of learning unbounded OR/AND Boolean functions are mainly coming from the sample complexity, not from the searching.

To deal with noisy data sets, we use the ϵ value method to find the original BLN models. We show that the DFL algorithm can correctly find the original BLNs even when there are substantial amount of noisy samples, as high as 20%, in the data sets.

This chapter is organized as follows. In Section 4.1, we will describe current methods for learning BLNs, and discuss two open problems in learning Boolean functions. In Section 4.2, we define the inference problem of BLNs, and introduce a method to obtain correct truth tables from noisy data sets. In Section 4.3, we introduce the evaluation criterion for learning BLNs. In Section 4.4, we prove the complexity of the DFL algorithm for learning OR/AND BLNs. In Section 4.5, we analyze performance of the DFL algorithm for learning the general BLNs. In Section 4.6, we perform experiments to validate our proof about the complexity of the DFL algorithm. In Section 4.7, we discuss the related models to BLNs, and show that the DFL algorithm is also useful to learn these related models. In Section 4.8, we summarize this chapter and propose some future directions for the DFL algorithm on the learning of Boolean functions/networks.

4.1 Current Methods For Learning Boolean Networks

Since (1) the architectures of GRNs are of the primary importance; (2) the number of microarray gene expression experiments is limited; (3) Boolean networks (BLNs) are relatively computational tractable; BLNs as models have received much attention in reconstructing GRNs from gene expression data sets [6–9, 95, 105, 113, 119, 148, 184]. Liang *et al.* [113] proposed the REVEAL algorithm, with $O(N \cdot n^{k+1})$ complexity, to reconstruct BLNs from binary transition pairs. Akutsu *et al.* [6] introduced an algorithm for the same purpose, but the complexity of it, $O(k \cdot 2^{2^k} \cdot N \cdot n^{k+1})$, is even worse. Akutsu *et al.* [7] proposed another algorithm with complexity of $O(N^{\omega-2} \cdot n^{k+1} + N \cdot n^{k+\omega-2})$ where $\omega = 2.376$ to find the BLNs with high probability. Schmulevich *et al.* [148] introduced an algorithm with the complexity of $O(k \cdot 2^{2^k} \cdot N \cdot n^{k+1})$. Lähdesmäki *et al.* [105] proposed an algorithm with $O(\binom{n}{k} \cdot n \cdot N \cdot poly(k))$ complexity, where $poly(k)$ is k in most cases. These

Table 4.1: The summary of complexities of different algorithms for learning BLNs.

Time Complexity	Reference
$O(N \cdot n^{k+1})$	[113]
$O(k \cdot 2^{2^k} \cdot N \cdot n^{k+1})$	[6]
$O(N^{\omega-2} \cdot n^{k+1} + N \cdot n^{k+\omega-2})$	[7]
$O(k \cdot 2^{2^k} \cdot N \cdot n^{k+1})$	[148]
$O(\binom{n}{k} \cdot n \cdot N \cdot poly(k))$	[105]

algorithms are computationally expensive, with their complexities of at least $O(N \cdot n^{k+1})$, as summarized in Table 4.1.

In the field of machine learning, there are also many algorithm introduced for learning Boolean functions [29, 35, 66, 94, 114, 120, 121, 123, 142]. If these algorithms are modified to learning BLNs of bounded indegree of k, i.e., n Boolean functions with k inputs, their complexities are also at least $O(N \cdot n^{k+1})$.

Akutsu *et al.* [9] proposed an approximation algorithm called GREEDY1 with the complexity of $O((2 \ln N + 1) \cdot N \cdot n^2)$, but the success ratio of the GREEDY1 algorithm is not satisfactory, especially when k is small. For example, the success ratio of the GREEDY1 algorithm is only around 50% no matter how many learning samples are used when $k = 2$ for the general Boolean functions.

Hence, more efficient and accurate algorithms are indispensable to simulate GRNs with BLNs on a large scale. In our work [184], we introduced the Discrete Function Learning (DFL) algorithm with the expected complexity of $O(k \cdot (N + \log n) \cdot n^2)$ to reconstructing qualitative models of GRNs from gene expression data sets. In this chapter, we investigate the two open problems listed below with the DFL algorithm.

Problem 1 *Are there efficient algorithms for inferring AND/OR functions with unbounded indegree* [9]? (This problem is NP-hard.)

Problem 2 *Are there algorithms with $o(N \cdot n^k)$ complexity for learning Boolean functions* *with bounded indegree k* [7]? (This problem is of polynomial time.)

For the first problem, we will prove that the complexity of the DFL algorithm is $O((N+\log n)\cdot n^2)$ for the unbounded OR/AND functions, in the worst case given enough samples. For the second problem, we will prove that for learning OR/AND functions with bounded indegree k, the complexity of the DFL algorithm is strictly $O(k\cdot(N+\log n)\cdot n)$ given enough samples. The experimental results soundly support that the time complexity of the DFL algorithm is $O(k\cdot(N+\log n)\cdot n)$ for inferring OR/AND Boolean functions, and $O(k\cdot(N+\log n)\cdot n^2)$ for inferring OR/AND BLNs. Supported by experimental results, the complexity of the DFL algorithm is still $O(k\cdot(N+\log n)\cdot n)$ for learning general Boolean functions in most cases.

For the noisy data sets, the DFL algorithm is used in combination with the ϵ value method, which is called the ϵ function method in our early work [184]. The experimental results show that the DFL algorithm can correctly find the original BLNs in over 98% data sets, even when there are 20% noisy instances in the data sets.

4.2 Learning Boolean Networks With The DFL Algorithm

In this section, we first define the problem of learning BLNs. Then, we discuss how to obtain correct truth table from noisy data sets.

4.2.1 Problem Definition

The problem of inferring the BLN model of the GRN from input-output transition pairs (time series of gene expression) is defined as follows, where the difference between the problem defined in Definition 3.2.1 is that the function set $\mathbf{F}$ here is a set of Boolean functions.

Definition 4.2.1 (Inference of BLNs) *Let* $\mathbf{V} = \{X_1, \ldots, X_n\}$. *Given a transition table* $\mathbf{T} = \{(\mathbf{v}_j, \mathbf{v}_j^{'}) : j = 1, 2, \ldots, N\}$, *find a set of Boolean functions* $\mathbf{F} = \{f_1, f_2, \cdots, f_n\}$, *so that* $X_i(t+1)$ *is calculated from* f_i *as follows*

$$X_i(t+1) = f_i(X_{i1}(t), \ldots, X_{ik}(t)).$$

Akutsu *et al.* [7] proposed another set of problems for inferring BLNs. The **CONSISTENCY** problem defined by Akutsu *et al.* [7] is to decide whether or not there exists a Boolean network consistent with the given examples, and output one if it exists. So, the **CONSISTENCY** problem is the same as Definition 4.2.1.

To solve the problem in Definition 4.2.1, the version of the DFL algorithm in Table 3.1 is also applicable, since BLNs are a special type of qualitative models of GRNs as discussed in Section 3.2.1.

4.2.2 Obtaining Correct Truth Table From Noisy Data Sets

When data sets are noisy, the DFL algorithm, combined with the ϵ value method, can still find the correct structure of BLNs, but it is probable that there are two different instances of $(\mathbf{Pa}(X_i^{'}), X_i^{'})$ for each instance of $\mathbf{Pa}(X_i^{'})$. Recall that the counts of different instances of $(\mathbf{Pa}(X_i^{'}), X_i^{'})$ are obtained in the learning process, as discussed in Section 2.4. In the noisy data sets, these count values can be used to obtain the correct truth tables for $X_i^{'}$. Specifically, the DFL algorithm finds all the instances of $(\mathbf{Pa}(X_i^{'}), X_i^{'})$ for each instance of $\mathbf{Pa}(X_i^{'})$ and chooses the one with largest count value.

4.3 Evaluation Criterion for Learning Boolean Networks

In the context of GRN, it is important to find the correct regulatory relations between genes. Therefore, we use the sensitivity of the edges (the inputs of functions) in the models to evaluate the performance of the DFL algorithm for inferring BLNs. The structure sensitivity measures the percentage of correctly predicted edges of the BLNs by the DFL algorithm. Formally, the structure sensitivity is defined in Equation 3.4, i.e.,

$$Sensitivity = \frac{TP}{TP + FN},$$

where the TP is true positive predictions, i.e., correctly predicted edges, the FN is the false negative predictions, i.e., the missing edges. The denominator, $TP + FN$, is the total number of edges in the original BLNs. For instance, if the original Boolean network is defined by $X_i^{'} = f(X_1, X_2, X_3)$, and the DFL algorithm obtains a network defined by $X_i^{'} = f(X_1, X_3, X_{10})$ from a data set, then the sensitivity is 2/3.

Let us consider the factors which will affect the sensitivity of the DFL algorithm. As shown in Figure 4.1, the $I(X_j; X_i^{'})$ of the Boolean network is affected by two factors, sample size N and noise level of data sets λ. In a data set with noise level λ, there are λ percent of samples, whose $x_i^{'} \neq f(\mathbf{pa}(x_i))$.

First, we consider the effect of N. Based on Theorem 2.2.4, the $I(X_j; X_i^{'})$ will be zero for $j = 4, \ldots, 10$ given enough samples in the example of Figure 4.1. Actually, when $N = 1000$, the $I(X_j; X_i^{'})$ for $j = 4, \ldots, 10$ is almost zero, as shown in Figure 4.1. But when N is small, the $I(X_j; X_i^{'})$ will be very different from the *Golden Rule*. For instance, the DFL algorithm finds that $X_i^{'} = f(X_2, X_7, X_9)$, which is incorrect, since $I(X_2, X_7, X_9; X_i^{'}) = H(X_i^{'})$ when $N = 20$ in the example of Figure 4.1. This is due the fact that $I(X_j; X_i^{'})$ is not zero when sample size is small. Since $p(\mathbf{x})$ is unknown from

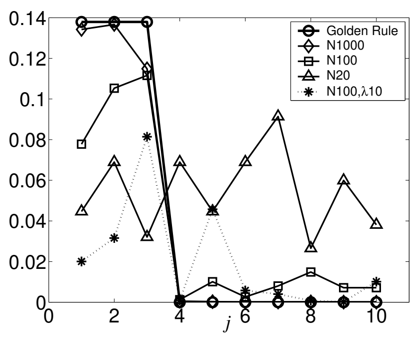


Figure 4.1: The $I(X_j; X_i^{'})$ for OR BLNs with 10 variables, where $X_i^{'} = X_1 + X_2 + X_3$. The unit is bit. The curve marked with circle is learned from the truth table of $X_i^{'} = X_1 + X_2 + X_3$, so it is the ideal case, or the *Golden Rule*. The curves marked with circles, diamonds, squares, triangles and stars represent the values obtained from truth table of $X_i^{'} = X_1 + X_2 + X_3$, data sets of $N = 1000$, $N = 100$, $N = 20$ and $N = 100$ with 10% noise respectively.

small data sets, $I(X_j; X_i^{'})$ for $j = 4, \ldots, 10$ is not zero, as shown by the curve for $N = 20$ in Figure 4.1. Therefore, merely by coincidence, there probably exist other subsets of $\mathbf{V}$ which satisfy the criterion of Theorem 2.2.2. Consequently, it is probable that the DFL algorithm cannot find the original BLNs. Note that Theorem 2.4.1 is correct no matter how many learning samples are provided. In case of small sample size, like $N = 20$ in the example, the obtained BLNs are still consistent with the learning data sets. But, the sensitivity of the DFL algorithm becomes 1/3 for this example, since only 1/3 edges of the original network is correctly identified. In this case, the consistency used by Akutsu *et al.* [6] is not a suitable criterion to measure the performance of a learning algorithm. This the reason for which we use the sensitivity to evaluate the performance of the DFL algorithm.

Then, we consider the effect of noise level λ. As shown by the dotted curve in Figure 4.1, the $I(X_j; X_i^{'})$ for the noisy data set is quite different from the *Golden Rule*. The DFL algorithm finds the correct network structure $\{X_1, X_2, X_3\} \to X_i'$, by setting $K = 3$ and $\epsilon = 0.68$. The sensitivity of the DFL algorithm is still 1 for the noisy data set in this example.

Subsequently, the DFL algorithm finds the correct truth table by using the method introduced in Section 4.2.2. For the example shown in Figure 4.1, the DFL algorithm finds $((1,0,1),1)$ with count 10 for the instance $(1,0,1)$ of $\mathbf{Pa}(X_i') = \{X_1, X_2, X_3\}$ in the noiseless data sets with 100 samples, but finds $((1,0,1),1)$ with count 9 and $((1,0,1),0)$ with count 1 in the data sets with 100 samples and 10% noise. Hence, the DFL algorithm will choose $((1,0,1),1)$, which is correct, for the instance $(1,0,1)$ of $\{X_1, X_2, X_3\}$ in this case. Similarly, the whole truth table of X_i' can correctly be found.

4.4 The Analysis Of Some Special Boolean Networks

In this section, we first analyze the mutual information between variables in special BLNs where all relations are logic OR(AND) operation. Then, we propose the theorems about the complexity of the DFL algorithm for learning OR/AND BLNs.

4.4.1 The Mutual Information in OR Boolean Networks

Formally, we define the OR BLN as follows.

Definition 4.4.1 *The OR BLN of a set of binary variables $\mathbf{V} = \{X_1, \ldots, X_n\}$ is, $\forall\, X_i$*

$$X_i' = X_{i1}(t) + \ldots + X_{ik}(t), \tag{4.1}$$

where the "$+$" is the logic OR operation.

Table 4.2: The truth table of $B^{'} = A + C$.

A	C	$B^{'}$
0	0	0
0	1	1
1	0	1
1	1	1

Next, we derive the value of $I(X_{ij}; X_i^{'})$, $\forall X_{ij} \in \mathbf{Pa}(X_i^{'})$. The data sets are generate with the original generation BLNs. Then, based on Theorem 2.1.4, if enough samples are provided, the distributions of $\mathbf{Pa}(X_i^{'})$ and $X_i^{'}$ will tend to be those in the truth table of $X_i^{\prime} = f(\mathbf{Pa}(X_i^{\prime}))$. Consider the simple function $B^{'} = A + C$ in Table 4.2.

Without loss of generality, we derive $I(A; B^{'})$. From the truth table of $B^{'}$, the $H(A)$, $H(B^{'})$, and $H(A, B^{'})$ are obtained as follows. First, we have

$$H(A) = -2 \times \frac{1}{2} \log \frac{1}{2} = 1(bit),$$

and

$$H(B^{'}) = -\frac{1}{4} \log \frac{1}{4} - \frac{3}{4} \log \frac{3}{4} = 0.81(bits).$$

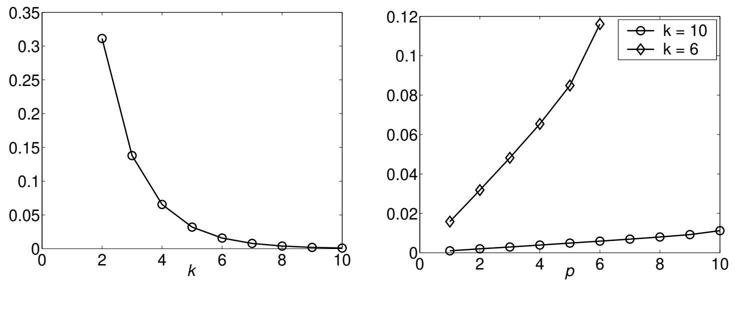
Then, we compute the joint entropy $H(A, B^{\prime})$. There are only three possible instance for the tuple $(A, B^{'})$, i.e., $(0, 0)$, $(0, 1)$ and $(1, 1)$. Therefore, we obtain

$$H(A, B^{'}) = -\frac{1}{4} \log \frac{1}{4} - \frac{1}{4} \log \frac{1}{4} - \frac{2}{4} \log \frac{2}{4} = 1.5(bits).$$

Finally, we obtain

$$I(A; B^{'}) = H(A) + H(B^{'}) - H(A, B^{'}) = 0.31(bits). \tag{4.2}$$

Generally, we have Theorem 4.4.1 to compute the $I(X_{ij}; X_i^{'})$ in OR BLNs.



(a) (b)

Figure 4.2: Mutual information in OR function $X_i' = X_{i1} + \ldots + X_{ik}$. The unit is bit. (a) The $I(X_{ij}; X_i')$ as a function of k, $\forall X_{ij} \in \mathbf{Pa}(X_i')$. (b) $I(\{X_{i(1)}, \ldots, X_{i(p)}\}; X_i')$ as a function of p, where $k = 6, 10$, and p goes from 1 to k.

Theorem 4.4.1 *Given enough samples for an OR BLN over* $\mathbf{V}$*, the mutual information between* $\forall X_{ij} \in \mathbf{Pa}(X_i') = \{X_{i1}, \ldots, X_{ik}\}$ *and* X_i' *is*

$$I(X_{ij}; X_i') = \frac{1}{2} - \frac{2^k - 1}{2^k} \log \frac{2^k - 1}{2^k} + \frac{2^{k-1} - 1}{2^k} \log \frac{2^{k-1} - 1}{2^k}. \tag{4.3}$$

From Equation 4.3, we see that $I(X_{ij}; X_i')$ is strictly positive, and tends to zero when $k \to \infty$, as shown in Figure 4.2 (a). In Figure (a). In Figure 4.2 (a), when $k = 2$, the 0.31 bits, which is the same as the derived $I(A; B')$ in Equation 4.2. Intuitively, when k increases, there would be more "1" in the X_i' column of the truth table while only one "0" whatever value the k is. That is to say, X_i' tends to take the value "1" with higher probability, or there is less uncertainty in X_i' when k increases, which causes $H(X_i')$ to decrease. From Theorem 2.1.6, $H(X_i') = I(\mathbf{Pa}(X_i'); X_i')$, thus, $I(\mathbf{Pa}(X_i); X_i')$ also decreases. Therefore, each X_{ij} shares less information with X_i' when k increases.

Similar to Theorem 4.4.1, we have Theorem 4.4.2 for computing $I(\{X_{i(1)}, \ldots, X_{i(p)}\}; X_i^{'})$, $\forall X_{i(j)} \in \mathbf{Pa}(X_i^{'}), j = 1, 2, \ldots, p$ and $p \le k$. Here, the parentheses in $X_{i(j)}$s are used to denote that the order of input variables is irrelevant to the value $I(\{X_{i(1)}, \ldots, X_{i(p)}\}; X_i^{'})$. From Theorem 4.4.2, we further have Theorem 4.4.3.

Theorem 4.4.2 *Given enough samples. In OR BLNs with maximum indegree k over $\mathbf{V}$, $\forall 1 \le p \le k$, $X_{i(1)}, X_{i(2)}, \ldots, X_{i(p)} \in \mathbf{Pa}(X_i^{'})$, the mutual information between $\{X_{i(1)}$, $X_{i(2)}, \ldots, X_{i(p)}\}$ and $X_i^{'}$ is*

$$I(\{X_{i(1)}, X_{i(2)}, \ldots, X_{i(p)}\}; X_i^{'}) = \frac{p}{2^p} \frac{2^k - 1}{2^k} \log \frac{2^k - 1}{2^k} + \frac{2^{k-p} - 1}{2^k} \log \frac{2^{k-p} - 1}{2^k}. \quad (4.4)$$

Theorem 4.4.3 *Given enough samples. In OR BLNs with maximum indegree k over $\mathbf{V}$, $\forall 2 \le p \le k$, $I(\{X_{i(1)}, X_{i(2)}, \ldots, X_{i(p)}\}; X_i^{'}) > I(\{X_{i(1)}, X_{i(2)}, \ldots, X_{i(p-1)}\}; X_i^{'})$.*

From Theorem 4.4.3, it is known that when variables from $\mathbf{Pa}(X_i^{'})$ are added to the candidate parent set $\mathbf{U}$ for $X_i^{'}$, the mutual information $I(\mathbf{U}; X_i^{'})$ is increasing, which is also shown in Figure 4.2 (b).

4.4.2 The Complexity Analysis for Bounded OR Boolean Networks

Since the DFL algorithm only checks $\sum_{i=0}^{k-1}(n-i) \approx kn$ combinations in the searching process for each gene in OR BLNs, we obtain the following theorem.

Theorem 4.4.4 *Given enough samples for an OR BLN with maximum indegree k over $\mathbf{V}$, then the DFL algorithm can identify the OR BLN in $O(k \cdot (N + \log n) \cdot n^2)$ time strictly.*

Enough learning samples are required in Theorem 4.4.4, since if sample size is small, it is possible that variables in $\mathbf{Pa}(X_i^{'})$ do not share larger mutual information than variables in $\mathbf{V} \setminus \mathbf{Pa}(X_i^{'})$ do. In the example demonstrated in Figure 4.1, it is shown that when

$N = 20$, $I(X_7; X_i^{'}) > I(X_j; X_i^{'}), j = 1, 2, 3$. However, $\mathbf{Pa}(X_i^{'}) = \{X_1, X_2, X_3\}$ in this example, so it takes more steps before the DFL algorithm finally finds the target subset $\{X_1, X_2, X_3\}$. Therefore, the complexity of the DFL algorithm becomes worse than $O(k \cdot (N + \log n) \cdot n^2)$ in these cases. Fortunately, based on Theorem 3.2.1, a sufficient sample size for successfully inferring BLNs can be known *apriori*. For instance, for learning an OR BLN with 100 genes and bounded indegree of 3, the sample size can be obtained by multiplying $(2^k + k \log_2 n)$ in $\Omega(2^k + k \log_2 n)$ of Theorem 3.2.1 with 3 to 4. In our experiments, we find that when k increases, a larger constant should be used to obtain the enough learning sample size by multiplying $(2^k + k \log_2 n)$.

If there are some variables taking their inverse, then the OR BLN will change. We call these kinds of OR BLNs "generalized OR BLNs".

Definition 4.4.2 *The generalized OR BLN of a set of binary variables* $\mathbf{V} = \{X_1, \ldots, X_n\}$ *is,* $\forall\, X_i$

$$X_i^{'} = X_{i1}(t) + \ldots + X_{ik}(t), \tag{4.5}$$

where the "+" is the logic OR operation, X_{ij}*s can also take their inverse.*

For generalized OR BLNs, the DFL algorithm also keeps its time complexity of $O(k \cdot (N + \log n) \cdot n^2)$.

Corollary 4.4.1 *Given enough samples for a generalized OR BLN with maximum indegree* k *over* $\mathbf{V}$*, then the DFL algorithm can identify the BLN in* $O(k \cdot (N + \log n) \cdot n^2)$ *time strictly.*

In a binary system, there are only two values for variables. If we replace the 0 in OR BLN truth table with 1 and vice versa, the resulted BLN will have an opposite probability of 1 and 0 to those of the original OR BLN. It is easy to show that such a BLN is an AND BLN defined in the following.

Definition 4.4.3 *The AND BLN of a set of binary variables* $\mathbf{V} = \{X_1, \dots, X_n\}$ *is,* $\forall\ X_i$

$$X_i^{'} = X_{i1}(t) \cdot \ldots \cdot X_{ik}(t), \tag{4.6}$$

where the "$\cdot$" is the logic AND operation.

From Theorem 4.4.4, it is straightforward to obtain the following corollary.

Corollary 4.4.2 *Given enough samples for an AND BLN with maximum indegree* k *over* $\mathbf{V}$*, then the DFL algorithm can identify the BLN in* $O(k \cdot (N + \log n) \cdot n^2)$ *time strictly.*

Similarly to generalized OR BLN, we define generalized AND BLN in the following and obtain Corollary 4.4.3.

Definition 4.4.4 *The generalized AND BLN of a set of binary variables* $\mathbf{V} = \{X_1, \dots, X_n\}$ *is,* $\forall\ X_i$

$$X_i^{'} = X_{i1}(t) \cdot \ldots \cdot X_{ik}(t), \tag{4.7}$$

where the "$\cdot$" is the logic AND operation, X_{ij}*s can also take their inverse.*

Corollary 4.4.3 *Given enough samples for a generalized AND BLN with maximum indegree* k *over* $\mathbf{V}$*, then the DFL algorithm can identify the BLN in* $O(k \cdot (N + \log n) \cdot n^2)$ *time strictly.*

From Theorem 4.4.4 and Corollary 4.4.2, it is straightforward to know that the DFL algorithm can find a OR/AND function with k inputs is $O(k \cdot (N + \log n) \cdot n)$ time. **Problem 2** in Section 4.1 is solved for the OR/AND functions with the DFL algorithm.

4.4.3 The Complexity Analysis for Unbounded OR Boolean Networks

From Theorem 4.4.4, we see that the complexity of the DFL algorithm will become $O((N + \log n) \cdot n^3)$ if the k becomes n. That is to say that if the indegree of the variables is unbounded in OR/AND Boolean functions, the complexity of the DFL algorithm is $O((N + \log n) \cdot n^2)$ given enough samples.

But according to Theorem 3.2.1, it needs $\Omega(2^n + n \log_2 n)$ samples to successfully find the BLNs of indegree n. In other word, $N \sim (2^n + n \log_2 n)$, i.e., the sample complexity for inferring unbounded BLNs is NP-hard. The NP-hard sample complexity in turn makes the complexity of the DFL algorithm be NP-hard in this case, even if enough samples are provided. Therefore, **Problem 1** in Section 4.1 is still NP-hard even enough samples are provides.

However, if the indegree of OR/AND BLNs is undetermined but known to be much smaller than n, which is often true in real gene expression data sets (as discussed in Section 2.6.1), the DFL algorithm is still useful. In these cases, the expected cardinality K can assigned as n, and the DFL algorithm can automatically find how many variables are sufficient for each X_i'. From Theorem 4.4.4, the DFL algorithm still has the complexity of $O(k \cdot (N + \log n) \cdot n^2)$ for OR/AND BLNs given enough samples.

4.5 The Analysis of General Boolean Functions

There are 2^{2^k} Boolean functions for X_i' with k inputs. In our empirical studies, most of the 2^{2^k} Boolean functions can be found in $O(k \cdot (N + \log n) \cdot n)$ time. In the rest cases, $I(X_{ij}; X_i') = 0, \forall X_{ij} \in \mathbf{Pa}(X_i')$, such as in the exclusive OR (XOR) function, e.g., $X_1 \bigoplus X_2$, and the inversion of the XOR function, e.g., $\neg(X_1 \bigoplus X_2)$. We discuss the later cases and the constant functions in this section.

4.5.1 The Cases Which Require More Computation Time

In some Boolean functions, such as the XOR functions, the $I(X_{ij}; X_i^{'}) = 0$, $\forall X_{ij} \in \mathbf{Pa}(X_i^{'})$. This makes it very unlikely to rank the X_{ij} in the front part of the list after the sort step in line 7 of Table 2.2. However, the DFL algorithm can still find the correct subset $\{X_{i1}, \ldots, X_{ik}\}$ for each $X_i^{'}$ with $O(N \cdot n^k)$ time in the worst case, since the DFL algorithm guarantees the exhaustive search of the searching space $\mathcal{S}_k$.

Fortunately, in the empirical studies to be performed in Section 4.6, the worst case happens with very low probability, about 1%, for inferring random Boolean functions of indegree 3. The experimental results show that although the DFL algorithm uses more steps for finding the target subsets for those functions whose $I(X_{ij}; X_i^{'}) = 0$ than for other functions, its complexity is still in polynomial time for each $X_i^{'}$.

In the context of GRNs, the XOR function is also unlikely to happen between different regulators of a gene. In some cases, a gene can be activated by several activators, and any of these activators is strong enough to activate the gene. This introduces an "OR" logic between these activators. In other cases, several activators must simultaneously bind to their binding sites in the *cis*-regulatory region to turn on the gene, which introduces an "AND" logic. Repressors often are used to turn off the gene, so it introduces the "NOT" logic. Suppose that the regulators act with XOR relation, then there need to have odd number of regulators with high expression level (in logic 1 state). For example, a gene X has one activator A and one repressor R, then $X = A \bigoplus R = A\overline{R} + \overline{A}R$, where the second term in the right side is clearly unreasonable. The second term in the right side says that if the activator is on its low level and the repressor is in its high level, then the gene G will be turned on. From above analysis, it is known that such a situation will not happen in a real biological system.

4.5.2 The Constant Functions

The DFL algorithm will output that "X'_i is a constant" for two extreme cases, i.e., X'_i $= f(X_{i1}, \ldots, X_{ik}) = 1$ or 0, $\forall(x_{i1}, \ldots, x_{ik})$. Since in these two cases, the X'_i is a constant, there is no entropy for X'_i. In other words, the information content of X'_i is zero. Therefore, there is no need to know which subset of features are the genuine inputs of X'_i in these two cases.

In case of GRNs, some genes also show a constant expression level in a specific biological process. For example, a large amount of genes do not show a significant change in their expression level during the yeast *Saccharomyces cerevisiae* cell cycle in the study of [158]. These genes are considered as *house-keeping* genes and are removed before further analysis of the expression data sets [158].

4.6 Results

In this section, we first introduce the synthetic data sets of BLN models that we use. Then, we perform experiments for various data sets to validate the efficiency of the DFL algorithm. In the following sections, we carry out experiments on small data sets and noisy data sets to examine the sensitivity of the DFL algorithm.

4.6.1 Synthetic Data Sets of Boolean Networks

We present the synthetic data sets of BLNs in this section. For a BLN consisting of n genes, the total state space would be 2^n. The $\mathbf{v}$ of a transition pair is randomly chosen from 2^n possible instances of $\mathbf{V}$ with the Discrete Uniform Distribution, i.e., $p(i) = \frac{1}{2^n}$, where i is randomly chosen one value from 0 to $2^n - 1$ inclusively. Since the DFL algorithm examines different subsets in the kth layer of $\Delta Tree$ with lexicographic order, the run time of the DFL algorithm may be affected by the different position of the target subsets in the kth layer of $\Delta Tree$. Therefore, we select the first and the last k variables in $\mathbf{V}$ as the inputs for all $X_i^{'}$. The data sets generated from the first k and last k variables are named as "head" and "tail" data sets. There are 2^{2^k} different Boolean functions when the indegree is k. Then, we use OR function (OR), AND function (AND), or one of the Boolean functions randomly selected from 2^{2^k} possible functions (RANDOM) to generate the $\mathbf{v}^{'}$, i.e., $f_1 = f_2 = \ldots = f_n$. If a data set is generated by OR function defined with the first or last k variables, then we name it as an OR-h or OR-t (OR-tail) data set, and so on.

4.6.2 Experiments for Time Complexity

As introduced in Section 4.4, the complexity of the DFL algorithm is $O(k \cdot (N + \log n) \cdot n^2)$ for OR/AND BLNs. We will first perform experiments for OR/AND BLNs to further validate our analysis. Then, we perform experiments for random BLNs to examine the complexity of the DFL algorithm for them.

Complexity for Bounded OR/AND Boolean Networks

In all experiments of this section, the expected cardinality K and ϵ of the DFL algorithm is set to k of the generation BLNs and 0 respectively. In this study, we will perform three types of experiments to investigate the effect of k, n and N respectively. In each type of experiments, we will change only one of k, n, N, and keep the other two unchanged. We generate 20 OR and 20 AND data sets for each k, N and n, specifically, 10 OR-h, 10 OR-t, 10 AND-h and 10 AND-t data sets. Then we use the average value of these 20 data sets as the run time for this k, N and n value.

First, we perform the experiments for various k, when $n = 1000$, $N = 600$. The run times are show in Figure 4.3 (a). Then, we perform the experiments for various N, when

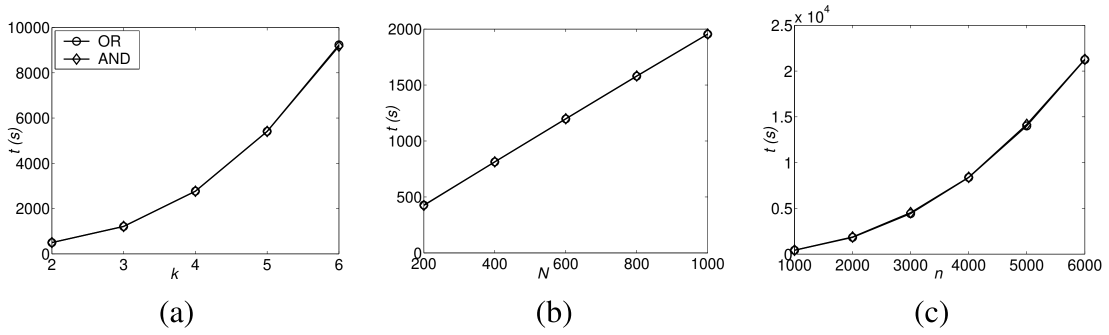


Figure 4.3: The run time, t (vertical axes, shown in seconds), of the DFL algorithm for inferring the bounded BLNs. The values shown are the average of 20 data sets. The curves marked with circles and diamonds are for OR and AND data sets respectively. (a) The run time vs k, when $n = 1000$ and $N = 600$. (b) The run time vs N, when $n = 1000$ and $k = 3$. (c) The run time vs n, when $k = 3$ and $N = 200$.

$n = 1000$, $k = 3$. The run times of these experiments are shown in Figure 4.3 (b). Finally, we perform the experiments for n, and let $k = 3$, $N = 200$. The run times are shown in Figure 4.3 (c). In all experiments for various k, N and n, the DFL algorithm successfully finds the original BLNs.

As shown in Figure 4.3, the DFL algorithm uses almost the same time to learn OR and AND BLNs. As introduced in Theorem 4.4.4 and Corollary 4.4.2, the DFL algorithm has the same complexity of $O(k \cdot (N + \log n) \cdot n^2)$ for learning OR and AND BLNs. The run time of the DFL algorithm grows slightly faster than linear growth with k, as shown in Figure 4.3 (a). This is due to the fact that the computation of entropy and mutual information needs more time when k increases. As shown in Figure 4.3 (b), the run time of the DFL algorithm grows linearly with N. As shown in Figure 4.3 (c), the run time of the DFL algorithm grows quasi-squarely with n, and a BLN with 6000 genes can correctly be found in a modest number of hours.

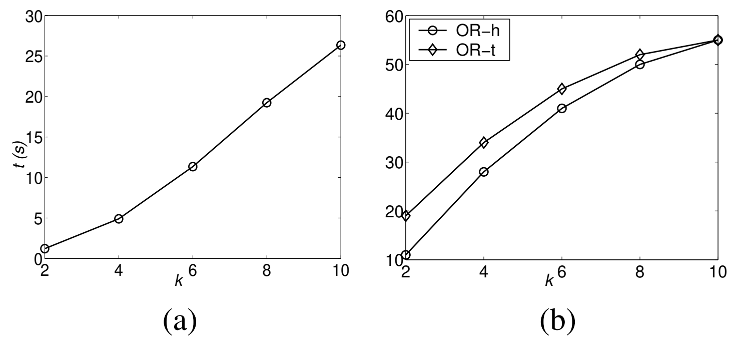


Figure 4.4: The efficiency of the DFL algorithm for the unbounded OR data sets. The values shown are the average of 20 OR data sets. (a) The run time, t (vertical axis, shown in seconds), of the DFL algorithm to infer the unbounded BLNs. (b) The number of the subsets checked by the DFL algorithm for learning one OR Boolean function. The curves marked with circles and diamonds are for OR-h and OR-t data sets respectively.

Complexity for Unbounded OR/AND Boolean Networks

To examine the complexity of the DFL algorithm for unbounded OR/AND BLNs, we generate 20 OR data sets (10 OR-h and 10 OR-t) of $N = 10000$, $n = 10$, and the k is chosen from 2 to 10. Then, we apply the DFL algorithm to these data sets. In all experiments of this section, the expected cardinality K and ϵ of the DFL algorithm is set to 10 and 0 respectively, since it is assumed that the DFL algorithm does not know the indegree of BLNs in prior.

The DFL algorithm successfully finds the original BLNs for all data sets. The average run times of the DFL algorithm for these data sets are shown in Figure 4.4 (a). The number of the subsets checked by the DFL algorithm for learning one OR Boolean function is shown in Figure 4.4 (b).

As shown in Figure 4.4 (a), the run time of the DFL algorithm grows linearly with k

for the unbounded OR data sets. In Figure 4.4 (b), the number of subsets checked by the DFL algorithm is exactly $\sum_{i=0}^{k-1}(n-i)$ for the unbounded OR data sets. In other words, the complexity of the DFL algorithm is $O((N+\log n)\cdot n^3)$ in the worst case for learning the unbounded OR BLNs given enough samples. The DFL algorithm checked a lightly more subsets for the OR-t data sets than for the OR-h data sets, when $k < 10$. This is due to the fact that the DFL algorithm examines different subsets in the kth layer of $\Delta Tree$ with lexicographic order, as introduced in Section 4.6.1.

Complexity for General Boolean Networks

In all experiments of this section, the expected cardinality K and ϵ of the DFL algorithm is set to k of the generation BLNs and 0 respectively. To examine the complexity of the DFL algorithm for random BLNs, we examine the random BLNs of $k=2$ and $k=3$.

First, we generate 16 RANDOM-h and 16 RANDOM-t data sets with $k=2$, $n=100$ and $N=100$, so that each of the 16 RANDOM-h and the 16 RANDOM-t data sets is for one of the 2^{2^2} possible Boolean functions of indegree 2. Then, we run the DFL algorithm on these data sets. We find that the DFL algorithm can correctly find the original BLNs in about 1 second after checking $O(k\cdot n)$ subsets for each $X_i^{'}$ except two cases, where the Boolean functions for $X_i^{'}$ are the XOR function and the inversion of the XOR function. As discussed in Section 4.5, the DFL algorithm may be inefficiently for inferring these two functions. In our experiments, the DFL algorithm still correctly finds the original BLNs for these two special cases in tens of seconds after checking thousands of subsets, which is less than n^2. Hence, the worst complexity of the DFL algorithm happens with 1/8 frequency for inferring random Boolean functions with indegree of 2.

Next, we generate 200 data sets, 100 RANDOM-h and 100 RANDOM-t data sets, with Boolean functions of indegree $k=3$ randomly chosen from 2^{2^3} functions. The DFL algorithm counts the checked subsets for inferring one Boolean function, denoted with m.

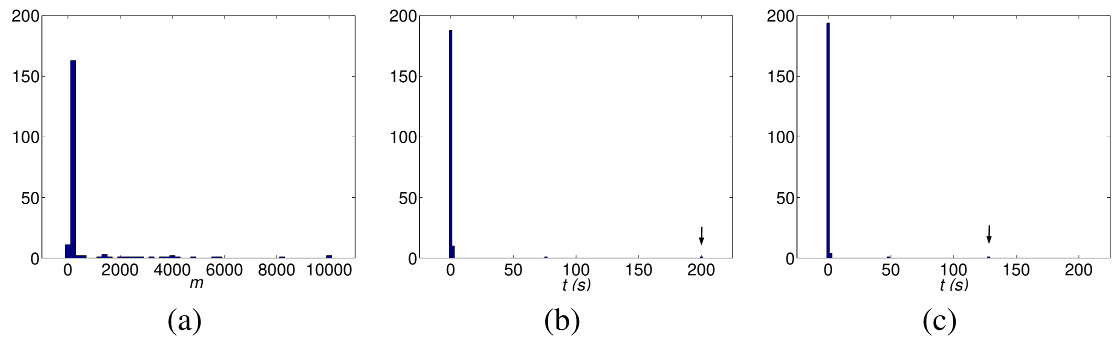


Figure 4.5: The histograms of the number of subsets checked, m, and run time of the DFL algorithm for learning one Boolean function in RANDOM data sets, when $n = 100$, $k = 3$ and $N = 200$. For part (b) and (c), the cases pointed by arrows are the worst ones. (a) The histogram of m without using redundancy matrix $\mathbb{R}$. (b) The histogram of run time, t (horizontal axis, shown in seconds). (c) The histogram of run time after using the redundancy matrix $\mathbb{R}$ introduced in Section 2.8.2.

The histogram of m is shown in Figure 4.5 (a). The run times for these data sets are shown in Figure 4.5 (b).

From Figure 4.5 (a), it is shown that the original Boolean function for X'_i can be found after checking $O(k \cdot n)$ subsets in 178 of out the 200 random functions. The complexity of the DFL algorithm will become $O(k \cdot (N + \log n) \cdot n^2)$ for reconstructing the BLNs for the 178 random functions. The corresponding run times of the 178 cases are only a few seconds, as demonstrated in Figure 4.5 (b). As shown in Figure 4.5 (a), for 20 out of the remaining 22 cases, the original Boolean function for X'_i can be found after checking several thousands or less than n^2 subsets of $\mathbf{V}$. The last two cases are learned after the DFL algorithm checked less than $n^3 = 10^6$ subsets of $\mathbf{V}$. The last two cases are generated with special Boolean functions, similar to $X'_i = X_1 \cdot \neg X_2 \cdot \neg X_3 + \neg X_1 \cdot X_2 \cdot X_3$. In these Boolean functions, $I(X_{ij}; X'_i)$ is zero, which makes the DFL algorithm be more computationally complex than for other data sets. In summary, the worst time complexity of the DFL algorithm, $O(N \cdot n^k)$, happens with about $2/200 = 1\%$ frequency for inferring random Boolean functions with indegree of 3.

For the 200 RANDOM data sets, the DFL algorithm finds the correct BLNs in 196 cases, and finds 2/3 correct edges of the original BLNs in the remaining 4 cases. Hence, the average sensitivity of the DFL algorithm is $596/600 \approx 99.3\%$ for inferring general BLNs from noiseless data sets.

From the experimental results for general BLNs, it is known that the DFL algorithm can find most Boolean functions with k inputs in $O(k \cdot (N + \log n) \cdot n)$ time.

To prove the usefulness of the redundancy matrix $\mathbb{R}$ introduced in Section 2.8.2, we also perform the same experiments on these 200 RANDOM data sets after deploying the redundancy matrix in the DFL algorithm. Figure 4.5 (c) demonstrates that the run time of the worst case has been reduced from 203 to 127 seconds, which equals to a reduction of 37%. This is slightly smaller than the 50% reduction analyzed in Section 2.8.2. We attribute this to the access and exchange of the memory used by $\mathbb{R}$.

4.6.3 Experiments for Sensitivity

As discussed in Section 4.3, the sensitivity of the DFL algorithm is affected by sample size and noisy level of data sets. We perform experiments to examine the sensitivity of the DFL algorithm for different sample size and noise levels of data sets.

Experiments of Small Data Sets

From Theorem 3.2.1, it is known that the sufficient sample size for inferring BLNs is related to the indegree k and the number of variables n in the networks. Therefore, we apply the

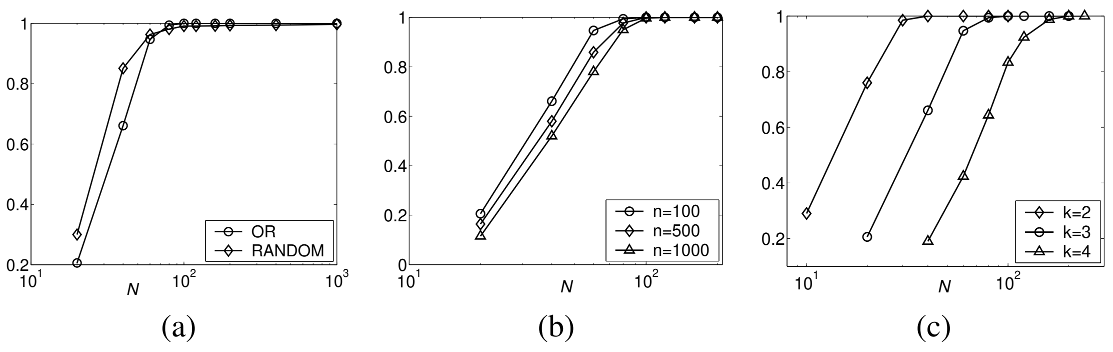


Figure 4.6: The sensitivity of the DFL algorithm vs sample size N. The values shown are the average of 200 data sets. (a) The sensitivity vs N for OR and RANDOM data sets, when $n = 100$, $k = 3$. The curves marked with circles and diamonds are for OR and RANDOM data sets respectively. (b) The sensitivity vs N for OR data sets, when $n = 100, 500, 1000$, and $k = 3$. The curves marked with circles, diamonds and triangles are for data sets of $n = 100, 500$ and 1000 respectively. (c) The sensitivity vs N for OR data sets, when $k = 2, 3, 4$, and $n = 100$. The curves marked with diamonds, circles and triangles are for data sets of $k = 2, 3$ and 4 respectively.

DFL algorithm to 200 OR (100 OR-h and 100 OR-t) and 200 RANDOM (100 RANDOM-h and 100 RANDOM-t) data sets with $k = 3$, $n = 100$ with various N. Then, we apply the DFL algorithm to 200 OR (100 OR-h and 100 OR-t) data sets, where $k = 3$, $n = 100, 500, 1000$ and various N. Finally, we apply the DFL algorithm to 200 OR (100 OR-h and 100 OR-t) data sets, where $k = 2, 3, 4$, $n = 100$ and various N. The relation between the sensitivity of the DFL algorithm and N are shown in Figure 4.6. In all experiments of this section, the expected cardinality K and ϵ of the DFL algorithm is set to k of the generation BLNs and 0 respectively.

From Figure 4.6, it is shown that the sensitivity of the DFL algorithm grows approximately linearly with the logarithmic value of N, but becomes 1 after a certain N value except the RANDOM data sets in part (a). For the RANDOM data sets, the sensitivity of

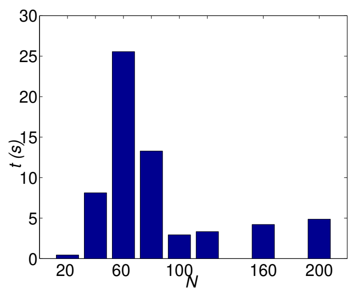


Figure 4.7: The run time, t (vertical axis, shown in seconds), of the DFL algorithm for small OR data sets, where $n = 100$ and $k = 3$. The values shown are average of 200 data sets.

the DFL algorithm has increases to 99.3% when $N = 200$, and further to 99.7% when $N = 1000$. That means, if the data is enough, the DFL algorithm can correctly identify the original OR BLNs, and correctly find the original RANDOM BLNs with very high probability.

In Figure 4.6 (a), it is shown that the sensitivity of the DFL algorithm for RANDOM data sets is slightly higher than that for the OR data sets, when the sample size N is smaller than a certain value, but becomes lower after this value. As shown in Figure 4.6 (b), the sensitivity of the DFL algorithm for various n shows a small decrease when n increases and N is the same. From Figure 4.6 (c), it is shown that the sensitivity for various k shows a large decrease when k increases and N is the same. This is due to the different effect of k and n in deciding the enough sample size. In Theorem 3.2.1, the enough sample size N grows exponentially with k but linearly with $\log n$.

In this section, we find that when the sample size is small, the DFL algorithm may use much more time than $O(k \cdot (N + \log n) \cdot n^2)$. For example, the run time of the DFL algorithm for the 200 small OR data sets of $n = 100$ and $k = 3$ used in this section is shown in Figure 4.7. As shown in Figure 4.7, the complexity of the DFL algorithm is bad

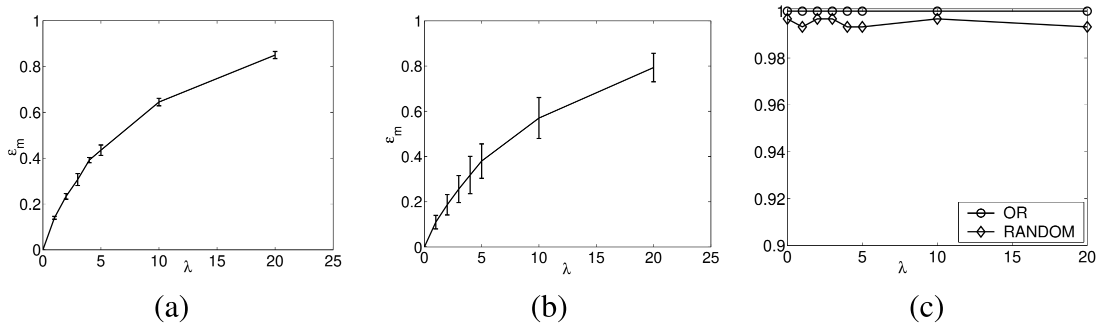


Figure 4.8: The setting and performance of the DFL algorithm for noisy data sets, whose $k = 3$, $n = 100$ and $N = 1000$. The values shown are the average for 100 data sets. (a) The minimum ϵ value, ϵ_m, vs noise level λ (%) in the OR data sets. (b) The minimum ϵ value, ϵ_m, vs noise level λ (%) in the RANDOM data sets. (c) The sensitivity of the DFL algorithm vs λ for noisy OR/RANDOM data sets. The curves marked with circle and diamond are for OR and RANDOM data sets respectively.

when the sample size N falls into the region from 20 to 100, but resumes linear growth after N is bigger than 100. This is due to the fact that the DFL algorithm requires enough learning samples to find OR/AND BLNs in $O(k \cdot (N + \log n) \cdot n^2)$ time, as discussed in Section 4.4.2.

Experiments of Noisy Data Sets

The noisy data sets are generated by randomly selecting λ percent samples, then inverting their output values $\mathbf{v}'$. To examine the performance of the DFL algorithm when dealing with noisy data sets, we generate 100 OR/RANDOM data sets (50 "head" and 50 "tail" data sets) with different noise levels λ from 1% to 20%. The expected cardinality K of the DFL algorithm is still set to k of the generation BLNs in experiments of this section.

Then, we run the DFL algorithm by choosing different ϵ values for these data sets with the restricted learning method introduced in Section 2.6.2. The relation of ϵ_m and λ for OR/RANDOM data sets is shown in Figure 4.8 (a)/(b), where the ϵ_m value is the average of the 10 sets for each λ value. The relation of ϵ_m and λ for AND data sets is similar to that for OR data sets. As demonstrated in Figure 4.8 (a), the ϵ_m is increasing with the increase of λ value. This means that the missing part of the entropy of Y, as demonstrated by the shaded region of Figure 2.9 (b), tends to increase when there is more and more noise in the data sets. The variances of the ϵ_m in RANDOM data sets are bigger than those in OR data sets, as demonstrated in Figure 4.8 (a) and (b). This is reasonable, since the $I(\mathbf{Pa}(X_i^{'}); X_i^{'})$ and $H(X_i^{'})$ are more diverse in RANDOM data sets than those in OR data sets.

The sensitivity of the DFL algorithm maintains 1 for all noisy OR data sets, as demonstrated in Figure 4.8 (c). In other words, for all noisy OR data sets, the DFL algorithm can correctly find the original BLNs. As shown in Figure 4.8 (c), the sensitivity of the DFL algorithm for RANDOM data sets does not decrease significantly even when λ increases to 20%. The DFL algorithm correctly finds the original BLNs for over 98% noisy RANDOM data sets, and find 2/3 correct edges of the BLNs for the rest RANDOM data sets.

The DFL algorithm also correctly finds the truth table with the method introduced in Section 4.2.2 for all noisy data sets when the sensitivity is one. For instance, the obtained rules for one RANDOM-h data sets are shown in Table 4.3.

As shown in Table 4.3, the left rules have significantly larger counts than their counterparts on the right side. By using the method introduced in Section 4.2.2, the rules on the right side will be eliminated and the truth table for this data sets is the 8 rules on the left side. By applying the Karnaugh-map to the obtained truth table, the obtained Boolean function is actually the $X_i^{'} = \neg X_1 \cdot X_2 + X_1 \cdot \neg X_2 \cdot X_3$ in the origination Boolean network. Up till now, the original Boolean network has successfully and completely been identified by the DFL algorithm from this data sets with 10% noise. In addition, the total count value of the rules on the right side is 100 which is exactly $10\% \times 1000$. This means that the rules

Table 4.3: The obtained Boolean rules from one noisy RANDOM-h data set with 1000 samples and 10% noise. In the original BLN, $X_i^{'} = \neg X_1 \cdot X_2 + X_1 \cdot \neg X_2 \cdot X_3$.

Rules From Noiseless Samples					Rules From Noisy Samples				
X_1	X_2	X_3	$X_i^{'}$	Count	X_1	X_2	X_3	$X_i^{'}$	Count
0	0	0	0	137	0	0	0	1	10
0	0	1	0	123	0	0	1	1	8
0	1	0	1	97	0	1	0	0	10
0	1	1	1	105	0	1	1	0	12
1	0	0	0	98	1	0	0	1	19
1	0	1	1	118	1	0	1	0	15
1	1	0	0	107	1	1	0	1	12
1	1	1	0	115	1	1	1	1	14
Total Count				900					100

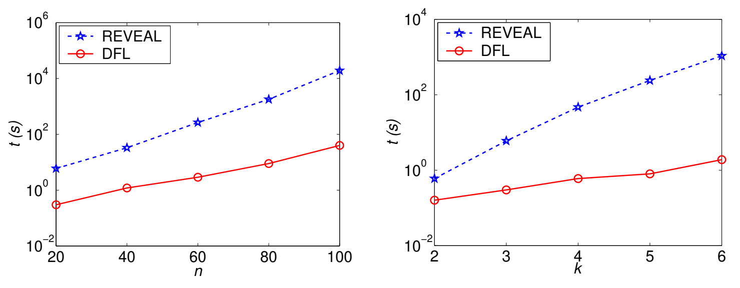
on the right side are coming from the 10% noise in the data set.

The run times of the DFL algorithm do not change severely for data sets with different noise levels. In other words, the DFL algorithm is still efficient when the data sets are noisy.

4.6.4 Comparisons of Time Complexity with Existing Methods

In this section, we will compare the time complexity of the DFL algorithm with the REVEAL algorithm [113], which is a benchmark algorithm for reconstructing BLNs from state transition pairs. As introduced in Section 4.1, the time complexity of the REVEAL algorithm is $O(N \cdot n^{k+1})$ [184]. We have also implemented the REVEAL algorithm with the Java programming language in the Discrete Function Learner software. We will use the noiseless OR data sets to compare the run time of the two algorithms. For all experiments, the expected cardinality K is set to the real number of input variables k and ϵ to 0.

In this section, we use Equation 3.2 to determine the sample size N. Therefore, the time complexity of the DFL algorithm becomes $O(k \cdot n \cdot (N + \log n)) = O(k \cdot n \cdot (2^k + k \log n))$ for learning one Boolean function and $O(k \cdot n^2 \cdot (2^k + k \log n))$ for learning a BLN. Thus,



(a) (b)

Figure 4.9: Comparison of the run times of the DFL algorithm and the REVEAL algorithm. In all experiments, both the DFL algorithm and the REVEAL algorithm can correctly find the original model. (a) when n increases alone, (b) when k increases alone.

the run time of the DFL algorithm will grow in an exponential way with k if the sample size N is calculated with Equation 3.2.

Experiments when k is fixed, n increases

In this section the indegree of each gene k is fixed to 3, and the number of transition pairs is calculated with Equation 3.2 where c is 3. The number of genes goes from 20 to 100.

The experiment results are shown in Figure 4.9 (a), where the time is the average value of 5 different OR-h and 5 different OR-t data sets. The run time values are shown in logarithmic values. In all experiments of this kind, both the DFL algorithm and the REVEAL algorithm can find the original BLNs correctly. However, the DFL algorithm is significantly faster than the REVEAL algorithm as shown in Figure 4.9 (a).

Table 4.4: The success ratio (%) of the GREEDY1 algorithm [9] for noisy data sets of general BLNs, where the indegree $k = 3$ and $n = 1000$ (Table courtesy of Akutsu et al. [9]).

	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$
$N = 500$	68%	71%	29%
$N = 1000$	71%	72%	60%
$N = 2000$	77%	73%	76%

Experiments when n is fixed, k increases

In this section, the number of genes n is fixed to 20, and k is increased from 2 to 6. Similar to the results of the prior section, both the DFL algorithm and the REVEAL algorithm can find the original BLNs correctly. However, the run times of the DFL algorithm are significantly smaller than those of the REVEAL algorithm in all cases, as shown in Figure 4.9 (b), where the time is also the average value of 5 different OR-h and 5 different OR-t data sets. The run time values are also shown in logarithmic values.

In Figure 4.9 (b), the run time of the DFL algorithm grows approximately linearly in logarithmic coordinate, which means an exponential growth in ordinary coordinate.

4.6.5 Comparisons of Robustness to Noisy Data Sets with Existing Methods

In [9], the GREEDY1 algorithm was used to learn general (RANDOM) BLNs from noisy data sets, as shown in Table 4.4. From Table 4.4, it can be seen that when the sample size is small, the performance of the GREEDY1 algorithm decreases with the increase of the noise level λ. When the sample size is large enough, the GREEDY1 algorithm finds the original BLNs with a success ratio of about 75%. In comparison, as shown in Figure 4.8 (c), the DFL algorithm correctly finds the original RANDOM BLNs with a much higher success ratio of 98%, where $n = 100$, $N = 1000$ and $0 \leq \lambda \leq 0.2$. The experimental results of the DFL algorithm for $n = 1000$ and $N = 2000$ should be similar to that for $n = 100$ and $N = 1000$. Because according to Theorem 3.2.1, the sample size for successfully finding the original BLNs has lower bound of $\Omega(2^k + k \log_2 n)$. This means that the samples size N should grows in logarithmic value of n. Hence, when n changes from 100 to 1000, the corresponding sample size N should be changed with $\log_2 1000 / \log_2 100 = 1.5$ fold. The original sample size becomes $N = 1000 \times 1.5 = 1500$, which is smaller than 2000. In other words, $N = 2000$ is large enough.

The above comparisons demonstrate that the DFL algorithm has a better robustness to noise than the GREEDY1 algorithm. It is difficult to compare the robustness of the DFL algorithm to those of other methods in the literature, such as methods in [6, 95, 105, 113, 119], since these methods cannot handle noisy data sets.

4.7 Related Models

In this section, we discuss the related models, the Probabilistic Boolean Networks (PBNs) and Dynamic Bayesian Networks (DBNs). We will show that the DFL algorithm combined with the ϵ value method can be used to infer PBNs and DBNs. More discussion about the relationships of BLNs, PBNs and DBNs is given in [127, 155].

4.7.1 Probabilistic Boolean Networks

To cope with the uncertainty, Shmulevich *et al.* [155] introduced the PBNs. The basic idea is to extend the BLN to accommodate more than one possible functions for each node [155]. A PBN $G(\mathbf{V}, \mathbf{F})$ is defined by a set of variables $\mathbf{V} = \{X_1, \ldots, X_n\}$ and a function matrix, $\mathbf{F} = \{F_1, \ldots, F_n\}$, where F_i is defined by Equation 4.8.

$$F_i = \{f_j^{(i)} : j = 1, \ldots, l(i)\} \tag{4.8}$$

where each $f_j^{(i)}$ is a possible function determining the value of gene X_i and $l_{(i)}$ is the number of possible functions for gene X_i. The functions $f_j^{(i)}$ is referred to as predictors, since the process of inferring these functions from measurements or equivalently, of producing a minimum-error estimate of the value of a gene at the next time point, is known as prediction in estimation theory [155].

The major difference between the PBNs and standard BLNs lies in the $\mathbf{F}$. As shown in Table 4.3, the noisy rules learned from noisy data sets contain two truth table of the three input variables, shown on the left and right side respectively. In BLNs, the functional relations are deterministic, hence only the rules on left side are used as the estimated truth table of X_i'.

However, if we consider this situation from another aspect, the rules on the right side can also be seen as alternative gene expression patterns. That is to say, we can consider building a PBN model with the rules in Table 4.3. Formally, we let $F_i = \{f_1, f_2\}$, where $f_1 = \neg X_1 \cdot X_2 + X_1 \cdot \neg X_2 \cdot X_3$ and $f_2 = X_1 \cdot X_2 + \neg X_1 \cdot \neg X_2 + \neg X_2 \cdot \neg X_3$. The probability that f_i is selected is estimated by the total counts of rules, i.e., $p(f_1) = 900/1000 = 0.9$ and $p(f_2) = 0.1$. Hence, the DFL algorithm combined with the ϵ can be used to build PBNs efficiently.

4.7.2 Dynamic Bayesian Networks

Bayesian networks have been used to model GRNs [73, 75, 89, 150]. However, since directed circles are not allowed in standard Bayesian networks, Murphy and Mian [127] and Ong *et al.* [129] used the DBNs to model GRNs. As shown in Theorem 2.2.3, X_i' and $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{Pa}(X_i')$ are independent given $\mathbf{Pa}(X_i')$ in BLNs. Hence, the BLN is a special case of DBN, where the relations between variables are deterministic.

Similar to the discussion in Section 4.7.1, the DFL algorithm combined with the ϵ value method can also be used to learn DBNs, even the relations between variables are probabilistic. The CPT of variables can be estimated with the frequencies of rules, for the same instances of $\mathbf{Pa}(X_i^{'})$. For the example in Table 4.3, for $(0,0,0)$ of $\{X_1, X_2, X_3\}$, we can have $p(0|000) = \frac{137}{137+10} = 0.932$ and $p(1|000) = 0.068$ as estimation of $P(X_i^{'}|\mathbf{Pa}(X_i^{'}))$ based on Table 4.3.

There is one limitation when learning DBNs with the DFL algorithm. Recall that it is assumed that $\forall 1 \leq i,j \leq n$, $X_i(t)$, $X_j(t)$ are independent in Section 3.2.1. That means in the DBNs learned with the DFL algorithm, the $\mathbf{Pa}(X_i^{'})$ are all from the prior time step.

4.8 Conclusions

The main contributions of this chapter are three-fold.

First, we prove the DFL algorithm can learn OR/AND Boolean functions with $O(k \cdot (N + \log n) \cdot n)$ time complexity in Theorem 4.4.4. In other words, the **Problem 2** introduced in Section 4.1 can be solved with the DFL algorithm, for OR/AND Boolean functions. For general Boolean functions, the experimental results show that the DFL algorithm still maintains its complexity of $O(k \cdot (N + \log n) \cdot n)$ in most cases. Sample distribution is another point to be noted. Although the learning samples of this study are drawn with the Discrete Uniform Distribution, the requirement of enough samples in Theorem 4.4.4 is distribution free, since Theorem 2.1.4 is correct regardless of sample distribution.

Second, the DFL algorithm still maintains $O(k \cdot (N + \log n) \cdot n)$ complexity for unbounded OR/AND Boolean functions given enough samples, as discussed in Section 4.4. However, based on Theorem 3.2.1, the sample size for the unbounded OR/AND Boolean functions is NP-hard, which makes the the inference problem of unbounded OR/AND functions, i.e., **Problem 1** in Section 4.1, still be NP-hard.

Third, we demonstrate that the DFL algorithm combined with the ϵ value method can find the original BLNs from noisy data sets, even when there have been 20% noisy instances in the learning data sets. These results suggest that the DFL algorithm is effective when applied to real gene expression data sets since gene expression data sets are often noisy. We also discuss that relationship between the BLNs and the PBNs, DBNs. We show that the DFL algorithm combined with the ϵ value method is also useful in learning PBNs and DBNs.

We introduce the sensitivity, which is affected by sample size N and noise level of data sets, as the criterion to evaluate the performance of the DFL algorithm. Experimental results show that the sensitivity of the DFL algorithm is more severely affected by the sample size N, than the noise level of data sets, which suggest that it is more advisable to use a large data sets with reasonable noise level than to use a small data sets with small noise level for inferring GRN models from gene expression profiles.

The major limitation of BLNs is that the expression levels of genes are simply represented with two states, ON and OFF. However, BLNs are good initial points for finding more realistic models of GRNs. To understand the operation of whole systems of regulatory interactions, computational models are essential: for organizing experimental extensions and tests at each stage of construction of the model, to check on consistency, and to integrate experimental results with the current network architecture by means of simulation [56]. BLNs are simple and very suitable for large scale simulation. BLNs can be used to reconstruct initial models of GRNs, then further biological experiments can be carried to refine the initial GRN models, as done by Yuh *et al.* [179, 180].

Since Boolean function learning algorithms have been used to solve many problems [29, 35, 66, 94, 114, 120, 121, 123, 142], the DFL algorithm can also find its applications in other fields, such as classification [183, 185, 188, 189, 192], pattern recognition, functional dependencies retrieving and association rules retrieving.

Chapter 5

Solving Classification Problems

CLASSIFICATION is one of the fundamental problems in artificial intelligence. The aim of classification is to form a classification function or a model from training data sets, then use it to predict new samples, as defined in Definition 2.2.1.

In this chapter, we will apply the DFL algorithm to solving classification problems. The 24 data sets in Table 5.1 are used to validate the performances of the DFL algorithm. The DFL algorithm shows good prediction performances in terms of accuracy, model complexity, and run time. We also compare the results of the DFL algorithm with those from other well-known methods and in literature.

This chapter is organized as follows. In Section 5.1, we describe the problem of cancer classification based on biological data. In Section 5.2, we describe an entropy based discretization method. In Section 5.3, we show the results of the DFL algorithm and other classification algorithms. In Section 5.4, we evaluate the results of the biological data sets used in the experiments. In Section 5.5, we summarize this chapter.

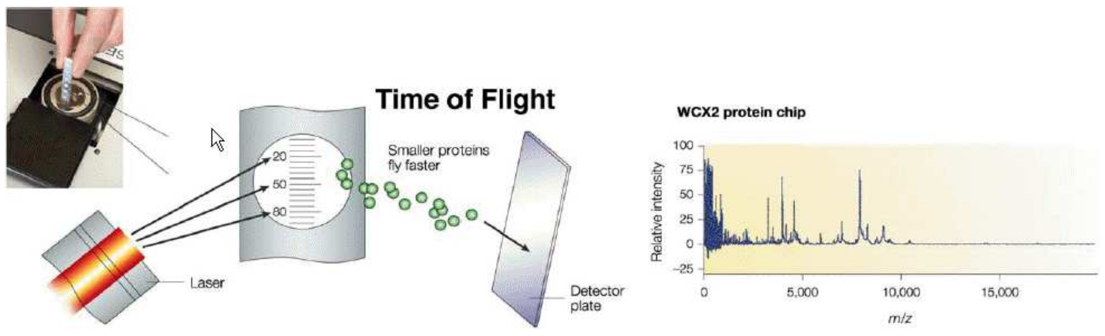


Figure 5.1: Overview of Surface-Enhanced Laser Desorption/Ionization (SELDI) type mass spectrometry technology. (Courtesy of Petricoin *et al.*, 2004 [135])

5.1 Cancer Classification with Biological Data

In this section, we first briefly introduce the technology of Surface-Enhanced Laser Desorption/Ionization (SELDI) type mass spectrometry. Then, we describe the motivation and challenge of cancer classification problem based on biological data sets.

5.1.1 A Brief Introduction to SELDI Technology

As shown in Figure 5.1, this type of Matrix-Assisted Laser Desorption/Ionization (MALDI) Time-Of-Flight (TOF) mass is proving very useful in high-throughput proteomic fingerprinting of serum, body fluids, and tissue. Using an automated methodology, the biological material is processed with a robotic sample dispenser, where one microliter of raw serum is applied to the surface of a protein-binding chip. Some laboratories prefractionate their samples beforehand, whereas others perform complex analysis for optimizing binding by diluting into a myriad of pH and salt permutations. Based on the underlying SELDI chip chemistry and the pH and buffer used, only a small subset of the proteins in the entire sample's proteomic and metabonemic repertoire will bind to the surface of the chip. The bound molecules are then treated with an organic acid MALDI matrix and washed and dried. The chip, which contains multiple patient samples, is inserted into a vacuum chamber where it is irradiated with a laser. The laser desorbs the adherent proteins, causing them to be launched as ions. The TOF of the ion prior to detection by an electrode is a measure of the mass to charge (*M/Z*) value of the ion. The ion spectra can be analyzed by computer-assisted pattern recognition tools that classify a subset of the spectra by their characteristic patterns of relative intensity.

Using this method, one microliter of raw unfractionated serum from a patient is analyzed by SELDI-TOF to create a proteomic signature of the serum (right part of Figure 5.1). This serum proteomic bar-code is comprised of potentially tens of thousands of protein signatures, which then require extensible data mining operations for analysis.

5.1.2 Cancer Classification with Biological Data

In this section, we will briefly describe the problem of cancer classification based on biological data, like microarray gene expression profiles [81] and proteomic profiles [130].

The challenge of cancer treatment has been to detect cancer as early as possible and/or to target specific therapies to pathogenetically distinct cancer types [81]. Therefore, early detection of cancer and/or its subtypes is central to apply accurate therapies and improve the cure and survival rate. With recent development of microarray and mass spectrometry technology, it is possible to detect cancer and/or its subtypes in the early stages [12, 18, 81, 96, 134, 136, 173]. To diagnose early stage cancer, new technologies are introduced to analyze the expression pattern or the proteomic patterns in serum, since the pathological changes within the an organ might be reflected in gene expression levels [81] or serum protein profiles [134].

How to use these biological data to build accurate and meaningful classification models introduces another a big challenge, since biological data sets are often high-dimensional but the sample size is relatively small [92, 112].

There have been many successful stories of use biological data to identify tumor and/or its subtypes in the literature. However, most methods use complex models, like the Support Vector Machines and k-Nearest-Neighbors, to perform classification. According to the principle of Occam's razor, simple models are preferable than complex ones, if they can produce comparable prediction performances. Hence, the complex models are not optimal in terms of model complexity and very hard to understand, although accurate in practice. Some examples of performing cancer classification based on gene expression profiles include classification of leukemia subtype by Golub *et at.* [81], colon tumor by Alon *et al.* [12], multi-class leukemia by Armstrong *et al.* [18], central nervous system (CNS) tumor by Pomeroy *et al.* [138], breast cancer by van't Veer *et al.* [167], diffused large B-cell lymphoma (DLBCL) by Alizadeh *et al.* [11] and Shipp *et al.* [154], lung cancer by Beer *et al.* [25], Gordon *et al.* [82], Bhattacharjee *et al.* [28] and Wigle *et al.* [171]. Some examples of performing cancer classification based on proteomic profiles include classification of ovarian cancer by Cohen *et al.* [46], Petricoin *et al.* [134] and Wang *et al.* [170], prostate cancer by Adam *et al.* [4], Petricoin *et al.* [135] and Gretzer *et al.* [84], breast cancer by Becker *et al.* [24].

In Section 4.6, we will use the DFL algorithm to perform cancer classification based on microarray gene expression profiles [18, 81, 154] and on proteomic profiles [134].

5.2 A Supervised Discretization Method

We use a widely used supervised discretization method introduced by Fayyad and Irani [67] to discretize the continuous features. Following the notation in [62, 67], we will briefly summarize the discretization algorithm. Let partition boundary T separate set S into S_1

and S_2. Let there be k classes $C_1, \cdots, C_k$. Let $P(C_i, S_j)$ be the proportion of examples in S_j that have class value C_j. The class entropy of a subset S_j, $j = 1, 2$ is defined as:

$$Ent(S_j) = -\sum_{i=1}^{k} P(C_j, S_j) log P(C_j, S_j).$$

Let S_1 and S_2 be induced with the boundary T of attribute A, then the class information entropy of the partition is given by:

$$E(A, T; S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2).$$

For a given attribute A, the boundary T_{min} is chosen to minimize $E(A, T; S)$ as a binary discretization boundary. This method is recursively used to the two partitions induced by T_{min}, until some stop criteria is reached, therefore creating multiple intervals on the attribute A.

The Minimum Description Length principle is used as the stop criterion of the partitioning by Fayyad and Irani [67]. The recursive partitioning within a set of values S stops iff

$$Gain(A, T; S) < \frac{log_c(N-1)}{N} + \frac{\delta(A, T; S)}{N}$$

where N is the number of instances in the set S, $Gain(A, T; S) = Ent(S) - E(A, T; S)$, $\delta(A, T; S) = log_2(3^k - 2) - [k \cdot Ent(S) - k_1 \cdot Ent(S_1) - k_2 \cdot Ent(S_2)]$, and k_i is the number of class labels represented in set S_i.

After the discretization process, a substantial number of features, which are not contributing to the class distinction, are assigned with only one value. Meanwhile, the remaining discriminatory features are assigned with limited value intervals. For example in our experiments, the *Zyxin* gene in the ALL data set is one of the genes most highly correlated with the ALL-AML class distinction [81]. In the discretization process, the expression values of the *Zyxin* gene are discretized into two intervals, $(-\infty - 994]$ and $(994 - \infty)$. This discretization method has been implemented by the *Weka*[1] software [69, 172].

5.3 Results

First, we briefly introduce the benchmark data sets used in this section. Then, we show the performance of the DFL algorithm on these data sets.

5.3.1 Data Sets

We use the 24 data sets summarized in Table 5.1 to compare the DFL algorithm with other classification and feature selection methods. Twenty out of the twenty four data sets are classic machine learning data sets from UCI machine learning repository [31]. The detailed description of these data sets are available at the supplementary wetsite of the thesis. We arrange the data sets in the ascending order of the number of features.

In all the data sets used, the missing values are dealt as an independent state marked with "?". Therefore, it is possible that the learned classifiers also have some missing values marked with "?".

For data sets with continuous features, we discretize their continuous features with the discretization algorithm introduced in Section 5.2. The discretization is carried out in such a way that the training data set is first discretized. Then the testing data set is discretized according to the cutting points of variables determined in the training data set. For the Breast data set, the attributes are numerical with some limited integers. Therefore, we do not apply the pre-discredization method to this data set.

[1]The *Weka* software, available at http://www.cs.waikato.ac.nz/∼ml/weka/, is written with the Java language and is an open source software issued under the GNU General Public License.

Table 5.1: The benchmark data sets used in the experiments for comparison.

	Dataset	Feature NO.[1]	Classes	Training	Testing	Missing	Source
1	Lenses	4 (4D)	3	24	LOO[2]	0	UCI
2	Iris	4 (4N)	3	100	50	0	UCI
3	Monk1	6 (D)	2	124	432	0	UCI
4	Monk2	6 (D)	2	169	432	0	UCI
5	Monk3	6 (D)	2	122	432	0	UCI
6	LED	7 (7D)	10	2000	1000	0	UCI
7	Nursery	8 (8D)	5	12960	CV10[3]	0	UCI
8	Breast	9 (9N)	2	699	CV10	16	UCI
9	Wine	13 (13N)	3	119	59	0	UCI
10	Credit	15 (9D6N)	2	460	230	67	UCI
11	Vote	16 (16D)	2	435	CV10	392	UCI
12	Zoo	16 (D)	7	101	LOO	0	UCI
13	ImgSeg	19 (N)	7	210	2100	0	UCI
14	Mushroom	22 (22D)	2	8124	CV10	2480	UCI
15	LED+17	24 (D)	10	2000	1000	0	UCI
16	Ionosphere	34 (34N)	2	234	117	0	UCI
17	Chess	36 (36D)	2	2130	1066	0	UCI
18	Anneal	38 (29D9N)	6	798	100	22175	UCI
19	Lung	56 (56D)	3	32	LOO	0	UCI
20	Ad	1558 (1555D3N)	2	2186	1093	2729	UCI
21	ALL	7129 (7129N)	2	38	34	0	[81]
22	DLBCL	7129 (7129N)	2	55	22	0	[154]
23	MLL	12582 (12582N)	3	57	15	0	[18]
24	Ovarian	15154 (15154N)	2	169	84	0	[134]

[1] The number does not include the class attribute. "D" and "N" represents discrete and numerical attributes respectively. [2] LOO and [3] CV10 stands for leave-one-out and 10 fold cross validation respectively.

5.3.2 Comparison with Other Classification Methods

In this research, we use the restricted learning method introduced in Section 2.6.2 to obtain optimal models for the DFL algorithm, with the searching scope of the ϵ from 0 to 0.8. As discussed in Section 2.6.1, the expected cardinality K is set to a small integer, like 20, for the high-dimensional data sets and to n for the low-dimensional data sets. The optimal settings are shown in Table D.1. The optimal classifiers and prediction details of the DFL algorithm are available at the supplementary website of this thesis.

The optimal ϵ values are obtained from cross validation on the training data sets. Since

Table 5.2: The comparison of accuracies for the DFL algorithm and other well-known classification methods. The unit is percent. The accuracies for those data sets with numerical attributes are for discretized/numerial data sets. The best result for each data set is shown in bold face.

	Data Set	DFL	C4.5	NB	1NN	kNN[1]	SVM
1	Lenses	75.0	**83.3**	70.8	60.8	70.8	65.4
2	Iris	96.0	94.0/92.0	94.0/94.0	94.0/94.0	92.0/**98.0**	94.0/94.0
3	Monk1	**100.0**	75.7	71.3	78.7	77.8	72.2
4	Monk2	**73.8**	65.0	61.6	**73.8**	62.3	67.1
5	Monk3	**97.2**	**97.2**	**97.2**	82.9	92.8	**97.2**
6	LED	74.9	74.6	75.1	71.0	**75.6**	75.3
7	Nursery	93.1	97.1	90.3	78.6	**98.1**	93.1
8	Breast	95.0	94.7	**97.3**	95.6	95.2	95.8
9	Wine	**98.3**	93.2/93.2	98.3/98.3	98.3/94.9	96.6/93.2	98.3/94.9
10	Credit	**88.3**	84.3/87.8	87.4/76.5	78.7/81.7	86.5/84.8	86.5/87.4
11	Vote	**95.7**	95.3	90.0	92.7	92.8	95.5
12	Zoo	92.8	92.1	94.1	**96.2**	92.1	96.0
13	ImgSeg	90.6	89.7/91.0	88.3/80.1	88.8/**91.9**	86.3/89.8	90.6/87.5
14	Mushroom	**100.0**	**100.0**	95.5	**100.0**	**100.0**	**100.0**
15	LED+17	**75.4**	73.3	74.6	47.3	64.4	74.8
16	Ionosphere	**94.9**	91.5/88.0	92.3/82.1	92.3/88.0	92.3/84.6	**94.9**/87.2
17	Chess	97.4	**99.5**	87.3	90.4	94.2	95.2
18	Anneal	**99.0**	94.0/95.0	97.0/79.0	97.0/98.0	98.0/96.0	94.0/91.0
19	Lung	**62.5**	43.8	53.1	35.7	46.9	37.5
20	Ad	**95.0**	**95.0/95.0**	93.5/93.6	90.5/93.1	92.4/92.6	**95.0/95.0**
21	ALL	**94.1**	91.2/91.2	85.3/88.2	76.5/73.5	82.4/67.7	82.4/85.3
22	DLBCL	**95.5**	**95.5**/81.8	**95.5**/81.8	**95.5**/81.8	**95.5**/81.8	**95.5/95.5**
23	MLL	**100.0**	86.7/80.0	86.7/**100.0**	86.7/80.0	80.0/86.7	**100.0/100.0**
24	Ovarian	98.8	92.9/97.6	76.2/84.5	88.1/90.5	92.9/91.7	98.8/**100.0**
	average	**91.0**	87.5/86.8	85.5/84.0	82.9/82.1	85.9/84.6	87.3/86.8

[1] The k value of the kNN algorithm is set to 5.

the distribution of features and the class attribute may be a slightly different in cross validation, a wider region around the optimal ϵ value found from cross validation is validated in training and testing process. For instance, as shown in Figure 2.10 (a), in the 10-fold cross validation of the LED+17 training data set, the optimal ϵ value of 0.31 is found. Then, the DFL algorithm also reaches its best performance with this ϵ value in the training testing process.

Table 5.3: The summary of prediction performances of different algorithms in Table 5.2.

	Discretized D.S.			Continuous D.S.		
Algo. Pair	win	draw	lose	win	draw	lose
DFL:C4.5	17	4	3	16	4	4
DFL:NB	18	3	3	18	3	3
DFL:1NN	18	4	2	19	2	3
DFL:kNN	19	2	3	19	1	4
DFL:SVM	11	10	3	14	6	4
sum	83	23	14	86	16	18

We use the *Weka* software (version 3.4) to evaluate the performance of other classification methods. Specifically, we compare the DFL algorithm with the C4.5 algorithm by Quinlan [140], the Naive Bayes (NB) algorithm described by Langley *et al.* [106] and Duda *et al.* [63], the 1NN and k-Nearest-Neighbors (kNN) algorithm by Aha *et al.* [5] and the Support Vector Machines (SVM) algorithm by Platt [137]. For the SVM algorithm, the linear kernels are used. All these methods are implemented in the *Weka* software. The accuracies of all compared algorithms are listed in Table 5.2, where the values are the averages of 10 runs.

As shown in Table 5.2, the DFL algorithm performs well for the selected data sets. Especially for the data sets with large number of features, like data sets with index from 14 to 24, the accuracies of the DFL algorithm are more competitive than those from other compared classification methods, which suggests good generality of the DFL algorithm for high-dimensional data sets. For instance, the DFL algorithm correctly finds the seven relevant features of the LED+17 data set without any prior knowledge since the expected cardinality K is specified to 20 (see Figure 2.10), and obtains better prediction accuracies than other well-known algorithms. For another example, the DFL also correctly find the original generation function for the Monk1 data set (details available at the supplementary prediction details) and obtains 100% accuracy that is significantly better than those of other

Table 5.4: The comparison of the DFL algorithm and other methods in literature. The column names k, Acc., Al. and F.S. stand for the number of features in the classifiers, the accuracies, the algorithm used, the feature selection method used respectively. For all columns, NA stands for not available. For all data sets, the training/testing samples are the same as those in Table 5.1.

	DFL		Methods in Literature				
Data Set	k	Acc.	Acc.	k	Al.[2]	F.S.[3]	Literature
Monk1	3	100.0	100.0	6	CN2	NA	p.67, [162]
Monk2	6	73.8	79.6	6	ARQ	NA	p.67, [162]
Monk3	2	97.2	95.6	6	ID3	NA	p.67, [162]
			97.2	6	CC	NA	p.110, [162]
Nursery	5	93.1	99.1	8	ART	NA	[27]
Vote	4	95.7	95.9	16	ART	NA	[27]
Mushroom	4	100.0	98.5	22	ART	NA	[27]
Chess	19	97.4	92.3	36	TAN	NA	[74]
			96.4	NA	BAN	WSE	[98]
ALL	1	94.1	85.3	50	WV	S2N	[81]
			94.1-88.2	1000	SVM	S2N	[77]
			91.2	1	EP	E	[111]
			100.0	42	kNN	MB	[174]
MLL	2	100.0	90.0	40	kNN[1]	S2N	[18]
			80.0	20	C45	χ^2	[110]
			93.3	20	SVM	χ^2	[110]
			93.3	20	kNN	χ^2	[110]
			100.0	20	PCL	χ^2	[110]
			100.0	20	NB	χ^2	[110]

[1] The kNN classifier in [18] misclassified 1 sample out of 10 independent testing samples. [2] For Al. column, the CN2, ARQ, ID3, CC, ART, TAN, BAN, WV, SVM, EP, kNN, C45, PCL and NB represent the CN2 [162], ARQ [162], ID3 [162], Cascade Correlation, ART [27], TAN [74], BAN [98], Weighted-Voting [81], Support Vector Machine, Emerging Pattern [111], k-Nearest-Neighbors, C4.5, Prediction by Collective Likelihoods [110] and Naive Bayes algorithm respectively. [3] For the F.S. column, the WSE, S2N, E, MB and χ^2 are the wrapper subset evaluation method, ranking with the *signal-to-noise* statistic [81], ranking with entropy [67], Markov Blanket [174] and ranking with χ^2-statistic respectively.

algorithms.

We summarize the prediction performances of different classification algorithms in Table 5.3. For two algorithms in Table 5.2, we count the number of the data sets, where the first algorithm performs better, equally to, and worse than the second one. As shown in Table 5.3, the DFL algorithm performs better than other compared algorithms when applied to most of selected data sets with all features. The average accuracies shown in Table 5.2 also reflect this result.

In Table 5.4, we also compare some of our results with those in the literature, on condition that the training/testing samples are the same as those in Table 5.1. As shown in Table 5.4, the DFL algorithm also performs fairly well when compared to the results in the literature. Except for the Monk2, Nursery, Vote and ALL data set, the DFL algorithm obtains better or equal prediction accuracies for the data sets in Table 5.4. However, the DFL algorithm uses much fewer features than the methods in the literature. This suggests that the DFL algorithm finds preferable models to those used in these literature, based on the principle of Occam's razor.

5.3.3 Comparison of Model Complexity

First, we compare the model complexities of different classification algorithms. From Figure 6.3, it can be seen that the classifiers of the DFL algorithm are very simple, 23 out of 24 learned models with fewer than 10 features (details available at Table D.1 and D.2). The model from the C4.5 algorithm is comparable to our models (details available at the supplementary Table S15 to S16), but the performances of the C4.5 algorithm are not better than our method. The NB, 1NN, kNN and SVM algorithms build very complex models, using all features of the data sets. The complex models from these algorithms make it difficult for the users to understand which subset of features is really important in contributing to the class distinctions between samples. When handling multi-class data sets, such as the LED data sets, the SVM algorithm and the NB algorithm solve the problems by building individual one-vs-all (OVA) pairwise classifiers for each class. Although effective in practice, this method also makes the model even more complex than individual classifiers obtained from the SVM algorithm and the NB algorithm. In comparison, the DFL algorithm just builds one model for multi-class data sets.

Next, we compare the model complexity of the DFL algorithm with those in literature. As shown in Table 5.4, the DFL algorithm uses much fewer features than those methods in literature and obtains comparable or more competitive prediction accuracies.

5.3.4 Comparison of Efficiency

Since all compared algorithms are implemented with the Java language and all experiments are performed on the same computer, the comparisons of their efficiency are meaningful. In this section, we will compare the training time of the DFL algorithm with those of the C4.5, NB and SVM algorithm, as shown in Figure 5.2. The detailed training times are also provided in the supplementary Table S17 and S18. As shown in Table 5.1, there are 7 data sets with cross validation as testing data sets. For these 7 data sets, the training times of the DFL algorithm shown in Figure 5.2 are the total run times of the corresponding cross validation tests. We do not compare the training times of the DFL algorithm with those of the 1NN and kNN algorithm, since these two algorithms actually build very simple models and spend most of their run times on testing.

As shown in Figure 5.2, the training time of the DFL algorithm and the C4.5 algorithm is comparable. The NB and SVM algorithm is more and less efficient than the DFL algorithm respectively. Particularly, the DFL algorithm is very efficient when applied to high-dimensional data sets, like the last 4 data sets in Figure 5.2.

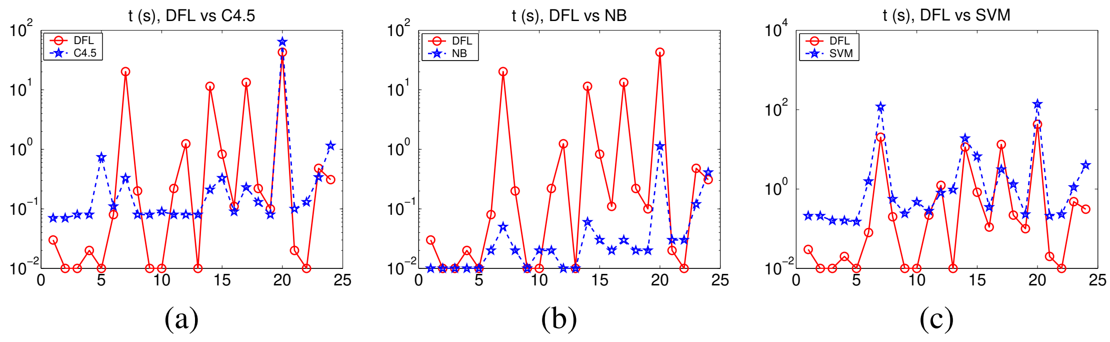


Figure 5.2: The comparison of training time of different classification algorithms. The results are for the discretized data sets. The horizontal axis is the index of data sets. In all parts, the curves marked with circles and pentagrams represent the training time of the DFL algorithm and other classification algorithms. (a) The training time of the DFL algorithm and the C4.5 algorithm. (b) The training time of the DFL algorithm and the NB algorithm. (c) The training time of the DFL algorithm and the SVM algorithm.

5.4 Biological Evaluation

The ALL data set consists of 72 bone marrow or peripheral blood samples of patient with acute myeloid leukemia (AML) or acute lymphoblastic leukemia (ALL) [81]. The training data set contains 27 ALL and 11 AML and the testing data set contains 20 ALL and 14 AML samples.

There are three classes in the MLL data set, ALL, AML and MLL. MLL is a new subtype of leukemia with rearranged *MLL* gene [18]. The MLL data set consists of 57 training samples, with three classes, 20 ALL, 20 AML and 17 MLL. In [18], Armstrong *et al.* did validation on an independent testing data set of 10 samples, and made 1 error. The testing data set that we used consists of 15 samples, 4 ALL, 8 AML, and 3 MLL.

There are two classes in the DLBCL data set, Diffuse Large B-Cell Lymphomas (DL-BCL) and Follicular Lymphoma (FL) [154]. The training/testing split schema used in this

Table 5.5: The classifier for the ALL data set learned with the DFL algorithm.

CST3	Class	Count
$(-\infty - 1419.5]$	ALL	27
$(1419.5 - \infty)$	AML	10
$(-\infty - 1419.5]$	AML	1

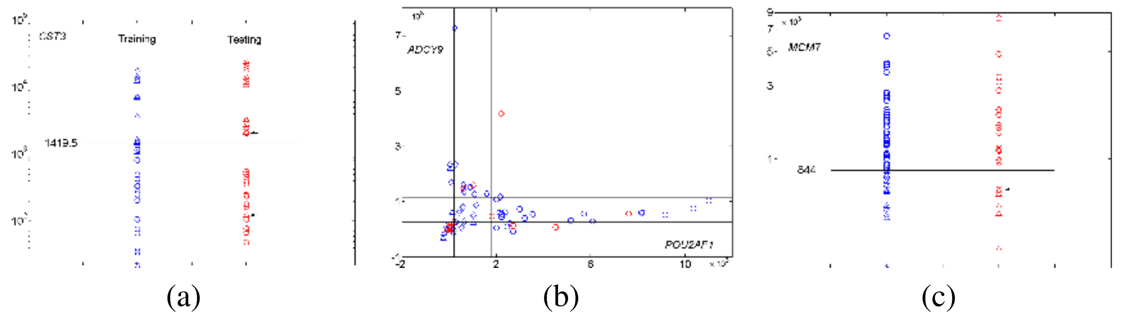


Figure 5.3: The comparisons of the expression values of the genes chosen by the DFL algorithm. In part (a) and (b), ALL, AML and MLL samples are represented with circles, triangles and diamonds respectively. In all parts, blue and red samples are from training and testing data sets respectively. The black solid lines are the cutting points of the genes introduced in the discretization preprocessing. (a) The expression values of *CST3* in the ALL data set. The two samples pointed by arrows are the incorrect predictions. (b) The expression values of *POU2AF1* and *ADCY9* in the MLL data set. (c) The expression values of the *MCM7* in the DLBCL data set. The sample pointed by an arrow is the incorrect prediction. The DLBCL and FL samples are represented with circles and triangles respectively.

thesis is 55:22, and DLBCL sample ratio of 41:17 and FL sample ratio of 14:5. The training/testing samples used are randomly chosen from the original data set.

As shown in Table 5.5, the DFL algorithm learns the optimal classifier of three rules for the ALL data set. The cutting point of the $CST3$ gene is determined by the discretization preprocessing [67]. The first rule of Table 5.5 means that if the expression level of $CST3$ gene is smaller than or equal to 1419.5, then the sample is an ALL sample.

Figure 5.3 shows the expression values of the genes chosen by the DFL algorithm in the ALL, MLL and DLBCL data sets. In part (a) and (c) of Figure 5.3, the expression values are shown in logarithmic values, so some samples with negative expression values are not shown. As shown in Figure 5.3 (a), the classifier in Table 5.5 only makes two incorrect predictions in the ALL testing data set. *CST3* (Cystatin C, M27891) is one of the 50 genes most highly correlated with the ALL-AML class distinction in the classification model of Golub *et al.* [81]. In Figure 5.3 (b), it can be seen that the samples in the MLL testing data set are all correctly classified in the *EA space* defined by the two genes *POU2AF1* and *ADCY9*. *POU2AF1* is one of the genes required for the appropriate B-cell development and one of the genes that are specifically expressed in MLL, ALL or AML [18]. From Figure 5.3 (b), it can be seen that most AML, MLL and ALL samples are located in the left, central and right regions divided by the cutting points of the *POU2AF1* expression values respectively. *ADCY9* is not as discriminative as *POU2AF1*, however, it serves as a good complement to *POU2AF1*. *POU2AF1* captures 77% diversity (entropy) of the class attribute in the MLL training data set, but the combination of *POU2AF1* and *ADCY9*, as a vector, captures 94.7% of the same measurement. For the DLBCL data set, the DFL algorithm selects *MCM7* (*CDC47* homolog) gene, which is associated with cellular proliferation and one of the genes highly correlated with the class distinctions [154].

5.5 Discussions and Conclusions

The fundamental difference between the DFL algorithm and other classification methods lies in the underlying philosophy of the algorithms, as shown in Figure 5.4. What the DFL algorithm does is to estimate the classification functions directly (based on Theorem 2.2.2) with low-complexity models, as demonstrated in Table 2.4. However, other classification methods are trying to approximate the classification functions with complex models, like

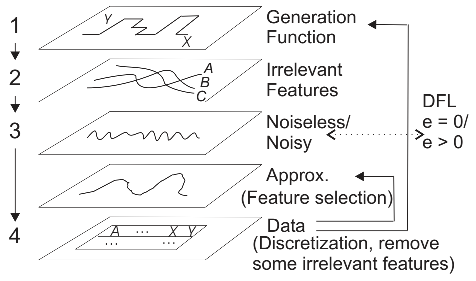

1
2
3
4
Y
X
A
B
C
Generation Function
Irrelevant Features
Noiseless/ Noisy
Approx. (Feature selection)
Data (Discretization, remove some irrelevant features)
DFL
e = 0/
e > 0

Figure 5.4: The philosophy of the DFL algorithm and other classification algorithms. $Y = f(X)$ is the generation function. The 1, 2, 3 and 4 are four steps in the production of data sets. The arrows on the left represent the production process of the data sets. In the first step, the generation function generates the original data sets. In the second and the third step, irrelevant features and noise are introduced into the data sets respectively. The arrows on the right stand for the learning philosophy of different algorithms. Other algorithms, like Multi-Layer Perceptrons and SVMs, are approximating the generation function with complex models from noisy data sets. The feature selection process is an optional step for these algorithms. However, the DFL algorithm directly estimates the generation function with low-complexity models. As indicated by the dotted arrow, when the data sets are noisy or noiseless, the DFL algorithm uses the positive or zero ϵ values. The discretization step [67] is optional for all algorithms, and helps to remove some irrelevant features from continuous data sets.

what have been done by the Multi-Layer Perceptrons (MLPs) and the SVMs with different kernels. The complex MLP and SVM models are “black boxes” and very hard to understand, while the DFL algorithm provides understandable low-complexity models without loss of prediction performances, as shown in Table 5.2 and 5.4.

The DFL algorithm, combined with the weighted 1NN prediction method, is used to solve classification problems. We demonstrate that the DFL algorithm can obtain comparable accuracies to existing well-known classification algorithms and those in the literature, with lower-complexity models. By comparing the training time, the DFL algorithm is shown to be both accurate and efficient, especially for the high-dimensional data sets.

Chapter 6

Performing Feature Selection

FEATURE SELECTION is fundamental for avoiding the *curse of dimensionality*. We will discuss the feature selection problem with the ILA in this chapter. By choosing a *Markov Blanket* $\mathbf{X}$ of Y as the candidate feature subsets, the irrelevant and redundant features can be automatically eliminated and prevented from deteriorating the performances of classification algorithms.

In this chapter, we use the DFL algorithm to choose informative and discriminatory feature subsets for other algorithms. We show that the DFL algorithm can find small subsets of features, on which the classification algorithms can obtain better prediction performance with less time.

This chapter is organized as follows. In Section 6.1, we will describe the motivation of feature selection problem, and introduce two problems which are not formally solved by current methods. In Section 6.2, we describe related feature selection methods, and discuss their limitations. In Section 6.3, we discuss the two problem introduced in Section 6.1 with the ILA. Section 6.4 shows the experimental results and compare the performances of the DFL algorithm with those from other feature selection methods. In Section 6.5, we discuss the differences between the DFL algorithm and other feature selection methods. Finally, we summarize the work of this chapter in Section 6.6.

6.1 Introduction

In solving classification problems, many induction algorithms suffer from the *curse of dimensionality* [104]. There exist two aspects for this curse. First, when the number of features increases, the run time of the algorithms grows very fast, even exponentially. Second, in data sets with a large number of features, the data points are often scarce which makes it very difficult to correctly estimate the parameters of classification models or find good margins between the classes. Therefore, the number of learning samples needed to find models grows quickly, sometimes also exponentially. Unfortunately, the data sets are actually limited, which makes many algorithms suffer from the risk of overfitting the training data sets with their complex models. Furthermore, the inclusion of irrelevant, redundant and noisy attributes in the model building process phase can also result in poor predictive performance and increased computation [87].

To overcome the overfitting problems, the principle of Occam's razor is often used in building classification models. According to the principle of Occam's razor, a small set of informative and discriminatory features is preferable to a large number of features, if the models built over the small set of features can obtain comparable prediction performances to those built over large number of features [104]. Hence, feature selection is critical to overcome the overfitting problems by finding the informative and discriminatory features, to improve the performance of classification algorithm, and to further avoid the *curse of dimensionality*. Based on the different nature of the metric used to evaluate features, feature selection methods fall into two categories, called "filter" and "wrapper" methods [100]. In the filter methods, the feature selection is performed as a preprocessing step and often independent of the classification algorithms which will be applied to the processed data sets later. The wapper method [103] evaluates a subset of features by applying a target learning algorithm to the training data set with cross validation, and selects the subset of features which produces the highest accuracy in the cross validation process. The evaluation with cross validation makes the wrapper method very inefficient when dealing with the high-dimensional data sets like gene expression profiles.

In this chapter, we will concentrate on the filter methods. The existing filter feature selection methods often require a predefined number of features k. However, the performance of algorithms applied on the selected features may be sensitive to the predefined k, as demonstrated by Koller and Sahami [104]; Peng *et al.* [133]; Chow and Huang [45].

In a more general view, the following two problems are central to the effectiveness of the feature selection methods, for both filters and wrappers.

Problem 1 *How to know the selected features are complete to determine the class value, at least for the training data set?*

Problem 2 *How to determine the optimal subset of features?*

In this chapter, we discuss the two problems with the ILA. As shown in Theorem 2.3.1, the feature subset $\mathbf{U}$ which satisfies $I(\mathbf{U};Y) = H(Y)$ is *Markov Blanket* of Y. By choosing a *Markov Blanket* $\mathbf{U}$ of Y as the candidate feature subset, the irrelevant and redundant features can be automatically eliminated from deteriorating the performances of classification algorithms. We name the subset of the features in the *Markov Blanket* or its estimation as the *essential attributes*, or the EAs for short. We use the Discrete Function Learning (DFL) algorithm [184, 189] to efficiently find or estimate the *Markov Blanket* $\mathbf{U}$ of Y from the exponential number of subsets.

6.2 Related Work

In this section, we will describe current feature selection methods, and discuss their limitations. First, we categorize current feature selection methods. Then, we specifically describe feature selection methods based on information theory. Finally, we analyze the shortcomings of them.

6.2.1 Categorization of Feature Selection Methods

Feature selection methods fall into two main categories, those evaluating individual features and those evaluating feature subsets.

In the individual feature selection methods, the evaluation statistics for each feature are calculated, then a feature ranking list is provided in predefined order of the statistics. The statistics used for individual feature selection include information gain [87, 115, 174], signal-to-noise (S2N) statistic [18, 77, 81, 154], correlation coefficient [167], t-statistic [115], F-statistic [61], χ^2-statistic [110, 115] and others. The main shortcoming of these individual feature selection methods lies in that a larger than necessary number of redundant top features with similar value patterns, like gene expression patterns, are selected to build the models. Hence, such choice often brings much redundancy to the models, since the selected features carry similar information about the class attribute. According to the principle of Occam's razor, these models are not optimal although accurate, since they are often complex and suffer from the risk of overfitting the training data sets [174]. In addition, the large number of features in the predictors makes it difficult to know which features are really useful for recognizing different classes.

In the feature subset selection methods, a search algorithm is often employed to find the optimal feature subsets. In evaluating a feature subset, a predefined score is calculated for the feature subset. Since the number of feature subsets grows exponentially with the number of features, heuristic searching algorithms, such as forward greedy selection, are often employed to solve the problem. Examples of feature subset selection methods are CFS (Correlation-based Feature Selection) [86], CSE (Consistency-based Subset Evaluation) [116], and the WSE (Wrapper Subset Evaluation) [103]. Most feature subset selection methods use heuristic scores to evaluate feature subset under consideration, such as CFS and CSE methods. As discussed in Section 6.1, the WSE method is inefficient, especially when dealing with high-dimensional data sets.

There is another popular way of categorizing these algorithms, called "filter" and "wrapper" methods [100], as discussed in Section 6.1.

Logistic regression is a regularization or shrinkage methods which trim the hypothesis space by constraining the magnitudes of parameters [30, 174]. L_1 regularization uses a penalty term which encourages the sum of the absolute values of the parameters to be small [128]. L_2 regularization encourages the sum of the squares of the parameters to be small [128]. It has been frequently been observed that L_1 regularization in many models causes many parameters to equal zero, so that the parameter vector is sparse [128]. This makes it a natural candidate in feature selection settings, such as the work in [40,153,194,195], where we believe that many features should be ignored [128]. In comparison, feature selection methods normally search good features in the combinatorial space of feature subsets [174] based on some measures, such as the high-dimensional mutual information used by the DFL algorithm [189].

6.2.2 Feature Selection Methods Based on Information Theory

Some feature selection methods based on mutual information have been introduced by Dumais *et al.* [65], Yang and Pedersen [176], Vidal-Naquet and Ullman [169], Fleuret [68], Chow and Huang [45] and Peng *et al.* [133]. These methods also fall into two categories.

In the first category, features are ranked according to their mutual information with the class label. Then, the first k features [65] or the features with a bigger mutual information than a predefined threshold value [176] are chosen.

The second category is feature subset selection. In this category, the forward selection searching algorithm, i.e., greedy algorithm, is often used to find the predefined k features. In the first iteration, the X_i which shares the largest mutual information with Y is selected to the target feature subset $\mathbf{U}$. Then, in the next step, the selection criterion is how much information can be added with respect to the already existing $X_{(1)}$. Therefore, the $X_{(2)}$ with maximum $I(X_i, X_{(1)}; Y) - I(X_{(1)}; Y)$ is added to $\mathbf{U}$ [169]. Formally, the features $X_{(1)}, \ldots, X_{(k)}$ are selected with the following criteria, $X_{(1)} = \arg\max_i I(X_i; Y)$ and

$$X_{(s)} = \arg\max_{X_i \in \mathbf{P}_{s-1}} \min_{X_{(j)} \in \mathbf{U}_{s-1}} [I(X_i, X_{(j)}; Y) - I(X_{(j)}; Y)], \tag{6.1}$$

where $\forall s$, $1 < s \leq k$, $i = 1, \ldots, (n - s + 1)$, $j = 1, \ldots, (s-1)$, and $\mathbf{P}_s$ is the feature pool by removing $X_{(1)}, \ldots, X_{(s)}$, with $\mathbf{P}_1 = \mathbf{V} \setminus X_{(1)}$, $\mathbf{P}_s = \mathbf{P}_{s-1} \setminus X_{(s)}$, and $\mathbf{U}_s$ is the set of selected features, with $\mathbf{U}_1 = \{X_{(1)}\}$, $\mathbf{U}_s = \mathbf{U}_{s-1} \cup \{X_{(s)}\}$.

From Theorem 2.1.2, we have

$$I(X_i, X_{(j)}; Y) = I(X_{(j)}; Y) + I(X_i; Y|X_{(j)}),$$

then

$$I(X_i; Y|X_{(j)}) = I(X_i, X_{(j)}; Y) - I(X_{(j)}; Y).$$

Therefore, Equation 6.1 is equivalent to maximizing conditional mutual information, $\min_{X_{(j)} \in \mathbf{U}_{s-1}} I(X_i; Y|X_{(j)})$ [68] in Equation 6.2.

$$X_{(s)} = \arg\max_{X_i \in \mathbf{P}_{s-1}} \min_{X_{(j)} \in \mathbf{U}_{s-1}} I(X_i; Y|X_{(j)}). \tag{6.2}$$

Battiti [23] introduced a heuristic algorithm to find the feature subsets. In this method, the mutual information $I(X_i; Y)$ of a new feature X_i is penalized by a weighted sum of the $I(X_i; X_{(j)})$, where $X_{(j)} \in \mathbf{U}_{s-1}$. This method is similar to those in [68, 169], but not theoretically formulated.

Chow and Huang [45] proposed an approximation method to evaluate mutual information between continuous features and the class attribute. Then, Chow and Huang [45] used the heuristic criteria feature relevance criterion (FRC) and feature similarity criterion (FSC) in Equation 6.3 and 6.4 respectively to choose features with a forward selection process.

$$FRC(X_i) = I(\{\mathbf{U}, X_i\}; Y) \tag{6.3}$$

$$FSC(X_i) = \arg\max_{X_{(j)} \in \mathbf{U}} \left(\frac{I(X_i; X_{(j)})}{H(X_{(j)})}\right) \tag{6.4}$$

This method essentially finds the most relevant feature with maximal $FRC(X_i)$, then evaluates its redundancy by calculating $FSC(X_i)$ with respect to the selected features individually. If $FSC(X_i)$ is larger than a predefined threshold value, it is considered as a redundant feature and will not be chosen [45].

Peng *et al.* [133] proposed to use $X_{(1)} = \arg\max_i I(X_i; Y)$ and Equation 6.5 to choose a new feature.

$$X_{(s)} = \arg\max_{X_i \in \mathbf{P}_{s-1}} [I(X_i; Y) - \frac{1}{s-1} \sum_{X_{(j)} \in \mathbf{U}_{s-1}} I(X_i; X_{(j)})] \tag{6.5}$$

Peng *et al.* [133] also used an approximation method to calculate the mutual information between continuous features and the class attribute.

Koller and Sahami [104] proposed to choose *Markov Blanket* of the class attribute as the candidate features for classification problems, and introduced an approximation algorithm to estimate the *Markov Blanket* of the class attribute. The method by Koller and Sahami

[104] tries to find a subset of features which minimize the distance between the distribution of the selected feature subsets and the distribution of all features. The backward selection algorithm is used to eliminated features which minimize the expected cross-entropy [104] until some predefined number of features have been eliminated.

6.2.3 Limitations of Current Feature Subset Selection Methods

For all existing feature subset selection method based on mutual information, one common major shortcoming is that the candidate feature is evaluated with respect to every individual feature in the selected features subset $\mathbf{U}_{s-1}$ step by step. The motivation underlying Equation 6.1 and 6.2 is that X_i is good only if it carries information about Y, and if this information has not been caught by any of the $X_{(j)}$ already picked [68]. However, it cannot be known whether the existing features as a vector have captured the information carried by X_i or not. Another shortcoming is that it needs to specify the number of features k in prior. As discussed in Section 6.1, the performances of algorithms applied to the selected features may be sensitive to the predefined k. In addition, it also introduces some redundant computation when evaluating the new feature X_i with respect to the already picked features $X_{(j)} \in \mathbf{U}_{s-1}$, which will be discussed further in Section 6.5.

Furthermore, there are still two problems which are not formally solved by previous methods. First, to find the most informative feature subsets, the aim is to maximize the conditional mutual information, $I(Y; X_{(s)}|\mathbf{U}_{s-1}) = H(Y|\mathbf{U}_{s-1}) - H(Y|\mathbf{U}_{s-1}, X_{(s)})$, as shown in Equation 2.7. $H(Y|\mathbf{U}_{s-1})$ does not change when trying different $X_i \in \mathbf{V} \setminus \mathbf{U}_{s-1}$. Hence, the ultimate goal of feature subset selection is converted to finding $\{\mathbf{U}_{s-1}, X_{(s)}\}$ which minimizes $H(Y|\mathbf{U}_{s-1}, X_{(s)})$, as pointed out by [68]. But $H(Y|\mathbf{U}_{s-1}, X_{(s)})$ cannot be estimated with a training set of realistic size as it requires the estimation of 2^{k+1} probabilities [68]. Hence, Fleuret [68] and Vidal-Naquet and Ullman [169] proposed the estimated increase of the information content of the feature subset using Equation 6.1 and 6.2. Second, as pointed by Kollar and Sahami [104], finding either a true or even an approximate *Markov Blanket* might be very hard. In particular, the expected cross-entropy does not really test for the *Markov Blanket* property [104].

In the next section, we will demonstrate that it is unnecessary to compute $H(Y|\mathbf{U}_{s-1}, X_{(s)})$, as the problem can be directly solved by maximizing $I(\mathbf{U}_{s-1}, X_{(s)}; Y)$.

6.3 Choosing Essential Attributes with The Information Learning Approach

In this section, we show that the two problems introduced in Section 6.1 can formally be solved from the viewpoint of the ILA.

Let us recall the **Problem 1** in Section 6.1. Based on Theorem 2.2.2, it is known that if $I(\mathbf{X}; Y) = H(Y)$, then the subset $\mathbf{X}$ can fully determine the value of Y for the training data sets. And from Theorem 2.2.4, is is known that if $I(\mathbf{X}; Y) = H(Y)$, then $\forall \mathbf{Z} \in \mathbf{V} \backslash \mathbf{X}$, $\mathbf{Z}$ provides no information of Y. Hence, the $\mathbf{X}$ which satisfies $I(\mathbf{X}; Y) = H(Y)$ is the complete set of features needed to determine the value of Y for the training data set.

Recall the **Problem 2** in Section 6.1. Based on Theorem 2.3.1, it is known that if $I(\mathbf{X}; Y) = H(Y)$, then $\mathbf{X}$ is a *Markov Blanket* of Y. Hence, in the feature selection process of our method, the $I(\mathbf{X}; Y)$ is evaluated with respect to $H(Y)$. It is critical to compare $I(\mathbf{X}; Y)$ with $H(Y)$. Since by performing this comparison, the optimal subset of features can be automatically determined, without the need of specifying a predefined number of features or a threshold value of the mutual information.

As discussed in Section 2.4.1, when choosing candidate features, our approach maximizes the mutual information between the feature subsets and the class attribute. Suppose that $\mathbf{U}_{s-1}$ is the already selected feature subset in step $s-1$, and the DFL algorithm is trying to add a new feature $X_i \in \mathbf{V} \setminus \mathbf{U}_{s-1}$ to $\mathbf{U}_{s-1}$. Specifically, our method uses $X_{(1)} = \arg\max_i I(X_i; Y)$ and Equation 2.11 to add new features to $\mathbf{U}$, i.e.,

$$X_{(s)} = \arg\max_i I(\mathbf{U}_{s-1}, X_i; Y),$$

where $\forall s$, $1 < s \leq k$, $\mathbf{U}_1 = \{X_{(1)}\}$, and $\mathbf{U}_s = \mathbf{U}_{s-1} \cup \{X_{(s)}\}$.

As demonstrated in Section 2.4.1, the irrelevant and redundant features can be automatically removed, if the new candidate feature X_i is evaluated with respect to the selected features as a vector $\mathbf{U}_{s-1}$ by maximizing $I(\mathbf{U}_{s-1}, X_i; Y)$. Furthermore, the optimal subset of features can be determined by evaluating the $I(\mathbf{U}; Y)$ with respect to $H(Y)$.

6.4 Results

We will use the data sets in Table 5.1 to validate the DFL algorithm as a filter feature selection method. In this section, we first show the improvement of other classification algorithms when they are applied to the data sets filtered with the features chosen by the DFL algorithm, or in short, to the DFL features. Then, we compare the DFL algorithm with other feature selection methods.

6.4.1 The DFL Algorithm as A Filter Feature Selection Method

In this thesis, we use the restricted learning method introduced in Section 2.6.2 to obtain optimal models for the DFL algorithm, with the searching scope of the ϵ from 0 to 0.8. The optimal settings are shown in Table D.1, which is also available at the supplementary website of this thesis as Table S2. The features chosen by the DFL algorithm are listed in Table D.2.

In this section, we apply the well-known classification algorithms to the data sets filtered with the features chosen by the DFL algorithm, or in short, to the DFL features. The accuracy results are shown in Table 6.1, where the values are the averages of 10 runs. For comparison, the accuracies of the DFL algorithm, which is for all features, are also given in Table 6.1.

As shown in Table 6.1, the compared classification algorithms generally perform better for the Lenses, LED+17, Chess, Lung, ALL, MLL and Ovarian data sets when applied the the DFL features. In Table 6.2, we also check the improvements of the different algorithms when applied to the DFL features. Similar to Table 5.3, for a given classification algorithm, we count the number of data sets, where the algorithm applied to the DFL features performs better, equally to, or worse than to all features.

As shown in Table 6.2, the learning algorithms, except the SVM algorithm, performs better when applied to the DFL features than to the original data sets with all features. For the SVM algorithm, although there are more data sets whose accuracies are better on all features than on the DFL features, but the average accuracy of all data sets on the DFL features is better than that on all features, as shown in Table 6.1 and Table 5.2. Meanwhile, as the number of features chosen by the DFL algorithm is much smaller than the total number of features (see Table D.1), the run times of the compared well-known classification algorithms are dramatically reduced, as shown in Figure 6.1. For instance, the SVM algorithm uses 138 seconds to train a model for the discretized Ad data set with all features, and uses only 1.6 seconds to train a model for the discretized Ad data set with the DFL features. Meanwhile, as shown in Table 5.2 and 6.1, the accuracies of the SVM algorithm for the discretized Ad data set with all features, 95.0, is only 0.6 percent higher than that on the DFL features, 94.4. Hence, it may be a better trade-off.

Table 6.1: The accuracies of the well-known classification methods on the data sets filtered with the features chosen by the DFL algorithm. The unit is percent. The accuracies for those data sets with numerical attributes are for discretized/numerial data sets.

	Data Set	DFL	C4.5	NB	1NN	kNN[1]	SVM
1	Lenses	75.0	87.5	75.0	75.0	62.5	66.7
2	Iris	96.0	94.0/92.0	94.0/94.0	96.0/98.0	92.0/94.0	94.0/96.0
3	Monk1	100.0	88.9	72.2	97.2	77.8	72.2
4	Monk2	73.8	65.0	61.6	73.8	62.3	67.1
5	Monk3	97.2	97.2	97.2	88.9	97.2	97.2
6	LED	74.9	74.6	75.1	71.0	75.6	75.3
7	Nursery	93.1	93.1	89.1	89.9	93.1	90.4
8	Breast	95.0	94.8	96.2	94.6	94.4	95.0
9	Wine	98.3	93.2/93.2	96.6/98.3	93.2/96.6	96.6/98.3	94.9/98.3
10	Credit	88.3	87.4/87.4	87.4/87.4	44.3/44.3	87.4/87.4	87.4/87.4
11	Vote	95.7	94.9	92.2	94.9	95.7	94.9
12	Zoo	92.8	90.1	93.1	96.6	83.2	94.3
13	ImgSeg	90.6	90.4/90.8	90.8/84.3	86.8/92.0	89.8/91.1	90.7/76.1
14	Mushroom	100.0	100.0	98.6	100.0	100.0	100.0
15	LED+17	75.4	75.1	74.2	57.8	75.2	75.1
16	Ionosphere	94.9	93.2/94.9	95.7/94.0	89.7/91.5	94.9/87.2	94.9/80.3
17	Chess	97.4	99.0	90.5	97.0	96.4	96.1
18	Anneal	99.0	84.0/84.0	80.0/74.0	84.0/84.0	91.0/91.0	89.0/88.0
19	Lung	62.5	68.8	56.3	58.1	50.0	71.9
20	Ad	95.0	92.6/94.4	93.2/92.4	36.0/90.7	93.8/94.2	94.4/92.6
21	ALL	94.1	94.1/94.1	94.1/94.1	94.1/94.1	94.1/94.1	94.1/82.4
22	DLBCL	95.5	95.5/95.5	95.5/90.9	77.3/86.4	95.5/95.5	95.5/77.3
23	MLL	100.0	93.3/93.3	100.0/100.0	100.0/100.0	93.3/100.0	100.0/73.3
24	Ovarian	98.8	98.8/98.8	98.8/98.8	35.7/97.6	98.8/98.8	98.8/97.6
	average	91.0	89.4/89.5	87.4/86.7	80.5/86.2	87.2/87.3	88.7/85.2

[1] The k value of the kNN algorithm is set to 5.

6.4.2 Comparison with Other Feature Selection Methods

In this section, we compare the DFL algorithm with two well-known filter feature subset selection methods, the CFS method by Hall [86] and the CSE method by Liu and Setiono [116], and the wrappers subset selection method, i.e. the WSE method, by Kohavi and John [103]. We do not compare the DFL algorithm with the method proposed by Fleuret [68] and Vidal-Naquet and Ullman [169], since their methods can only deal with boolean

Table 6.2: The improvement of performances of different learning algorithms when applied to the features chosen by the DFL algorithm.

		Discretized Data Sets			Continuous Data Sets		
Algo.	Features	better	equally	worse	better	equally	worse
C4.5	DFL:all	11	7	6	9	7	8
NB	DFL:all	11	6	7	12	6	6
1NN	DFL:all	12	3	9	17	3	4
kNN	DFL:all	12	7	5	14	4	6
SVM	DFL:all	7	10	7	6	6	12
	sum	53	33	34	58	26	36

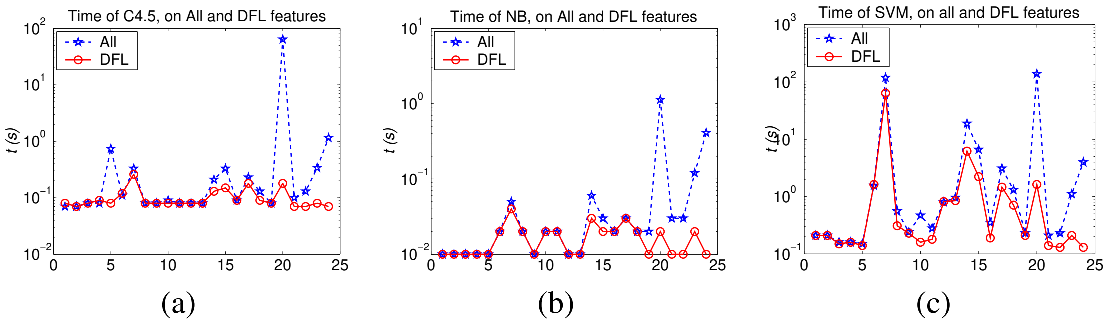


Figure 6.1: The comparison of training times of different classification algorithms on all features and on the features chosen by the DFL algorithm. The results are for the discretized data sets. The horizontal axis is the index of data sets. In all parts, the curves marked with circles and pentagrams represent the training times on DFL features and all features of the data sets. (a) The training times of the C4.5 algorithm. (b) The training times of the NB algorithm. (c) The training times of the SVM algorithm.

features. We use the CFS, CSE and WSE implemented by the *Weka* software to compare their results with those from the DFL algorithm. As discussed in Section 6.2.1, the forward selection is used with the CFS, CSE and WSE feature subset selection methods.

We choose three classification algorithms with different theoretical foundation, the

C4.5, NB and SVM algorithm, to validate different feature subsets selection methods. For the SVM algorithm, the linear kernels are used. These algorithms are applied to the DFL, CFS, CSE, and WSE features with discretized values and original numerical values. The results for discretized values are shown in Figure 6.2. The results for original numerical values are similar to discretized values and not shown here. But the results of both the discretized and numerical values are summarized in Table 6.3. The details of accuracies for different feature selection methods are also given in the supplementary Table S3 to S6.

The 1NN and kNN algorithm are also applied to the CFS and CSE features, but not the WSE features, since the WSE takes too much time to iterate for the 1NN and kNN algorithms. The results for the 1NN and kNN algorithms when applied to the CFS and CSE features are available at the supplementary Table S4 and S5. We also perform experiments for the continuous data sets, whose results are available at the supplementary Table S3 to S6.

When using the CFS algorithm to perform feature selection, the *Weka* software reports out of memory error for the continuous MLL and Ovarian data sets. The CSE and WSE algorithm do not find a candidate feature subset for the Monk2 data set. In addition, the WSE algorithm when coupled with the SVM algorithm does not find a candidate feature subset for the Lenses data set. Therefore, the accuracies for these cases are not shown in Figure 6.2.

From Figure 6.2, it is shown that the learning algorithms generally perform better on the DFL features when the number of features in the data sets are large, such as the data sets with index from 15 to 24, than on other features. This again consolidates the good generality of the DFL algorithm for the high-dimensional data sets.

We also summarize the comparison of accuracies obtained by different feature selection methods in Table 6.3. Similar to Table 5.3, for two feature selection methods, we count the number of data sets, where the classification algorithm applied to features of the first

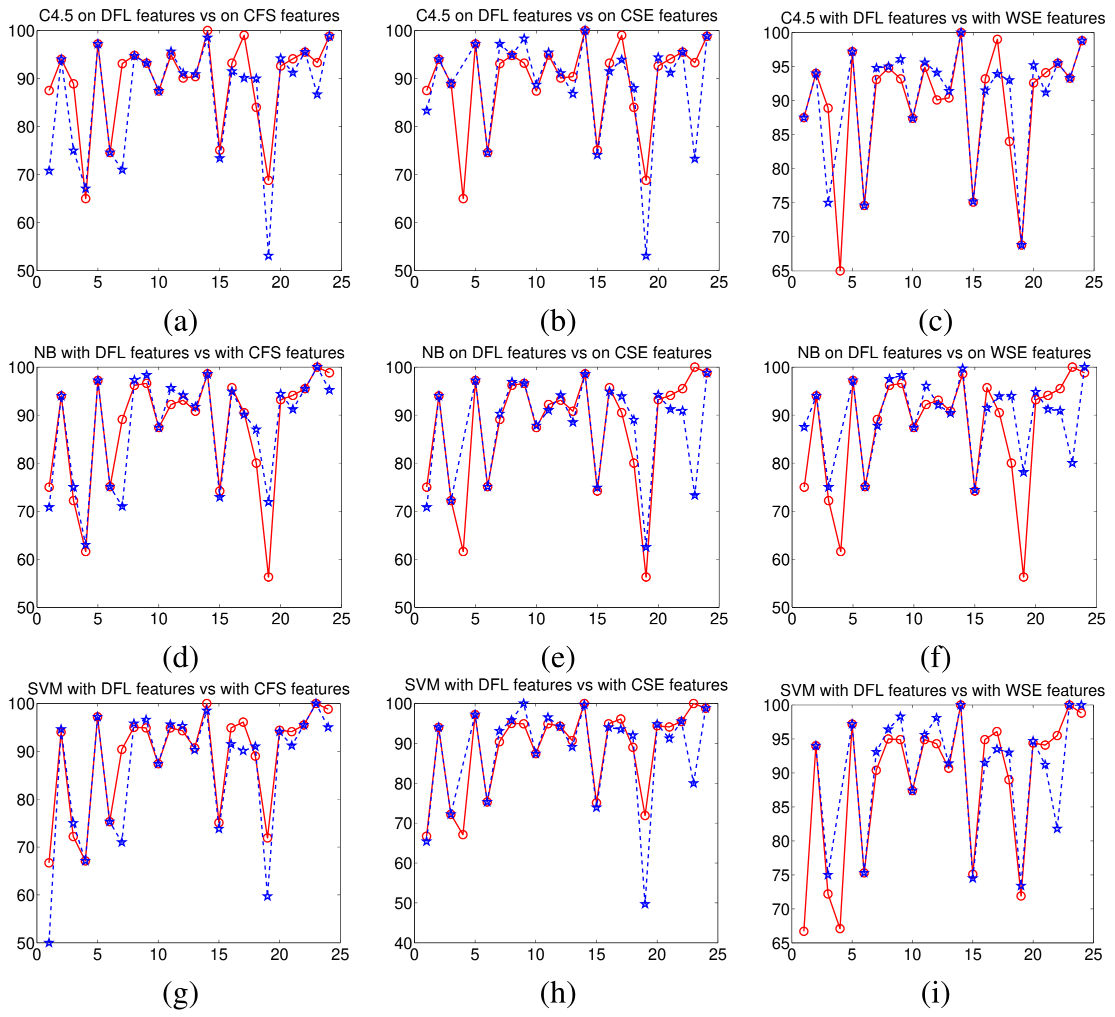


Figure 6.2: The comparison of accuracies for different feature subset selection methods. The values are for the discretized data sets. The curves marked with circles and pentagrams are for data sets filtered with the DFL algorithm and other feature subset selection methods respectively. Part (a) to (c) are the results for the C4.5 algorithm. Part (d) to (f) are the results for the NB algorithm. Part (g) to (i) are the results for the SVM algorithm.

Table 6.3: The comparison summary of accuracies obtained by different feature selection methods.

		Discretized D.S.			Continuous D.S.		
F.S. Pair	Learning Algo.	better	equally	worse	better	equally	worse
DFL:CFS	C4.5	11	7	6	13	5	4
	NB	8	6	8	8	5	9
	SVM	12	5	7	9	6	7
	sum	31	18	21	30	16	20
DFL:CSE	C4.5	8	7	8	9	6	8
	NB	8	6	9	10	5	8
	SVM	11	7	5	10	5	8
	sum	27	20	22	29	16	24
DFL:WSE	C45	4	10	9	7	7	9
	NB	7	4	12	7	4	12
	SVM	5	6	11	4	5	13
	sum	16	20	32	18	16	34

method performs better, equally to, or worse than applied to features of the second one.

From Table 6.3, it can be seen that the DFL algorithm chooses more discriminatory feature subsets than the CFS and CSE algorithm do, as the learning algorithms show better prediction performances on the DFL features than on those chosen by the CFS and CSE algorithm. The learning algorithms perform slightly better on the WSE features than on those chosen by the DFL algorithm, but the WSE algorithm chooses more features than the DFL algorithm does, as to be shown in Section 6.4.3.

6.4.3 Comparison of Model Complexity

The accuracy is only one aspect of the performance of each learning algorithm. The model complexity is another aspect of the performance of each one.

We compare the number of features chosen by different feature selection methods, as shown in Figure 6.3. The detailed results of the number of features chosen by different feature selection methods are also available at the supplementary Table S7 and S8. The feature

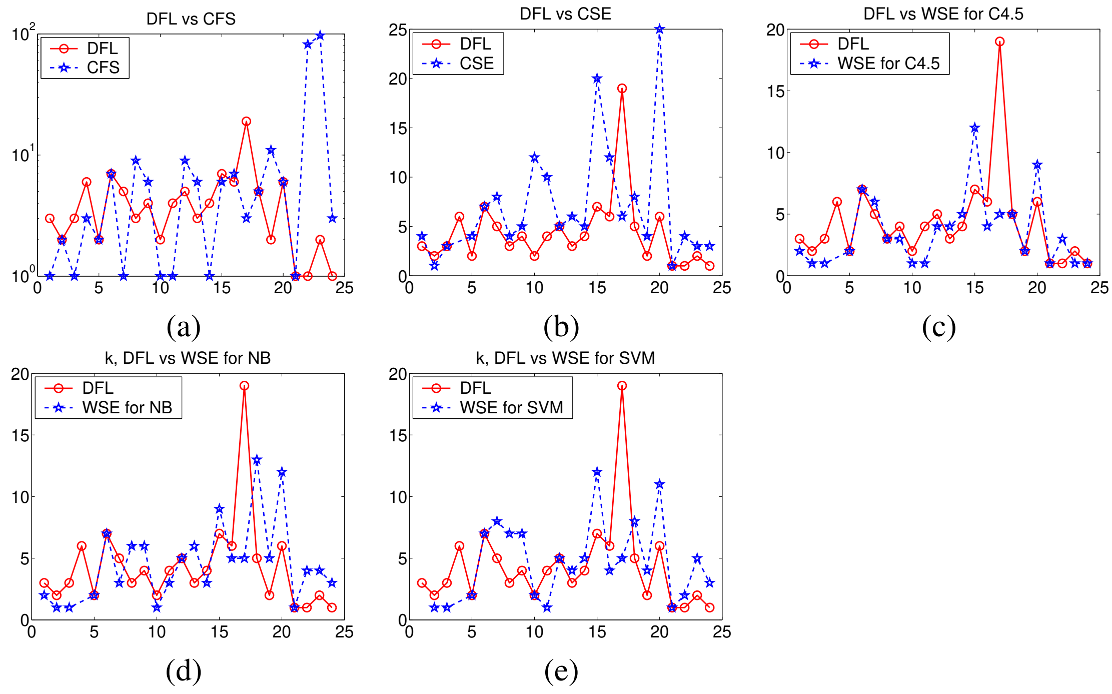


Figure 6.3: The number of features chosen by different feature selection methods. The results are for the discretized data sets. The horizontal axis is the index of data sets. In all parts, the curves marked with circles and pentagrams represent the number of features chosen by the DFL algorithm and other feature selection algorithms. (a) DFL vs CFS. (b) DFL vs CSE. (c) DFL vs WSE for C4.5. (d) DFL vs WSE for NB. (e) DFL vs WSE for SVM.

lists chosen by different feature selection methods are also available at the supplementary Table S9 to S14.

We also summarize the number of features for different feature selection methods in Table 6.4. Similar to Table 5.3, for two feature selection methods, we count the number of data sets, where the first method chooses smaller, equal, bigger number of features than the second one does. As summarized in Table 6.4, the DFL chooses comparable number of features to the CFS method, but less features than the CSE and WSE method do.

Table 6.4: The comparison summary of the number of features chosen by different feature selection methods.

		Discretized Data Sets			Continuous Data Sets		
F.S. Pair	Learning Algo.	smaller	equal	bigger	smaller	equal	bigger
DFL:CFS	NA[1]	9	6	8	7	6	9
DFL:CSE	NA	17	4	2	17	5	1
DFL:WSE	C4.5	6	9	8	8	7	8
	NB	10	4	9	9	7	7
	SVM	12	5	5	13	5	4
	sub sum	28	18	22	30	21	17
	total sum	54	26	34	54	30	29

[1] "NA" stands for not applicable.

6.4.4 Comparison of Efficiency

Next, we compare the run times of the DFL algorithm with other feature selection methods, as shown in Figure 6.4. The detailed training times are also provided in the supplementary Table S21 and S22. In Figure 6.4, it is shown that the the DFL algorithm uses less time than the CFS and CSE algorithm in most data sets, 18 and 20 out of the 24 data sets respectively. The DFL algorithm is overwhelmingly faster than the WSE algorithm for all the three learning algorithm used with the WSE algorithm. Especially for the high-dimensional data sets, those with index from 20 to 24, the DFL algorithm shows great reduction of run time when compared with other feature selection methods. These experimental results suggest that the DFL algorithm is faster than other compared feature selection methods. In addition, the WSE method uses much more time than other feature selection methods do, like the DFL, CFS and CSE method, as shown in Figure 6.4.

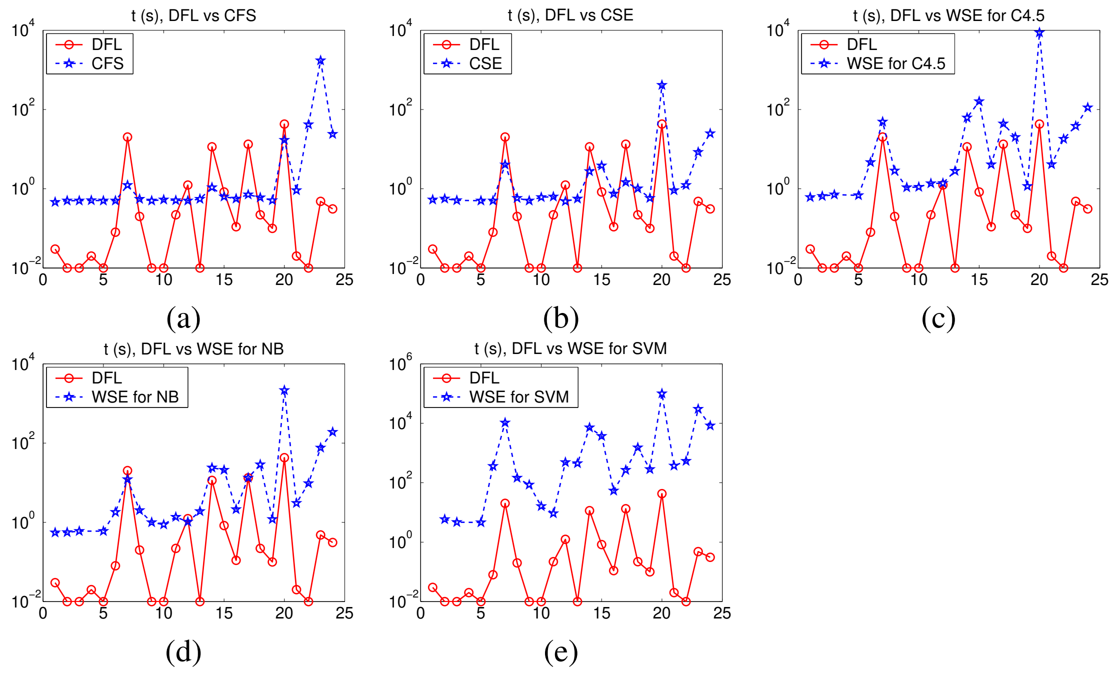


Figure 6.4: The run times of different feature selection methods. The results are for the discretized data sets. The horizontal axis is the index of data sets. In all parts, the curves marked with circles and pentagrams represent the run times of the DFL algorithm and other feature selection algorithms. (a) DFL vs CFS. (b) DFL vs CSE. (c) DFL vs WSE for C4.5. (d) DFL vs WSE for NB. (e) DFL vs WSE for SVM.

6.5 Discussions

The DFL algorithm can be categorized as a feature subset selection method or a filter method. However, the DFL algorithm is also different from other feature subset selection methods, like the CFS, CSE and WSE methods. Based on Theorem 2.2.2, the DFL algorithm can produce function tables for the training data sets, while other subset feature selection methods only generate a subset of features. Particularly, the DFL algorithm is different from existing feature subset selection methods based on information theory in the following four aspects.

First, the stopping criterion of the DFL algorithm is different from those of existing methods. The DFL algorithm stops the searching process based on Theorem 2.2.2. The existing methods stop the searching process with a predefined k or threshold value of the mutual information. Hence, the feature subsets selected by existing methods may be sensitive to the k or threshold value of the mutual information.

Second, the feature subset evaluation method of the DFL algorithm is also different from those in existing methods. $I(\mathbf{U};Y)$ is evaluated with respect to $H(Y)$ in the DFL algorithm. The DFL algorithm uses $X_{(1)} = \arg\max_i I(X_i;Y)$ and Equation 2.11, i.e.,

$$X_{(s)} = \arg\max_i I(\mathbf{U}_{s-1}, X_i; Y),$$

to evaluate a new feature, while existing methods use $X_{(1)} = \arg\max_i I(X_i;Y)$ and Equation 6.2 [68, 169], i.e.,

$$X_{(s)} = \arg\max_{X_i \in \mathbf{P}_{s-1}} \min_{X_{(j)} \in \mathbf{U}_{s-1}} I(X_i; Y | X_{(j)}),$$

or similar algebraic combinations of two dimensional mutual information [23, 45, 133] for the same purpose. As mentioned early in Section 2.4.1, to maximize $I(\mathbf{U}_{s-1}, X_i; Y)$ is equal to maximize $I(X_i; Y|\mathbf{U}_{s-1})$ in Equation 2.12. $I(X_i; Y|\mathbf{U}_{s-1})$ of Equation 2.12 is different from $I(X_i; Y|X_{(j)})$ of Equation 6.2 used in [68], where the new feature is evaluated with respect to individual features in $\mathbf{U}_{s-1}$. As intuitively shown in Figure 2.4, by considering the selected features as vectors, the redundancy introduced by new features to be added to $\mathbf{U}_{s-1}$ is automatically eliminated.

Furthermore, the maximization of $I(\mathbf{U}_{s-1}, X_i; Y)$ used in the DFL algorithm is more efficient than penalizing the new feature with respect to every selected features, as done by [23, 68, 169]. As analyzed in Section 2.8.1, to evaluate $I(\mathbf{U};Y)$, $O(n \cdot N)$ operations are needed when adding each feature, and $O(k \cdot n \cdot N)$ operations are necessary to choose k features in the DFL algorithm. However, in calculating $I(X_i, X_{(j)}; Y) - I(X_{(j)}; Y)$ [68, 169], since there are already $(s-1)$ features in $\mathbf{U}_{s-1}$ in the s iteration, there would be $(s-1) \times O(n \cdot N)$ operations in this iteration. Therefore, it needs $\sum_{s=1}^{k}(s-1) \times O(n \cdot N) \approx O(k^2 \cdot n \cdot N)$ operations to select k features, which is less efficient. The computational cost of the backward selection for approximating Markov Blanket is at least $O(2^k \cdot n \cdot N)$ [104], which is even worse than the $O(k^2 \cdot n \cdot N)$ of the forward selection in [68, 169]. In addition, the correlation matrix of all features needs to be computed in the approximation method of [104], which costs $O(n^2(\log n + N))$ operations.

Third, the searching method used by the DFL algorithm is also different from the greedy (forward) selection searching or the backward selection searching used by methods discussed above. In the DFL algorithm, the exhaustive search of all subsets with $\leq K$ features is guaranteed and can be terminated with the criterion of Theorem 2.2.2. In some data sets, $I(X_i; Y) = 0, \forall X_i \in \mathbf{X}$, as demonstrated by the example in Figure 6.5. The data set of this example is available at the supplementary website of this thesis. Existing feature selection methods based on mutual information [45, 68, 133, 169] will fail for this kind of data sets. For the example in Figure 6.5, it is shown that the three true relevant features, X_{21}, X_{29} and X_{60}, share smaller mutual information with Y than many other irrelevant features do. Actually, $\forall X_i \in \mathbf{V}, I(X_i; Y)$ should be zero in this data set since $\forall X_i \in \mathbf{V}$, X_i and Y are independent. But they are still larger than zero, although very small as shown in Figure 6.5, in practice. Hence, if a simple forward selection is used, existing feature selection methods in [45, 68, 133, 169] will choose X_{31}, which is an irrelevant feature, in the first round of the forward selection. Consider the selection criteria in Equation 6.2 [68, 169] and Equation 6.5 [133]. First, $I(X_i; X_j) = 0$, since $\forall X_i, X_j \in \mathbf{V}$, X_i and X_j are independent. Second,

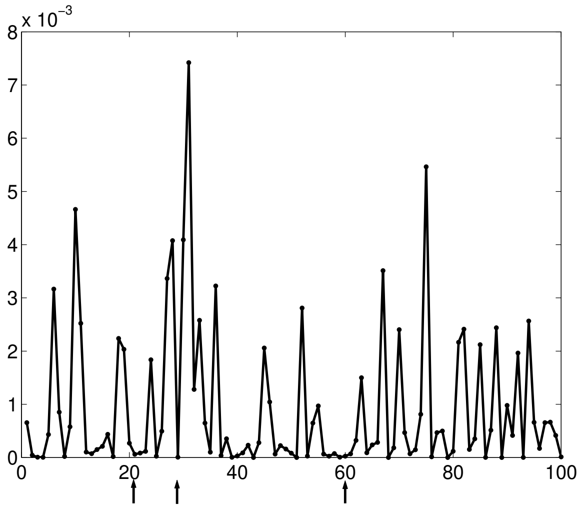


Figure 6.5: The $I(X_i;Y)$ in the data sets of 1000 samples generated with $Y = \neg(X_{21} \bigoplus X_{29} \bigoplus X_{60})$, and $\mathbf{V} = \{X_1, \ldots, X_{100}\}, \forall X_i, X_j \in \mathbf{V}, X_i$ and X_j are independent. The horizontal axis is the index of the features. The vertical axis is the $I(X_i;Y)$ shown in bits. The features pointed by the arrows are the relevant features.

$\forall X_i \in \mathbf{V}$, X_i and Y are independent. Consequently, the criteria in Equation 6.2 and Equation 6.5 will become $X_{(s)} = \arg\max_{X_i \in \mathbf{P}} I(X_i;Y)$. In later rounds, many other irrelevant features will be added to the candidate feature subset, which will also be incorrect, since they have larger mutual information than the relevant features do. However, the DFL algorithm can still find the correct feature subsets in polynomial time for this kind of data sets, since it guarantees the exhaustive searching of all subsets with $\leq K$ features and evaluates all selected features as a vector with Equation 2.11. For the example in Figure 6.5, the DFL algorithm successfully finds the correct feature subsets with less than 15 minutes in each fold of a 10 fold cross validation and obtains 100% prediction accuracy in the cross validation in our experiment.

Fourth and last, the methods in [68, 169] can only deal with binary features, however, the DFL algorithm can deal with multi-value discrete features as well.

In summary, three unique properties of the DFL algorithm are prerequisite to solve feature selection problems introduced by the data sets similar to that in Figure 6.5. First, the candidate features are considered as a vector to compute $I(\mathbf{U};Y)$. Second, $I(\mathbf{U};Y)$ is evaluated with respect to $H(Y)$ based on Theorem 2.2.2, which guarantees to find the correct feature subset. Last, the searching schema of the DFL algorithm guarantees to exhaustively search all subsets of $\mathbf{V}$ with $\leq K$ features, although the first round searching is greedy forward selection.

In Bayesian network fields, Friedman *et al.* [76] proposed an algorithm for learning Bayesian networks, called as the Sparse Candidate algorithm. The Sparse Candidate algorithm chooses parent nodes for the node under consideration with the similar idea of selecting those having strong relations. One of the statistics, $I(X_i; X_j, \mathbf{Pa}(X_i))$, used for evaluating the new parent node in the Sparse Candidate algorithm is similar to the $I(\mathbf{U}_{s-1}, X_i; Y)$ used in the DFL algorithm. However, there are two fundamental differences between the Sparse Candidate algorithm and the DFL algorithm. Firstly, the Sparse Candidate algorithm does not compare the statistic $I(X_i; X_j, \mathbf{Pa}(X_i))$ with the entropy of X_i, $H(X_i)$, but the DFL algorithm compares $I(\mathbf{U}_{s-1}, X_i; Y)$ with $H(Y)$. As discussed in Section 2.4.1, this comparison is critical. Since by comparing $I(\mathbf{U}_{s-1}, X_i; Y)$ with $H(Y)$, the DFL algorithm knows which set of variables is a *Markov Blanket* of Y, based on Theorem 2.3.1, and is complete in deciding the value of Y, based on Theorem 2.2.2. Consequently, the DFL algorithm avoids the exhaustive searching of all subsets of $\mathbf{V}$, which is NP-hard. Secondly, the forward selection is used in the Sparse Candidate algorithm. However, the DFL algorithm uses a better searching schema, which guarantees the exhaustive searching of all subsets of $\mathbf{V}$ with $\leq K$ features.

We briefly compare our approach and logistic regression methods at the result level because logistic regression does not belong to feature selection methods as discussed in Section 6.2. For the benchmark ALL data set [81], SLogReg [153] and BLogReg [40] made 4 and 5 prediction errors out of the total 72 samples in a *leave one out cross validation* with about 12 and 5 features. We also use the DFL algorithm to perform the same *leave one out cross validation* for this data set. The DFL algorithm makes 5 prediction errors with only one feature, D88422, with the parameters $\epsilon = 0.36$ and $K = 20$. The run times used by SLogReg and BLogReg in each fold are 2392.2s and 1.16s, as reported in [40]. In comparison, the DFL algorithm implemented with the Java language in our DFLearner software only uses 0.048s for the same computational task on a PC with Intel Pentium 4, 3.2GHz, CPU and 2GB memory, running Windows XP operating system. These suggest that the DFL algorithm performs better in terms of efficiency and model complexity than both SLogReg and BLogReg, but marginally loses to SLogReg in term prediction accuracy.

6.6 Conclusions

In this chapter, we analyze the feature selection problem from the information theory approach, with some concepts in graphical models. As discussed in Section 6.3, the **Problem 1** and **Problem 2** in Section 6.1 can be solved by comparing $I(\mathbf{U}; Y)$ with $H(Y)$, based on Theorem 2.2.2, 2.2.4 and 2.3.1. Then, based on Theorem 2.3.1, we use the DFL algorithm to efficiently find the *Markov Blanket* of Y.

The DFL algorithm is used as a filter feature subset selection method. We evaluate the features chosen by the DFL algorithm, and compare the results with those from existing feature subset selection methods, both filters and wrappers. The well-known classification methods show improvements of accuracies in most cases when applied to the essential attributes chosen by the DFL algorithm with significantly less time, as shown in Table 6.2 and in Figure 6.1 respectively. In Table 6.3, the comparisons with other feature selection methods show that the DFL algorithm chooses more informative and discriminatory features when compared to other filter methods, such as the CFS and CSE method.

Chapter 7

Conclusions

In this chapter, we will summarize the contributions of the thesis. We start by summarizing the comparisons between the ILA and other approaches of computational learning theory. Then, we summarize the contributions of this thesis. Finally, we analyze the limitations of the ILA and propose some future directions for continuing research.

7.1 Discussions

In this section, we compare the ILA with other approaches to computational learning theory and discuss their relations.

7.1.1 Comparison of The ILA and PAC Theory

First, we compare the ILA with the PAC theory. The PAC theory provides approximations of the original functions to a satisfactory threshold value with a high probability using noiseless training samples. The PAC theory also introduces the restrictions, i.e., there are fewer than or equal to k inputs, to the target functions to make the learning process be with polynomial complexity. However, the ILA can definitely find the original function in polynomial time under the same restrictions about the original function, as shown in Theorem 2.4.1. If there are no restrictions to the original functions, the ILA can also identify the original function when given enough samples, as discussed in Section 4.4.3. But the sample complexity grows exponentially, $N \sim (2^n + n\log_2 n)$ based on Theorem 3.2.1, in this situation. Thus, the learning process is NP-hard.

In the PAC theory, the learner updates the hypothesis when given a new sample. In this sense, the learner of the PAC theory makes use of the training data set from the dimension of the samples. However, it is a fact that for a given sample, the attributes are specified with deterministic values. In other words, there is no diversity, no randomness, no entropy for a given sample. In comparison, the ILA makes use of the training data sets from the dimension of attributes. In ILA, all samples are used as a collective set, in which attributes are not deterministic any more. The attributes can take different values with some probabilities. To put it another way, there appears diversity, randomness and entropy when the training data sets are used from the dimension of attributes. It is these attributes as random variables that carry the information about the concept under consideration. Thus, the DFL algorithm learns the models from the dimension of attributes.

7.1.2 Comparison of The ILA and VC Theory

Second, we compare the ILA with VC theory. The SVM methods select some samples in the training data sets as support vectors, and approximate the classification function with these vectors. There exist several shortcomings with the choice of support vectors. First, in the sense of choosing supporting vectors, the SVM methods do not reduce the dimensions of the classification problems. However, it is not the number of samples but the number of features (dimensions) of the data sets that mostly deteriorates the performances

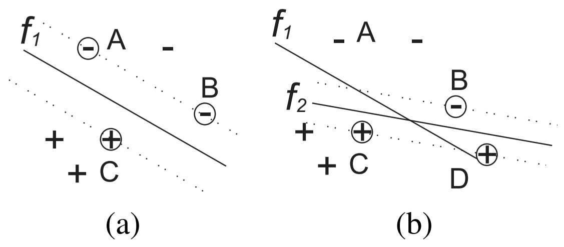


Figure 7.1: A schematic view of the unstableness of the classifiers obtained by the SVM algorithm. (a) The three samples enclosed in circles, {A,B,C}, are chosen as support vectors. The obtained classifier is the solid line f_1. (b) There appears a new sample D in the training data set. Sample A is not chosen as support vector any more. Instead, the new sample D is chosen as support vector, since the new set of support vectors, {B,C,D}, provides a maximum margin in the new data sets. The new classifier f_2 is quite different from the old classifier f_1.

of learning algorithms. From the statistical point of view, large training data set provides us a better knowledge repository. As shown in Theorem 2.2.4, the large sample size minimizes the undesirable effect from irrelevant features. Hence, it is more advisable to reduce the number of attributes than to reduce the number of training samples. Second, it is impossible to know whether a set of support vectors is sufficient to correctly find the classification functions. Third, the choice of support vectors often suffers from the risk of obtaining unstable classifiers, as shown in Figure 7.1. In Figure 7.1, we see that if there appear new training samples besides the old classification boundary, i.e., the old classifier, it is very likely that the new classification model is different from the old one, since the support vectors has changed in these cases.

In comparison, the DFL algorithm reduces the complexity of the classification problems from another dimension, i.e., from the number of attributes. Due to the merit of Theorem 2.2.2, the DFL algorithm can determine whether a set of EAs is a complete set of attributes needed to decide the class value. Precisely, the class attribute is an exact function of the

EAs when $\epsilon = 0$ based on Theorem 2.2.2. The models obtained by the DFL algorithm are also stable given enough samples, as to be discussed in Section 7.1.4.

7.1.3 Comparison of ILA and Bayesian Inference

Third, we compare the ILA with the Bayesian inference. Bayesian networks provide global conditional probability of the class attribute $P(Y|\mathbf{V})$, which is factorized with some local conditional probability encoded with the structure of the Bayesian networks. The global deciphering of the conditional independence between variables in Bayesian networks makes the learning and inference problems very difficult. As discussed in Section 1.1.3, both the learning and inference of Bayesian inference are NP-hard. However, there is no need to use expensive computational resource to obtain all the conditional independent relational between variables. It is the immediate parents $\mathbf{Pa}(Y)$ of the class attribute Y that give most information about Y. Based on Theorem 2.1.5, it is obvious that other variables except $\mathbf{Pa}(Y)$ will give no information about Y, since variables in $\mathbf{V}\backslash\mathbf{Pa}(Y)$ are conditional independent of Y given $\mathbf{Pa}(Y)$ in Bayesian networks.

Therefore, the ILA solves the problem with local conditional probability, $P(Y|\mathbf{U})$, as discussed in Section 2.1.2. To find complete $\mathbf{U}$, the ILA uses Theorem 2.2.2 and 2.2.4. Based on Theorem 2.2.2, $\mathbf{U}$ fully determines the value of Y if $I(\mathbf{U};Y) = H(Y)$. Based on Theorem 2.2.4, it is shown that other variables in $\mathbf{V} \setminus \mathbf{U}$ do not provide additional information about Y if $I(\mathbf{U};Y) = H(Y)$. The evaluation of $I(\mathbf{U};Y)$ with respect to $H(Y)$ is important, since it provides a criterion to determine whether the selected features are sufficient to decide Y or not. Consequently, the NP-hard exhaustive searching of all subsets of $\mathbf{V}$ is avoided in the ILA. For noiseless data sets, there would be only one rule for each $\mathbf{u}$. That means the probability of Y given $\mathbf{U}$ is one for each $\mathbf{u}$. For noisy data sets, the ILA provides a list of rules, $(\mathbf{u}, y)$, with the their counts in training data sets. These counts provide statistically meaningful predictions for the new samples. These count values are easily converted to conditional probabilities of Y given $\mathbf{U}$, as shown in Section 4.7.2.

In addition, there is an additional implicit assumption over the parent nodes of Y in Bayesian networks. Since the structures of Bayesian networks are DAGs (Directed Acyclic Graphs), it is assumed that there are no cycles among the parent nodes of Y. However, in our method, there is no such an assumption for the inputs of Y.

7.1.4 Comparison of ILA and Algorithmic Learning Theory

Fourth, we compare the ILA with the algorithmic learning theory. The algorithmic learning theory emphasizes that after some finite number of examples the system produces a correct hypothesis, which does not improve with any more given new samples.

Recall Theorem 2.4.1, in which the sample size is not explicitly required. Actually, when sample size is small, it is probable that the inputs of learned function $\mathbf{U}$ are not the inputs in the original function $\mathbf{X}$. See the example in Figure 4.1. When the sample size is only 20, the learned function is $X_i' = f(X_2, X_7, X_9)$, however the original function is $X_i' = f(X_1, X_2, X_3)$. The reason for this discrepancy lies in Theorem 2.1.4. Since when sample size is small, $I(\mathbf{Z}; Y) \neq 0, \forall \mathbf{Z} \in \mathbf{V} \backslash \mathbf{X}$, as discussed in Section 4.3. But when sample size is large, $I(\mathbf{Z}; Y)$ will become zero, based on Theorem 2.2.4. Hence, when sample size is growing, the models learned with the DFL algorithm will finally converge to a definite function, i.e., the original function, and will not change any more given additional samples. Based on Theorem 2.4.2, if the number of inputs of the original function is bounded by a constant, then the DFL algorithm can correctly find it in polynomial time given enough samples, with the lower bound defined by Theorem 3.2.2. This convergence of the DFL algorithm is essentially the convergence emphasized in the algorithmic learning theory.

For example, recall Theorem 4.4.4, in which we demonstrate that if enough samples are given, the DFL algorithm can correctly learn the original BLNs, i.e., a set of Boolean functions, in polynomial time. The BLN model learned with the DFL algorithm will not change after the number of learning samples grows to a certain value, with the lower bound defined by Theorem 3.2.1.

If the samples are noisy, the situation is much more complex, but Theorem 2.2.4 is also correct for noisy data sets. See example in Figure 2.3, where the $I(X_i;Y)$ of irrelevant features is probabilistically zero, for the noisy data sets with 2000 samples and 10% noise. This suggests that the functions estimated with the ϵ value method will contain the true inputs of the original functions with high probability, as demonstrated in Section 4.6.3.

7.2 Contributions

In this dissertation, we propose a new philosophy to understand learning as a procedure to acquire information of the concept under consideration, such as the class attribute in classification problems. Hence, we name it as information learning approach. The ILA is robust to noisy and efficient in practice, as demonstrated by the experimental results in Section 4.6.3, 4.6.2, 5.3.4 and 6.4.4. Generally, both the existing learning methods based on information theory and our approach belong to *Information Learning Approach*, although we use ILA to stand for our approach in the thesis.

Based on the ILA, we propose a new algorithm, called discrete function learning algorithm, to learn functions from data sets. The DFL algorithm is versatile and has been used to learn qualitative models of GRNs, to solve classification problems, and to find informative and discriminatory feature subsets for other classification algorithms.

We prove that the DFL algorithm can learn a bounded OR/AND Boolean function with $O(k\cdot(N+\log n)\cdot n)$ time compared to $o(N\cdot n^k)$ time before this work. For general Boolean functions, we have demonstrated that the DFL algorithm can still correctly find the original function in $O(k \cdot (N + \log n) \cdot n)$ steps with high probability. In Section 4.6.3, we have also shown that Boolean functions can efficiently and correctly be learned even when there are substantial amount, up to 20%, of noise in the learning data sets.

The searching method of the DFL algorithm extends the classical greedy algorithm [49]. The merit of our searching method lies in that it is greedy in the first round of searching as shown in Figure 2.6, but still guarantees to check all the subsets in the searching space $\mathcal{S}_K$, as shown in Figure 2.8. Although it takes more time in its worst complexity, the DFL algorithm can still find the correct models in polynomial time as proved in Theorem 2.4.2. As shown in Table 5.2, the DFL algorithm can find fairly good models for most problems after its first round greedy searching. But the exhaustive searching is also necessary in some situations, as shown in Figure 6.5.

We discussed the completeness of features with the ILA. We prove that complete feature subsets for classification problems can be obtained with the DFL algorithm.

The completeness of features makes it possible to overcome the *curse of dimensionality*, since many irrelevant and redundant features strictly proved to be useless for building accurate models are excluded. In many benchmark data sets, the DFL algorithm finds informative and discriminatory feature subsets for other classification methods.

We propose a new method to evaluate the performances of inference algorithm for learning qualitative models of GRNs. We propose to use the structure sensitivity as the criterion to evaluate the performance of inference algorithm for learning GRN models.

We implement the DFL algorithm for learning qualitative models, solving classification problems, and performing feature selection in an integrated software, called Discrete Function Learner.

7.3 Limitations and Future Work

In learning Boolean functions, there is one kind of functions that the DFL algorithm and many other algorithms cannot solve soundly. As discussed in Section 4.5, the DFL algorithm will use more time, but still polynomial, to find Boolean functions $Y = f(\mathbf{X})$ in which $I(X_i;Y) = 0, \forall X_i \in \mathbf{X}$, e.g., $Y = X_1 \bigoplus X_2$.

In Section 4.5, we discussed the constant functions. The DFL algorithm can correctly find the constant functions. But it introduces another problem which cannot be solved by the DFL algorithm. That is if the training samples are all with the same class attribute, then the entropy of the class attribute is zero. Consequently, the DFL algorithm cannot find the original function in this case. Nevertheless, this is an extreme case and unlikely to happen in practice. This problem can also be solved by drawing samples with the restriction of ergodicity, i.e., all possible samples are included in the training data sets.

The DFL algorithm may use more time to find the model when sample size is small. As shown in the example of Figure 4.1, when sample size is small, the irrelevant features may share more information with the concept under consideration. Consequently, the DFL algorithm will use more time to find the original functions, as shown in Section 4.6.3. However, these problems may be solved by increasing the sample size.

In the current implementation, the DFL algorithm will stop its searching when it finds the first feature subset to satisfy $I(\mathbf{X};Y) = H(Y)$ or $I(\mathbf{X};Y) \geq (1-\epsilon) \times H(Y)$ in the ϵ value method. In gene expression profiles or proteomic profiles, it is possible that there exist several subsets of features which are biologically meaningful and can give good prediction performance [134, 173]. In the future, the DFL algorithm can be used to find all feature vectors which capture $H(Y)$ or at least $(1-\epsilon) \times H(Y)$ with fewer than or equal to k features, and to find the prediction performances of the classifiers built over these feature vectors, by continuing the search process after the DFL algorithm finds the first satisfactory feature subset.

7.4 Perspective

As demonstrated in Section 4.6, the DFL algorithm can find most bounded Boolean functions in smaller than $o(N \cdot n^k)$ steps. This observation makes us propose the following conjecture about multi-value discrete functions.

Conjecture 7.4.1 *The DFL algorithm can correctly find most bounded multi-value discrete functions in smaller than $o(N \cdot n^k)$ steps with high probability.*

The DFL algorithm can be used to solve more problems than those shown in this thesis. Some promising future applications include, but not restricted to character recognition, image recognition, DNA sequence classification as coding or non-coding [185, 193], RNA splicing site classification [185, 193], association rule extraction from database, etc.

The searching schema of the DFL algorithm should be useful for solving many other combinatorial optimization problems. In computer science, many difficult problems are finally resorted to combinatorial searching, such as the learning problem of Bayesian networks [42, 43]. Many combinatorial searching problems are NP-complete and the greedy algorithm is often used to find the global or local optimal results. The searching method of the DFL algorithm provides an alternative choice to the greedy searching method for solving these problems.

We propose the information learning approach as a new approach to the computational learning theory by interpret learning as a procedure to acquire information about the concept under consideration. We propose the discrete function learning algorithm to implement the ILA. We have shown several applications of the DFL algorithm. We have demonstrated that bounded discrete functions are learnable in polynomial time with the DFL algorithm, even when the data sets are noisy. In general, Boolean functions are widely used in knowledge representation. In this sense, the DFL algorithm is a useful and efficient method for automatically learning knowledge from data sets.

In a more profound view, we hope that the ILA is useful in helping us to understand learning from the mechanistic perspective. In particular, as demonstrated by the lost PC and human DNA example in Section 1.2, there exist some essential attributes in describing a concept. If we can know these essential attributes with the aid of computational means from a small set of samples in prior, it will save enormous resources, like time and expense, when we are exploring to understand the concept.

Appendix A

The Proofs of the Theorems in Chapter 2

A.1 Jensen's Inequality

Jensen's inequality [122] deals with convex functions of only one variable. Let K be an interval in E^1 (Euclidean 1-space), and let $F(x)$ be a probability distribution function concentrated on K. Let K be the associated random variable, i.e., $P(X \leq x) = F(X)$. If the expectation $E(X)$ exists, and if $f(X)$ is a convex $\cup$[1], Jensen's inequality says that

$$E(f(X)) \geq f(E(X)). \tag{A.1}$$

Furthermore, if f is strictly convex, inequality (Equation A.1) is strict unless X is concentrated at a single point x_0, that is, $P(X = x_0) = 1$. Naturally, if f is instead convex $\cap$, the inequality reverses:

$$E(f(X)) \leq f(E(X)). \tag{A.2}$$

[1]The convex $\cup$ is also called concave and the convex $\cap$ is also called convex.

A.2 Proofs of the Theorems in Section 2.1

Theorem A.2.1 (Theorem 2.1.5) *Let* $\mathbf{V} = \{X_1, \dots, X_n\}$, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{X}$, $\mathbf{X}$ *and* $\mathbf{Z}$ *are independent. Given enough samples of* $\mathbf{V}$. *If* Y *and* $\mathbf{Z}$ *are conditional independent given* $\mathbf{X}$, *then empirical mutual information* $\hat{I}(Y; \mathbf{Z}) = 0$.

Proof. From Theorem 2.1.4, $p(\mathbf{x})$ can be correctly estimated from the enough samples.

Since Y and $\mathbf{Z}$ are conditional independent given $\mathbf{X}$, i.e.,

$$p(y|\mathbf{x}, \mathbf{z}) = p(y|\mathbf{x}). \tag{A.3}$$

Since $\mathbf{X}$ and $\mathbf{Z}$ are independent, we have

$$p(\mathbf{x}|\mathbf{z}) = p(\mathbf{x}) \tag{A.4}$$

Multiply both sides of Equation A.3 and Equation A.4, then sum up the two sides over all possible $\mathbf{x}$.

$$\begin{aligned} \sum_{\mathbf{x}} p(y|\mathbf{x}, \mathbf{z}) p(\mathbf{x}|\mathbf{z}) &= \sum_{\mathbf{x}} p(y|\mathbf{x}) p(\mathbf{x}) \\ p(y|\mathbf{z}) &= p(y). \end{aligned}$$

That is to say, when the probability of every instance $\mathbf{x}$ of $\mathbf{X}$ is known, Y and $\mathbf{Z}$ are independent. From Theorem 2.1.3, we have $\hat{I}(Y; \mathbf{Z}) = 0$. ■

Theorem A.2.2 (Theorem 2.1.6) *If* $Y = f(\mathbf{X})$, *then* $I(\mathbf{X}; Y) = H(Y)$.

Proof. Since Y is a deterministic function of $\mathbf{X}$, $p(Y = f(\mathbf{x})|\mathbf{X} = \mathbf{x}) = 1$, otherwise $p(Y|\mathbf{X}) = 0$. Hence $\forall \mathbf{x}$, $H(Y|\mathbf{X} = \mathbf{x}) = 0$. Thus, we get $I(\mathbf{X}; Y) = H(Y) - H(Y|\mathbf{X}) = H(Y)$. ■

The proof of Theorem A.2.2 can also be found in [83].

A.3 Proofs of the Theorems in Section 2.2

Theorem A.3.1 (Theorem 2.2.2, 1.2.2) *If the mutual information between* $\mathbf{X}$ *and* Y *is equal to the entropy of* Y*, i.e.,* $I(\mathbf{X};Y) = H(Y)$*, then* Y *is a function of* $\mathbf{X}$*.*

Proof. From the definition of mutual information, we obtain $I(\mathbf{X};Y) = H(Y) - H(Y|\mathbf{X}) = H(Y)$. Thus, $H(Y|\mathbf{X}) = 0$.

From the definition of conditional entropy,

$$\begin{aligned} H(Y|\mathbf{X}) &= -\sum_{\mathbf{x}}\sum_{y} p(\mathbf{x},y)\log p(y|\mathbf{x}) \\ &= -\sum_{\mathbf{x}}\sum_{y} p(\mathbf{x},y)\log \frac{p(\mathbf{x},y)}{p(\mathbf{x})} \\ &= 0 \end{aligned}$$

Hence, $p(\mathbf{x},y) = p(\mathbf{x})$. That is to say, for every value $\mathbf{x}$ of $\mathbf{X}$ with $p(\mathbf{x}) > 0$, there exists only one possible value y of Y that satisfy $p(\mathbf{x},y) = p(\mathbf{x}) > 0$. In other words, Y is a function of $\mathbf{X}$. ■

In [50, 178], it is proved that if $H(Y|X) = 0$, then Y is a function of X. Since $I(X;Y) = H(X) + H(Y|X)$, it is immediate to obtain Theorem A.3.1. Based on Theorem (or Corollary) A.3.2 to A.3.5, we propose Theorem A.3.6 (Theorem 2.2.3).

Theorem A.3.2 (Theorem 2.1.3) *For any discrete random vectors* Y *and* $\mathbf{Z}$*,* $I(Y;\mathbf{Z}) \geq 0$*. Moreover,* $I(Y;\mathbf{Z}) = 0$ *if and only if* Y *and* $\mathbf{Z}$ *are independent.*

Proof. From the definition of mutual information in Equation 2.4,

$$I(Y;\mathbf{Z}) = -\sum_{y}\sum_{\mathbf{z}} p(y,\mathbf{z}) \log \frac{p(y,\mathbf{z})}{p(y)p(\mathbf{z})}.$$

Hence, $I(Y;\mathbf{Z}) > 0$, and is zero when $\forall y, \mathbf{z}, p(y,\mathbf{z}) = p(y)p(\mathbf{z})$. In other words, $I(Y;\mathbf{Z}) = 0$, if and only if Y and $\mathbf{Z}$ are independent. ■

Proof of Theorem A.3.2 can also be found in [50]. Immediately from Theorem A.3.2, the following corollary is also correct.

Corollary A.3.1 (Corollary 2.1.1) $I(Y;\mathbf{Z}|\mathbf{X}) \geq 0$, *with equality if and only if* Y *and* $\mathbf{Z}$ *are independent given* $\mathbf{X}$.

Then, we discuss the relationship between $H(\mathbf{X})$ and $H(Y)$ when $Y = f(\mathbf{X})$ in the following theorem.

Theorem A.3.3 *If* $Y = f(\mathbf{X})$, *where* $\mathbf{X}$ *is a set of discrete random variables, then* $H(\mathbf{X}) \geq H(Y)$.

Proof. From the Definition of joint entropy in Equation 2.3,

$$\begin{aligned} H(\mathbf{X}, Y) &= H(\mathbf{X}) + H(Y|\mathbf{X}) = H(\mathbf{X}) \\ &= H(Y) + H(\mathbf{X}|Y) \geq H(Y) \end{aligned}$$

Thus, $H(\mathbf{X}) \geq H(Y)$. ■

Again, the proof of Theorem A.3.3 is also available in [50].

Theorem A.3.4 *Suppose that* $\mathbf{X}$ *is a set of discrete random variables, and* Y *are a finite discrete random variables. Then,* $min(H(\mathbf{X}), H(Y)) \geq I(\mathbf{X};Y) \geq 0$.

Proof. Since $I(\mathbf{X};Y) = H(\mathbf{X}) - H(\mathbf{X}|Y) = H(Y) - H(Y|\mathbf{X})$ and $H(\mathbf{X}|Y) \geq 0$, $H(Y|\mathbf{X}) \geq 0$. ■

Theorem A.3.5 (Theorem 2.2.1, 1.2.1) $I(\mathbf{X},\mathbf{Z};Y) \geq I(\mathbf{X};Y)$, *with equality if and only if* $p(y|\mathbf{x}) = p(y|\mathbf{x},\mathbf{z})$ *for all* $(\mathbf{x},y,\mathbf{z})$ *with* $p(\mathbf{x},y,\mathbf{z}) > 0$.

Proof.

$$\begin{aligned}
I(\mathbf{X};Y) - I(\mathbf{X},\mathbf{Z};Y) &= \sum_{\mathbf{x},y} p(\mathbf{x},y)\log\frac{p(\mathbf{x},y)}{p(\mathbf{x})p(y)} - \sum_{\mathbf{x},y,\mathbf{z}} p(\mathbf{x},y,\mathbf{z})\log\frac{p(\mathbf{x},y,\mathbf{z})}{p(\mathbf{x},\mathbf{z})p(y)} \\
&= \sum_{\mathbf{x},y,\mathbf{z}} p(\mathbf{x},y,\mathbf{z})\log\frac{p(y|\mathbf{x})}{p(y|\mathbf{x},\mathbf{z})} \\
&= E[\log\frac{p(y|\mathbf{x})}{p(y|\mathbf{x},\mathbf{z})}]
\end{aligned}$$

$\log(\cdot)$ is a convex $\cap$ function. Applying Jensen's inequality in Equation A.2, we have

$$\begin{aligned}
I(\mathbf{X};Y) - I(\mathbf{X},\mathbf{Z};Y) &\leq \log\sum_{\mathbf{x},y,\mathbf{z}} p(\mathbf{x},y,\mathbf{z})\frac{p(y|\mathbf{x})}{p(y|\mathbf{x},\mathbf{z})} \\
&= \log\sum_{\mathbf{x},y,\mathbf{z}} p(\mathbf{x},\mathbf{z})\cdot p(y|\mathbf{x}) \\
&= \log 1 = 0.
\end{aligned}$$

The conditions for equality are $\frac{p(y|\mathbf{x})}{p(y|\mathbf{x},\mathbf{z})} = 1$, i.e., for all $(\mathbf{x},y,\mathbf{z})$ with $p(\mathbf{x},y,\mathbf{z}) > 0$, $p(y|\mathbf{x}) = p(y|\mathbf{x},\mathbf{z})$. ■

Proof of Theorem A.3.5 can also be found in [122].

Theorem A.3.6 (Theorem 2.2.3) *If* $I(\mathbf{X};Y) = H(Y)$, $\mathbf{X} = \{X_{(1)},\ldots,X_{(k)}\}, \forall\mathbf{Z} \subseteq \mathbf{V}\backslash\mathbf{X}$, Y *and* $\mathbf{Z}$ *are conditional independent given* $\mathbf{X}$.

Proof. Let us consider $I(\mathbf{X}, \mathbf{Z}; Y)$, $\forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$. Firstly,

$$H(\mathbf{X}, \mathbf{Z}) = H(\mathbf{X}) + H(\mathbf{Z}|\mathbf{X}) \geq H(\mathbf{X}).$$

Secondly, from Theorem A.3.1, $Y = f(\mathbf{X})$. Then, from Theorem A.3.3, $H(\mathbf{X}) \geq H(Y)$. So, $H(\mathbf{X}, \mathbf{Z}) \geq H(\mathbf{X}) \geq H(Y)$. Thus, $min(H(\mathbf{X}, \mathbf{Z}), H(Y)) = H(Y)$. From Theorem A.3.4, we have

$$I(\mathbf{X}, \mathbf{Z}; Y) \leq min(H(\mathbf{X}, \mathbf{Z}), H(Y)) = H(Y) = I(\mathbf{X}; Y). \tag{A.5}$$

On the other hand, from Theorem A.3.5, we get

$$I(\mathbf{X}, \mathbf{Z}; Y) \geq I(\mathbf{X}; Y). \tag{A.6}$$

From both Equation A.5 and Equation A.6, we obtain $I(\mathbf{X}, \mathbf{Z}; Y) = I(\mathbf{X}; Y)$. Again from Theorem A.3.5, we get $p(y|\mathbf{x}, \mathbf{z}) = p(y|\mathbf{x})$. That is to say, Y and $\mathbf{Z}$ are conditional independent given $\mathbf{X}$. ■

Corollary A.3.2 (Corollary 2.2.1) *If* $I(\mathbf{X}; Y) = H(Y)$, $\mathbf{X} = \{X_{(1)}, \ldots, X_{(k)}\}$, $\forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$, *then* $I(Y; \mathbf{Z}|\mathbf{X}) = 0$.

Proof. Immediately from Theorem A.3.6 and Corollary A.3.1. ■

Theorem A.3.7 (Theorem 2.2.4) *Let* $\mathbf{V} = \{X_1, \ldots, X_n\}$, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{X}$, $\mathbf{X}$ *and* $\mathbf{Z}$ *are independent. Given enough samples of* $\mathbf{V}$. *If* $I(\mathbf{X}; Y) = H(Y)$, *then* $\forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$, *empirical mutual information* $\hat{I}(Y; \mathbf{Z}) = 0$.

Proof. From Theorem A.3.6, Y and $\mathbf{Z}$ are conditional independent given $\mathbf{X}$. Immediately from Theorem A.2.1, we have the results. ■

Theorem A.3.8 (Theorem 2.3.1) *If $I(\mathbf{X}; Y) = H(Y)$, $\mathbf{X} = \{X_{(1)}, \ldots, X_{(k)}\}$, then $\mathbf{X}$ is a Markov blanket of Y.*

Proof. From Theorem A.3.2, Y and $\mathbf{Z}$, $\forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$, are conditional independent given $\mathbf{X}$. Immediately from Definition 2.3.1, we have the result. ■

A.4 Proofs of the Theorems in Section 2.4

Theorem A.4.1 (Theorem 2.4.1) *Let $\mathbf{V} = \{X_1, \ldots, X_n\}$. The DFL algorithm can find a consistent function $Y = f(\mathbf{U})$ of maximum indegree K with $O((N + \log n) \cdot n^K)$ time in the worse case from $\mathbf{T} = \{(\mathbf{v}_i, y_i) : i = 1, 2, \ldots, N\}$.*

Proof. Since $|\mathbf{X}| = k$, $\mathbf{X}$ is included in the searching space $\mathcal{S}_K$, where $K \geq k$. Since $Y = f(\mathbf{X})$, $I(\mathbf{X}; Y) = H(Y)$ based on Theorem A.2.2. In the searching space $\mathcal{S}_K$, there exists at least one subset of $\mathbf{V}$, i.e., $\mathbf{X}$, which satisfies the criterion of Theorem A.3.1.

Since the maximum indegree of the function is $K \geq k$, the target subset $\mathbf{U}$ is included in the searching space $\mathcal{S}_K$. The DFL algorithm guarantees the check of all subsets in $\mathcal{S}_K$, which takes $O(N \cdot n^K)$ time. The sort step in line 7 of Table 2.2 will be executed for $O(n^{K-1})$ times, which takes $O(n^K \cdot \log n)$ time. Finally, based on Theorem A.3.1, the DFL algorithm will find a consistent function $Y = f(\mathbf{U})$ in $O((N + \log n) \cdot n^K)$ time in the worst case. ■

Theorem A.4.2 (Theorem 2.4.2) *Let $\mathbf{V} = \{X_1, \ldots, X_n\}$, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{X}$, $\mathbf{X}$ and $\mathbf{Z}$ are independent. Given enough samples of $\mathbf{V}$. The DFL algorithm can find the original generation function $Y = f(\mathbf{X})$ of maximum indegree K with $O((N + \log n) \cdot n^K)$ time in the worse case from $\mathbf{T} = \{(\mathbf{v}_i, y_i) : i = 1, 2, \ldots, N\}$.*

Proof. Since $|\mathbf{X}| = k$, $\mathbf{X}$ is included in the searching space $\mathcal{S}_K$, where $K \geq k$. Since $Y = f(\mathbf{X})$, $I(\mathbf{X}; Y) = H(Y)$ based on Theorem A.2.2. In the searching space $\mathcal{S}_K$, there exists at least one subset of $\mathbf{V}$, i.e., $\mathbf{X}$, which satisfies the criterion of Theorem A.3.1.

Based on Theorem A.3.7, $I(\mathbf{Z}; Y) = 0, \forall \mathbf{Z} \subseteq \mathbf{V} \backslash \mathbf{X}$. Hence, $\mathbf{X}$ is the only subset of $\mathbf{V}$ which satisfies the criterion of Theorem A.3.1 in $\mathcal{S}_K$.

The DFL algorithm guarantees the check of all subsets in $\mathcal{S}_K$, which takes $O(N \cdot n^K)$ time. The sort step in line 7 of Table 2.2 will be executed for $O(n^{K-1})$ times, which takes $O(n^K \cdot \log n)$ time. Finally, based on Theorem A.3.1, the DFL algorithm will find the original function $Y = f(\mathbf{X})$ in $O((N + \log n) \cdot n^K)$ time in the worst case. ■

Appendix B

The Proofs of the Theorems in Chapter 3

Theorem B.0.3 (Theorem 3.2.2) *$\Omega(b^k + k \log_b n)$ transition pairs are necessary in the worst case to identify the qualitative GRN models of maximum indegree $\leq k$ and the maximum base for variables $\leq b$.*

Proof. Firstly, we consider the number of mutually distinct qualitative GRN models of maximum indegree $\leq k$ and the maximum base for variables $\leq b$.

There are $\binom{n}{k} \approx n^k$ possible combinations of inputs for a given gene, and b^{b^k} possible discrete functions of base b for each gene. Thus, there are $\Omega((b^{b^k} \cdot n^k)^n)$ qualitative GRN models of maximum indegree $\leq k$ and the maximum base for variables $\leq b$.

Therefore, $\Omega(b^k \cdot n \log_2 b + nk \log_2 n)$ bits are required to represent a GRN model of maximum indegree $\leq k$ and of maximum indegree $\leq k$ and the maximum base for variables $\leq b$.

Lastly, we consider the number of transition pairs. For each transition pair, the information quantity is $n \log_2 b$ bits if the maximum base for variables $\leq b$. Hence, $\Omega(b^k + k \log_b n)$ transition pairs are required in the worst case. ■

Appendix C

The Proofs of the Theorems in Chapter 4

Theorem C.0.4 (Theorem 4.4.1) *Given enough samples for an OR BLN over* $\mathbf{V}$, *the mutual information between* $\forall X_{ij} \in \mathbf{Pa}(X_i^{'}) = \{X_{i1}, \ldots, X_{ik}\}$ *and* $X_i^{'}$ *is*

$$I(X_{ij}; X_i^{'}) = \frac{1}{2} - \frac{2^k - 1}{2^k} \log \frac{2^k - 1}{2^k} + \frac{2^{k-1} - 1}{2^k} \log \frac{2^{k-1} - 1}{2^k}. \tag{C.1}$$

Proof. The data sets are generated with the original Boolean functions of the BLNs. From Theorem 2.1.4, the empirical probability of $\mathbf{Pa}(X_i^{'})$ and $X_i^{'}$ in the Boolean functions $X_i^{'} = f_i(\mathbf{Pa}(X_i^{'}))$ will tend to be the probabilities in the truth table of f_i when sample size is large enough.

Without loss of generality, we consider $I(X_{i1}, X_i^{'})$. In the truth table of $X_i^{'}$, there are equal number of "0" and "1" for X_{i1}. Thus, $H(X_{i1}) = 1$.

In the truth table of $X_i^{'}$, there are 2^k lines totally. And in the column of $X_i^{'}$, there are only one "0", and $2^k - 1$ number of "1". Thus,

$$H(X_i^{'}) = -\frac{1}{2^k} \log \frac{1}{2^k} - \frac{2^k - 1}{2^k} \log \frac{2^k - 1}{2^k} = \frac{k}{2^k} - \frac{2^k - 1}{2^k} \log \frac{2^k - 1}{2^k}.$$

There are only three possible instances for the tuple $(X_{i1}, X_i^{'})$, i.e., $(0,0)$, $(0,1)$ and $(1,1)$. By counting the number of these patterns, and divided by the total number of lines, we have

$$H(X_{i1}, X_i^{'}) = -\frac{1}{2^k}\log\frac{1}{2^k} - \frac{2^{k-1}-1}{2^k}\log\frac{2^{k-1}-1}{2^k} - \frac{1}{2}\log\frac{1}{2}.$$

Therefore, from Equation 2.6, we obtain

$$\begin{aligned} I(X_{i1}; X_i^{'}) &= H(X_{i1}) + H(X_i^{'}) - H(X_{i1}, X_i^{'}) \\ &= \frac{1}{2} - \frac{2^k-1}{2^k}\log\frac{2^k-1}{2^k} + \frac{2^{k-1}-1}{2^k}\log\frac{2^{k-1}-1}{2^k}. \end{aligned}$$

■

Theorem C.0.5 (Theorem 4.4.2) *Given enough samples. In OR BLNs with maximum in-degree k over $\mathbf{V}$, $\forall 1 \le p \le k$, $X_{i(1)}, X_{i(2)}, \ldots, X_{i(p)} \in \mathbf{Pa}(X_i^{'})$, the mutual information between $\{X_{i(1)}, X_{i(2)}, \ldots, X_{i(p)}\}$ and $X_i^{'}$ is*

$$I(\{X_{i(1)}, X_{i(2)}, \ldots, X_{(p)}\}; X_i^{'}) = \frac{p}{2^p} - \frac{2^k-1}{2^k}\log\frac{2^k-1}{2^k} + \frac{2^{k-p}-1}{2^k}\log\frac{2^{k-p}-1}{2^k}. \quad \text{(C.2)}$$

Proof. From Theorem 2.1.4, the empirical probability of $\mathbf{Pa}(X_i^{'})$ and $X_i^{'}$ in the Boolean functions $X_i^{'} = f_i(\mathbf{Pa}(X_i^{'}))$ will tend to be the probabilities in the truth table of f_i when sample size is large enough.

Similar as in proof of Theorem C.0.4, we have

$$H(X_i^{'}) = \frac{k}{2^k} - \frac{2^k-1}{2^k}\log\frac{2^k-1}{2^k}. \quad \text{(C.3)}$$

Consider $X_i^{'} = X_1 + X_2 + X_3$ and $p = 2$ first. Without loss of generality, we derive $I(\{X_1, X_2\}; X_i^{'})$. For $H(X_1, X_2)$, as shown in Table C.1, there are $2^p = 4$ possible instances for the (X_1, X_2), i.e., (0,0), (0,1), (1,0), and (1,1). By counting the number of these patterns, and divided by the total number of lines, we get $H(X_1, X_2) = 2^p \times (-\frac{1}{2^p} \log \frac{1}{2^p}) =$ 2 (bits).

Table C.1: The truth table of $X_i^{'} = X_1 + X_2 + X_3$.

X_1	X_2	X_3	$X_i^{'}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Next, we derive $H(X_{i(1)}, \ldots, X_{i(p)}, X_i^{'})$. There are $2^p + 1 = 5$ possible instances for $(X_1, X_2, X_i^{'})$, i.e., (0,0,0), (0,0,1), (0,1,1), (1,0,1) and (1,1,1). Their probabilities are

$$\begin{aligned} p(0,0,0) &= \frac{1}{2^k}, \\ p(0,0,1) &= \frac{2^{k-p}-1}{2^k} = \frac{2^{k-2}-1}{2^k}, \\ p(0,1,1) &= \frac{2^{k-p}}{2^k} = \frac{2^{k-2}}{2^k}, \\ p(1,0,1) &= \frac{2^{k-p}}{2^k} = \frac{2^{k-2}}{2^k}, \\ p(1,1,1) &= \frac{2^{k-p}}{2^k} = \frac{2^{k-2}}{2^k}. \end{aligned} \tag{C.4}$$

Hence, we get

$$\begin{aligned} H(X_1, X_2, X_i^{'}) &= -\frac{1}{2^k} \log \frac{1}{2^k} - \frac{2^{k-2}-1}{2^k} \log \frac{2^{k-2}-1}{2^k} - 3 \times (\frac{2^{k-2}}{2^k} \log \frac{2^{k-2}}{2^k}) \\ &= \frac{3}{2} - \frac{1}{2^k} \log \frac{1}{2^k} - \frac{2^{k-2}-1}{2^k} \log \frac{2^{k-2}-1}{2^k}. \end{aligned}$$

Finally, from Equation 2.6, we have

$$\begin{aligned} I(\{X_1, X_2\}; X_i^{'}) &= H(X_1, X_2) + H(X_i^{'}) - H(X_1, X_2, X_i^{'}) \\ &= -\frac{1}{2^k}\log\frac{1}{2^k} - \frac{2^k-1}{2^k}\log\frac{2^k-1}{2^k} + 2 \\ &\quad -\frac{3}{2} + \frac{1}{2^k}\log\frac{1}{2^k} + \frac{2^{k-2}-1}{2^k}\log\frac{2^{k-2}-1}{2^k} \\ &= \frac{1}{2} - \frac{2^k-1}{2^k}\log\frac{2^k-1}{2^k} + \frac{2^{k-2}-1}{2^k}\log\frac{2^{k-2}-1}{2^k}. \end{aligned} \tag{C.5}$$

By generalizing 3 to p, we have

$$H(X_{i(1)}, \ldots, X_{i(p)}) = 2^p \times (-\frac{1}{2^p}\log\frac{1}{2^p}) = p(bits). \tag{C.6}$$

From Equation C.4, there are one instance of $(0, \ldots, 0, 0)$ for $(X_{i(1)}, X_{i(2)}, \ldots, X_{i(p)}, X_i^{'})$. There are $2^{k-p}-1$ instances of $(0, \ldots, 0, 1)$ for $(X_{i(1)}, X_{i(2)}, \ldots, X_{i(p)}, X_i^{'})$. There are $2^p - 1$ possible instances of $(X_{i(1)}, X_{i(2)}, \ldots, X_{i(p)}, X_i^{'})$ with the same probabilities of $\frac{2^{k-p}}{2^k}$. Hence,

$$\begin{aligned} H(X_{i(1)}, \ldots, X_{i(p)}, X_i^{'}) &= -\frac{1}{2^k}\log\frac{1}{2^k} - \frac{2^{k-p}-1}{2^k}\log\frac{2^{k-p}-1}{2^k} \\ &\quad -(2^p-1)(\frac{2^{k-p}}{2^k}\log\frac{2^{k-p}}{2^k}) \\ &= p + \frac{k}{2^k} - \frac{p}{2^p} - \frac{2^{k-p}-1}{2^k}\log\frac{2^{k-p}-1}{2^k}. \end{aligned} \tag{C.7}$$

Finally, by combing the Equation C.3, C.6, and C.7, we have the result, $\forall 1 \leq p \leq k$,

$$\begin{aligned} I(\{X_{i(1)}, \ldots, X_{i(p)}\}; X_i^{'}) &= H(X_{i(1)}, \ldots, X_{i(p)}) + H(X_i^{'}) - H(X_{i(1)}, \ldots, X_{i(p)}, X_i^{'}) \\ &= \frac{p}{2^p} - \frac{2^k-1}{2^k}\log\frac{2^k-1}{2^k} + \frac{2^{k-p}-1}{2^k}\log\frac{2^{k-p}-1}{2^k}. \end{aligned}$$

■

When $p = k$, from Theorem C.0.5, we have

$$I(X_{i(1)}, \dots, X_{i(k)}; X_i^{'}) = \frac{k}{2^k} - \frac{2^k - 1}{2^k} \log \frac{2^k - 1}{2^k} = H(X_i^{'}),$$

as shown in Equation C.3. This result is in accord with Theorem A.2.2, which validates Theorem C.0.5 from another aspect. For instance, as shown in Figure 4.2 (b), when $p = 6$, the $I(\{X_{i(1)}, \dots, X_{i(p)}\}; X_i^{'})$ in the curve for $k = 6$ is 0.116 bits, which should be equal to $H(X_i^{'})$ from Theorem A.2.2. Then, from Equation C.3, when $k = 6$, $H(X_i^{'}) = \frac{6}{64} - \frac{63}{64} \cdot \log_2(\frac{63}{64}) = 0.116$ bits too.

Theorem C.0.6 (Theorem 4.4.3) *Given enough samples. In OR BLNs with maximum indegree k over $\mathbf{V}$, $\forall 2 \leq p \leq k$,*
$I(\{X_{i(1)}, X_{i(2)}, \dots, X_{i(p)}\}; X_i^{'}) > I(\{X_{i(1)}, X_{i(2)}, \dots, X_{i(p-1)}\}; X_i^{'})$.

Proof. From Theorem C.0.5, we have

$$\begin{aligned} &I(\{X_{i(1)}, X_{i(2)}, \dots, X_{i(p)}\}; X_i^{'}) - I(\{X_{i(1)}, X_{i(2)}, \dots, X_{i(p-1)}\}; X_i^{'}) = \\ &\frac{p}{2^p} - \frac{p-1}{2^p - 1} + \frac{2^{k-p} - 1}{2^k} \log \frac{2^{k-p} - 1}{2^k} - \frac{2^{k-p+1} - 1}{2^k} \log \frac{2^{k-p+1} - 1}{2^k}. \end{aligned} \tag{C.8}$$

Since $\frac{p}{2^p} - \frac{p-1}{2^p-1} > 0$, and $\frac{2^{k-p}-1}{2^k} \log \frac{2^{k-p}-1}{2^k} - \frac{2^{k-p+1}-1}{2^k} \log \frac{2^{k-p+1}-1}{2^k} > 0$, therefore, Equation C.8 > 0. Then, we have the result. ■

Theorem C.0.7 (Theorem 4.4.4) *If given enough samples for an OR BLN with maximum indegree k over $\mathbf{V}$, then the DFL algorithm can identify the OR BLN in $O(k \cdot (N + logn) \cdot n^2)$ time strictly.*

Proof. First, consider the searching process in the first layer of the search graph like Figure 2.6. From Theorem A.3.7, we obtain $I(Z; X_i^{'}) = 0$ if $Z \in \mathbf{V} \backslash \mathbf{Pa}(X_i^{'})$. Meanwhile, from

Theorem C.0.4, $I(X_{ij}; X_i^{'}) > 0$. Thus, the true input values will be listed in front of the other variables Zs after the sort step in line 7 of Table 2.2. Without loss of generality, assume that $X_{ij} \in \mathbf{Pa}(X_i^{'})$ is listed in the first place.

In the following, the $\Delta_1(X_{ij})$ will be dynamically added to the second layer of the $\Delta Tree$. Now, consider the mutual information $I(X_{ij}, Z; X_i^{'})$, where Z is one of the variables in $\mathbf{V} \backslash X_{ij}$. First, if $Z \in \mathbf{V} \backslash \mathbf{Pa}(X_i^{'})$, from Theorem 2.2.4, $I(Z; X_i^{'}) = 0$. Since $\forall X_i, X_j \in \mathbf{V}$, X_i and X_j are independent variables, from Theorem 2.1.1, we get $H(X_i|X_j) = H(X_i)$. From Theorem 2.1.2, we have

$$\begin{aligned} I(X_{ij}, Z; X_i^{'}) &= I(Z; X_i^{'}) + I(X_{ij}; X_i^{'}|Z) \\ &= I(X_{ij}; X_i^{'}|Z) \\ &= H(X_{ij}|Z) - H(X_{ij}|X_i^{'}, Z) \\ &= H(X_{ij}) - H(X_{ij}|X_i^{'}) \\ &= I(X_{ij}; X_i^{'}). \end{aligned} \tag{C.9}$$

Second, if $Z \in \mathbf{Pa}(X_i^{'})$, from Theorem C.0.6, we have $I(\{X_{ij}, Z\}; X_i^{'}) > I(X_{ij}; X_i^{'})$. Combined with the results in Equation C.9, we have that $\forall Z \in \mathbf{Pa}(X_i^{'})$, $I(\{X_{ij}, Z\}; X_i^{'})$ is larger than the same measure when $Z \in \mathbf{V} \backslash \mathbf{Pa}(X_i^{'})$.

Therefore, in the second layer of the $\Delta Tree$, the combinations with two elements from $\mathbf{Pa}(X_i^{'})$ will be listed in front of other combinations. And so on so forth, until the DFL algorithm finally finds $\mathbf{Pa}(X_i^{'})$ in the kth layer of the $\Delta Tree$.

In the searching process, only $\sum_{i=0}^{k-1}(n-i) \approx kn$ subsets are visited by the DFL algorithm. Therefore, the complexity of the DFL algorithm becomes, $O(k \cdot (N + logn) \cdot n^2)$, where $logn$ is for sort step in line 7 of Table 2.2 and N is for the length of input table $\mathbf{T}$. ■

Corollary C.0.1 (Corollary 4.4.1) *If given enough samples for a generalized OR BLN*

Table C.2: The tuple $(X_{ij}, X_i^{'})$ and $(X_{ij}^{*}, X_i^{'})$, where k is two.

X_{ij}	$X_i^{'}$	X_{ij}^{*}	$X_i^{'}$
0	0	1	0
0	1	1	1
1	1	0	1
1	1	0	1

with maximum indegree k over $\mathbf{V}$, then the DFL algorithm can identify the BLN in $O(k \cdot (N + logn) \cdot n^2)$ time strictly.

Proof. We replace those X_{ij}s $\in \mathbf{Pa}(X_i^{'})$ that are taking their inverse with another variable X_{ij}^{*}, i.e., let $X_{ij}^{*} = \neg X_{ij}$, then the resulted BLN is an OR BLN. $H(X_i^{'})$ does not change in the new OR BLN.

To satisfy the criterion of Theorem 2.2.2, compare the mutual information $I(X_{ij}^{*}; X_i^{'})$ with $I(X_{ij}; X_i^{'})$. From Equation 2.6, we have

$$I(X_{ij}^{*}; X_i^{'}) = H(X_{ij}^{*}) + H(X_i^{'}) - H(X_{ij}^{*}, X_i^{'}).$$

$H(X_i^{'})$ remains the same value as the corresponding item in $I(X_{ij}; X_i^{'})$. In binary systems, there are only two states, i.e., "0" and "1". It is straightforward to obtain $H(X_{ij}^{*}) = H(X_{ij})$. Therefore, the only item changed in the $I(X_{ij}^{*}; X_i^{'})$ is the joint entropy $H(X_{ij}^{*}, X_i^{'})$. Next, we prove that $H(X_{ij}^{*}, X_i^{'}) = H(X_{ij}, X_i^{'})$.

Consider the tuple $(X_{ij}, X_i^{'})$. If we replace "0" of X_{ij} with "1" and vice versa, it becomes $(X_{ij}^{*}, X_i^{'})$, as shown in Table C.2. The three instances $(0, 0)$, $(0, 1)$ and $(1, 1)$ of $(X_{ij}, X_i^{'})$ change to $(1, 0)$, $(1, 1)$ and $(0, 1)$ of $(X_{ij}^{*}, X_i^{'})$. However, the probabilities (frequencies) of them are coincidently equal respectively. Thus, $H(X_{ij}^{*}, X_i^{'}) = H(X_{ij}, X_i^{'})$.

From Theorem C.0.7, the results are obtained. ■

Corollary C.0.2 (Corollary 4.4.2) *If given enough samples for an AND BLN with maximum indegree k over $\mathbf{V}$, then the DFL algorithm can identify the BLN in $O(k \cdot (N + logn) \cdot$*

n^2) *time strictly.*

Proof. In an AND BLN, the $P(X_i^{'} = 0)$ and $P(X_i^{'} = 1)$ is equal to the $P(X_i^{'} = 1)$ and $P(X_i^{'} = 0)$ in an corresponding OR BLN with the same $\mathbf{Pa}(X_i^{'})$ for all X_i. Therefore, $I(X_{ij}; X_i^{'})$ are the same as those of the corresponding OR BLN. From Theorem C.0.7, the result can be directly obtained. ■

Corollary C.0.3 (Corollary 4.4.3) *If given enough samples for a generalized AND BLN with maximum indegree k over $\mathbf{V}$, then the DFL algorithm can identify the BLN in $O(k \cdot (N + logn) \cdot n^2)$ time strictly.*

Proof. Similar to that of Corollary C.0.1. ■

Appendix D

The Detailed Settings

In Table D.1, the settings of the DFL algorithm for the data sets listed in Table 5.1 are given. The features chosen by the DFL algorithm for them are also given in Table D.2.

Table D.1: The settings of the DFL algorithm. To get optimal model, we change the epsilon value from 0 to 0.8, with a step of 0.01. For each epsilon value, we train a model with the DFL algorithm, then do corresponding test for the selected data sets. In our implementation of the DFL algorithm, the process to choose optimal model can be automatically fulfilled. For those data sets whose testing processes are performed with the cross validation, the number of features k and the number of the rules r in the classifier are from the most frequently obtained classifiers.

		Performances			Settings		Classifiers		
	Data Set	Accuracy (%)	Time(s)	n	K	ϵ	k	r	$k/n(\%)$
1	Lenses	75.0	0.03	4	4	0.26	3	12	75.0
2	Iris	96.0	0.01	4	4	0.08	2	6	50.0
3	Monk1	100.0	0.01	6	6	0	3	35	50.0
4	Monk2	73.8	0.02	6	6	0.21	6	168	100.0
5	Monk3	97.2	0.01	6	6	0.64	2	17	33.3
6	LED	74.9	0.08	7	7	0.29	7	207	100.0
7	Nursery	93.1	20.21	8	8	0.13	5	541	62.5
8	Breast	95.0	0.20	9	9	0.05	3	185	33.3
9	Wine	98.3	0.01	13	13	0.04	4	29	30.8
10	Credit	88.3	0.01	15	15	0.57	2	11	13.3
11	Vote	95.7	0.22	16	16	0.11	4	41	25.0
12	Zoo	92.8	1.24	16	16	0	5	21	31.3
13	ImgSeg	90.6	0.01	19	15	0.16	3	41	15.8
14	Mushroom	100.0	11.45	22	10	0	4	96	18.2
15	LED+17	75.4	0.83	24	20	0.31	7	286	29.2
16	Ionosphere	94.9	0.11	34	20	0.12	6	96	17.6
17	Chess	97.4	13.30	36	20	0.01	19	844	52.8
18	Anneal	99.0	0.22	38	20	0.04	5	44	13.2
19	Lung	62.5	0.10	56	20	0.44	2	12	3.6
20	Ad	95.0	42.80	1558	20	0.23	6	104	0.4
21	ALL	94.1	0.02	7129	20	0.3	1	3	0.014
22	DLBCL	95.5	0.01	7129	20	0.52	1	4	0.014
23	MLL	100.0	0.48	12582	20	0.06	2	11	0.016
24	Ovarian	98.8	0.31	15154	20	0.29	1	4	0.007
	average	91.0	3.82	1829	14	0.20	4	117	31.5

Table D.2: The features chosen by the DFL algorithm.

			Feature Index	
	Data Set	k	Discretized Data	Continuous Data
1	Lenses	2	1,3,4	1,3,4
2	Iris	2	3,4	3,4
3	Monk1	3	1,2,5	1,2,5
4	Monk2	6	1,2,3,4,5,6	1,2,3,4,5,6
5	Monk3	2	2,5	2,5
6	LED	7	1,2,3,4,5,6,7	1,2,3,4,5,6,7
7	Nursery	5	1,2,5,7,8	1,2,5,7,8
8	Breast	3	1,3,6	1,3,6
9	Wine	4	1,7,10,13	1,7,10,13
10	Credit	2	4,9	4,9
11	Vote	4	3,4,7,11	3,4,7,11
12	Zoo	5	3,4,6,9,13	3,4,6,9,13
13*	ImgSeg	3	1,13,15	2,17,19
14	Mushroom	4	5,20,21,22	5,20,21,22
15	LED+17	7	1,2,3,4,5,6,7	1,2,3,4,5,6,7
16	Ionosphere	6	3,5,6,12,21,27	3,5,6,12,21,27
17	Chess	19	1,3,4,6,7,10,15,16,17,18,20, 21,23,24,30,32,33,34,35	1,3,4,6,7,10,15,16,17,18,20, 21,23,24,30,32,33,34,35
18	Anneal	5	3,5,8,9,12	3,5,8,9,12
19	Lung	2	6,20	6,20
20	Ad	6	1,2,3,352,1244,1400	1,2,3,352,1244,1400
21*	ALL	1	234	1882
22*	DLBCL	1	55	506
23*	MLL	2	709,1550	2592,5083
24*	Ovarian	1	839	1679

* For these 5 data sets, the discretized data sets is preprocessed by deleting the features with only 1 value. Hence, the index of discrete data sets are different from those for continuous data sets. The match tables for these feature index are available at the supplementary website of this thesis.

Appendix E

The Extended Main Steps of The DFL Algorithm

The extended main steps of the DFL algorithm are listed in Table E.1. As shown in line 21 to 23 of Table E.2, $\mathbb{R}[\mathbf{X}[0]][\mathbf{X}[1]]$ will be set to true if all Δ supersets of $\mathbf{X}$ have been checked. When the DFL algorithm is trying to check $\mathbf{X}$ later, it will find that $\mathbf{X}$ and its Δ supersets have been checked in line 3 and 14 respectively. For instance, $\{A, B\}$ has been checked when the DFL algorithm is examining the supersets of $\{A\}$. Later, when it trying supersets of $\{B\}$, the DFL algorithm will revisit $\{A, B\}$, but this time the computation of $\{A, B\}$ and its supersets will be saved by checking $\mathbb{R}$. Thus, the computation of all subsets with equal to or more than 2 elements is reduce to half of the original computation without introducing redundancy matrix $\mathbb{R}$, as analyzed in Section 2.8.2.

Table E.1: The extended version of the DFL algorithm.

```
    Algorithm: DFL($\mathbf{V}, K, \mathbf{T}$)
    Input: a list $\mathbf{V}$ with $n$ variables, indegree $K$,
    $\mathbf{T} = \{(\mathbf{v}_i, y_i) : i = 1, \cdots, N\}$. $\mathbf{T}$ is global.
    Output: $f$
    Begin:
1     $\mathbb{R} \leftarrow$ boolean[n][n];        //initialize $\mathbb{R}$, default value is false
2     $L \leftarrow$ all single element subsets of $\mathbf{V}$;
3     $\Delta Tree.FirstNode \leftarrow L$;
4     calculate $H(Y)$;        //from $\mathbf{T}$
5     $D \leftarrow 1$;        //initial depth
6*    $f = Sub(Y, \Delta Tree, H(Y), D, K)$;
7   return $f$;
    End
```

* $Sub()$ is a subroutine listed in Table E.2.

Table E.2: The extended version of the subroutine of the DFL algorithm.

```
     Algorithm: $Sub(Y, \Delta Tree, H, D, K)$
     Input: variable $Y$, $\Delta Tree$, entropy $H(Y)$, current depth $D$, maximum indegree $K$
     Output: function table for $Y$, $Y = f(\mathbf{X})$
     Begin:
1    $L \leftarrow \Delta Tree.DthNode$;
2    for every element $\mathbf{X} \in L$ {
3*       if ( ($|\mathbf{X}| == 2$) && ($\mathbb{R}[\mathbf{X}[0]][\mathbf{X}[1]] ==$ true || $\mathbb{R}[\mathbf{X}[1]][\mathbf{X}[0]] ==$ true) ) {
4            continue;    //if $\mathbf{X}$ has been checked, continue to check next element in $L$
         }
5        calculate $I(\mathbf{X}; Y)$;    //from $\mathbf{T}$
6        if($I(\mathbf{X}; Y) == H$) {    //from Theorem 2.2.2
7            extract $Y = f(\mathbf{X})$ from $\mathbf{T}$;
8            return $Y = f(\mathbf{X})$ ;
         }
9        else if $(D == K)$ && $\mathbf{X}$ is the last element in $L$) {
10           return "Fail(Y)";
         }
     }
11   sort $L$ according to $I$;
12   for every element $\mathbf{X} \in L$ {
13       if($D < K$){
14           if ( ($|\mathbf{X}| == 2$) && ($\mathbb{R}[\mathbf{X}[0]][\mathbf{X}[1]] ==$ true || $\mathbb{R}[\mathbf{X}[1]][\mathbf{X}[0]] ==$ true) ) {
15               continue;
             }
16           $D \leftarrow D + 1$;
17           $\Delta Tree.DthNode \leftarrow \Delta_1(\mathbf{X})$;
18           $f = Sub(Y, \Delta Tree, H, D, K)$;
19           if $f \neq$ "Fail(Y)" {
20               return $f$;
             }
21           else if ($|\mathbf{X}| == 2$){
22               $\mathbb{R}[\mathbf{X}[0]][\mathbf{X}[1]] \leftarrow$ true;    //if all $\Delta(\mathbf{X})$ have been checked, set $\mathbb{R}[\cdot][\cdot]$ to true.
23               continue;
             }
         }
     }
24   return "Fail(Y)";    //fail to find function for $Y$
     End
```

* The && and || represent the logic AND and OR operations respectively.

Appendix F

Notation

The following uniform notation is used throughout the thesis. We use capital letters to represent discrete random variables, such as X and Y; lower case letters to represent an instance of the random variables, such as x and y; bold capital letters, like $\mathbf{X}$, to represent a vector; and lower case bold letters, like $\mathbf{x}$, to represent an instance of $\mathbf{X}$. The cardinality of $\mathbf{X}$ is represented with $|\mathbf{X}|$.

	DATA
X, X_i	a random variable, a feature in $\mathbf{V}$ or a gene in GRN models over $\mathbf{V}$
x, x_i	an instance of X, X_i
Y	the concept, the class attribute in the classification problems
$\mathbf{V}$	the feature set, with $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$
$\mathbf{v}$	an instance of $\mathbf{V}$, $\mathbf{v} = \{x_1, x_2, \ldots, x_n\}$
n	the number of features
N	the number of samples in the training data sets
$\mathbf{X}$	the inputs of the original function
$\|\mathbf{X}\|$	the cardinality of $\mathbf{X}$

$\|\|\mathbf{X}\|\|$	the number of difference instances of $\mathbf{X}$
	PARAMETERS OF THE DFL ALGORITHM
K	the expected cardinality of the EAs for the classification, or the expected indegree of the GRN models
ϵ	the ϵ value, one of the two parameters of the DFL algorithm
$\mathbf{T}$	the training data sets, $\mathbf{T} = \{(\mathbf{v}_i, \mathbf{v}_i^{'}) : i = 1, 2, \ldots, N\}$ for learning GRN models, or $\mathbf{T} = \{(\mathbf{v}_i, y_i) : i = 1, 2, \ldots, N\}$ for classification
$\Delta_i(A)$	the Δ i supersets of A, $\Delta_i(A) = \bigcup\{\mathbf{X} \cup \{A\}\}, \forall \|\mathbf{X}\| = i$
$\mathcal{L}_i$	the ith searching layer of $\mathbf{V}$, $\mathcal{L}_i = \bigcup \mathbf{X}, \forall \|\mathbf{X}\| = i$
$\mathcal{S}_K$	the searching space of functions with bounded number of inputs K, $\mathcal{S}_K = \bigcup_{i=1}^{K} \mathcal{L}_i$
	RESULTS
k	the actual cardinality of the EAs, $k \leq K$
$\mathbf{U}$	the set of EAs chosen by the DFL algorithm
$\mathbf{U}_{s-1}$	the features already chosen at $s-1$ step
f	the obtained function, the classification model
$X_i^{'}$	the expression level of gene X_i at time step $t+1$
f_i	the obtained function for a gene X_i in the GRN models, $X_i^{'} = f_i$
$\mathbf{F}$	the regulatory function set in GRN models, $\mathbf{F} = \{f_1, f_2, \ldots, f_n\}$
B	A Boolean network model, $B(\mathbf{V}, \mathbf{F})$
	DISTRIBUTION AND INFORMATION THEORY
$P(X)$	the probability distribution of random variable X
$p(x)$	the probability of X taking the value x, i.e., $P(X = x)$
$H(X)$	the entropy of X
$I(X;Y)$	the mutual information between X and Y

Appendix G

Abbreviations

The following abbreviations are used throughout the thesis.

Abb.	Full Text	Chapter/Section
1NN	1-Nearest-Neighbor	1.2, 2.7
ALL	Acute Lymphoblastic Leukemia	3.1.1, 5.4
AML	Acute Myeloid Leukemia	3.1.1, 5.4
BLNs	Boolean Networks	3.2.1, 4
C4.5	C4.5 classification algorithm	5.3.2
CAP	adenylyl Cylclase Associated Protein	3.1.3
CPT	Conditional Probability Table	2.1.2, 4.7
DAG	Directed Acyclic Graph	2.1.2, 7.1.3
DBN	Dynamic Bayesian Networks	4.7
DFL	Discrete Function Learning	1.2, 2.4
DLBCL	Diffused Large B-Cell Lymphomas	5.4
EAs	Essential Attributes	2.2
FL	Follicular Lymphoma	5.4
GLF	Generalized Logical Formalism	3.2.1

Abb.	Full Text	Chapter/Section
GRNs	Gene Regulatory Networks	3.1
ILA	Information Learning Approach	1.2, 2.2
kNN	k-Nearest-Neighbors	5.3.2
MLL	leukemia with rearranged *MLL* gene	5.4
NB	Naive Bayes classification algorithm	5.3.2
PAC	Probably Approximately Correct	1.1.1
PBNs	Probabilistic Boolean Networks	4.7
PCR	Polymerase Chain Reaction	3.1.1
PLDE	Piecewise Linear Differential Equation	3.2.1
SELDI	Surface-Enhanced Laser Desorption/Ionization	5.1
SVM	Support Vector Machine	1.1.2, 5.3.2
TFs	Transcription Factors	3.1.3
TOF	Time-Of-Flight	5.1
VC	Vapnik Chervonenkis	1.1.2

Bibliography

[1] http://en.wikipedia.org/wiki/law_of_large_numbers.

[2] http://search.eb.com/eb/article-9384410.

[3] http://stat-www.berkeley.edu/ stark/sticigui/text/gloss.htm#law_of_large_numbers.

[4] B.-L. Adam, Y. Qu, J. W. Davis, M. D. Ward, M. A. Clements, L. H. Cazares, O. J. Semmes, P. F. Schellhammer, Y. Yasui, Z. Feng, and J. Wright, George L. Serum Protein Fingerprinting Coupled with a Pattern-matching Algorithm Distinguishes Prostate Cancer from Benign Prostate Hyperplasia and Healthy Men. *Cancer Res*, 62(13):3609–3614, 2002.

[5] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Mach. Learn.*, 6:37–66, 1991.

[6] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Proceedings of Pacific Symposium on Biocomputing '99*, volume 4, pages 17–28, Hawaii, HI, 1999.

[7] T. Akutsu, S. Miyano, and S. Kuhara. Algorithm for identifying boolean networks and related biological networks based on matrix multiplication and fingerprint function. *Journal of Computation Biology*, 7(3/4):331–343, 2000.

[8] T. Akutsu, S. Miyano, and S. Kuhara. Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, 16(8):727–734, 2000.

[9] T. Akutsu, S. Miyano, and S. Kuhara. A simple greedy algorithm for finding functional relations: efficient implementation and average case analysis. *Theor. Comput. Sci.*, 292(2):481–495, 2003.

[10] C. F. Aliferis, I. Tsamardinos, and A. Statnikov. Hiton: a novel markov blanket algorithm for optimal variable selection. In *AMIA Annu Symp Proc.*, pages 21–25, 2003.

[11] A. Alizadeh, M. Eisen, R. Davis, C. Ma, I. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. Tran, X. Yu, J. Powell, L. Yang, G. Marti, T. Moore, J. J. Hudson, L. Lu, D. Lewis, R. Tibshirani, G. Sherlock, W. Chan, T. Greiner, D. Weisenburger, J. Armitage, R. Warnke, R. Levy, W. Wilson, M. Grever, J. Byrd, D. Botstein, P. Brown, and L. Staudt. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.

[12] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS*, 96(12):6745–6750, 1999.

[13] R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. *Lecture Notes in Computer Science*, 2034:19–32, 2001.

[14] I. Alvarez-Garcia and E. A. Miska. Microrna functions in animal development and human disease. *Development*, 132:4653–62, 2005.

[15] V. Ambros. The functions of animal micrornas. *Nature*, 431:350–5, 2004.

[16] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988.

[17] D. Angluin and C. H. Smith. Inductive inference: Theory and methods. *ACM Comput. Surv.*, 15(3):237–269, 1983.

[18] S. Armstrong, J. Staunton, L. Silverman, R. Pieters, M. den Boer, M. Minden, S. Sallan, E. Lander, T. Golub, and S. Korsmeyer. Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, 30:41–47, 2002.

[19] M. Arnone and E. Davidson. The hardwiring of development: organization and function of genomic regulatory systems. *Development*, 124:1851–1864, 1997.

[20] C. A. Ball, G. Sherlock, H. Parkinson, P. Rocca-Sera, C. Brooksbank, H. C. Causton, D. Cavalieri, T. Gaasterland, P. Hingamp, F. Holstege, M. Ringwald, P. Spellman, J. Stoeckert,

Christian J., J. E. Stewart, R. Taylor, A. Brazma, and J. Quackenbush. Standards for Microarray Data. *Science*, 298(5593):539b–, 2002.

[21] Z. Bar-Joseph, G. K. Gerber, T. I. Lee, N. J. Rinaldi, J. Y. Yoo, F. Robert, D. B. Gordon, E. Fraenkel, T. S. Jaakkola, R. A. Young, and D. K. Gifford. Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, 21:1337–1342, 2003.

[22] D. P. Bartel. Micrornas: Genomics, biogenesis, mechanism, and function. *Cell*, 116:281–297, 2004.

[23] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5:537–550, 1994.

[24] S. Becker, L. H. Cazares, P. Watson, H. Lynch, O. J. Semmes, R. R. Drake, and C. Laronga. Surfaced-Enhanced Laser Desorption/Ionization Time-of-Flight (SELDI-TOF) Differentiation of Serum Protein Profiles of BRCA-1 and Sporadic Breast Cancer. *Ann Surg Oncol*, 11(10):907–914, 2004.

[25] D. Beer, S. Kardia, C. Huang, T. Giordano, A. Levin, D. Misek, L. Lin, G. Chen, T. Gharib, D. Thomas, M. Lizyness, R. Kuick, S. Hayasaka, J. Taylor, M. Iannettoni, M. Orringer, and S. Hanash. Gene-expression profiles predict survival of patients with lung adenocarcinoma. *Nature Med*, 8:816–824, 2002.

[26] F. Bergadano and L. Saitta. On the error probability of boolean concept descriptions. In *Proceedings of the 4th European Working Session on Learning EWSL-89*, pages 25–36, 1989.

[27] F. Berzal, J.-C. Cubero, D. Sánchez, and J. M. Serrano. Art: A hybrid classification model. *Mach. Learn.*, 54(1):67 – 92, 2004.

[28] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. J. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerson. Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. *PNAS*, 98:1379013795, 2001.

[29] A. Birkendorf, E. Dichterman, J. Jackson, N. Klasner, and H. U. Simon. On restricted-focus-of-attention learnability of boolean functions. *Mach. Learn.*, 30(1):89–123, 1998.

[30] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[31] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[32] H. Bolouri and E. H. Davidson. Modeling transcriptional regulatory networks. *BioEssays*, 24:1119–1129, 2002.

[33] A. Brazma, A. Robinson, G. Cameron, and M. Ashburner. One-stop shop for microarray data. *Nature*, 403:699–700, 2000.

[34] R. J. Britten and E. H. Davidson. Gene Regulation for Higher Cells: A Theory. *Science*, 165(3891.):349–357, 1969.

[35] N. H. Bshouty. Exact learning boolean functions via the monotone theory. *Inf. Comput.*, 123(1):146–153, 1995.

[36] W. Buntine. *A Theory of Learning Classification Rules*. PhD thesis, Sydney, February 1990.

[37] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[38] M. Carey and S. T. Smale. *Transcriptional Regulation in Eukaryotes: Concepts, Strategies and Techniques*. Cole Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 2001.

[39] J. Case, S. Jain, and F. Stephan. Vacillatory and bc learning on noisy data. *Theor. Comput. Sci.*, 241(1-2):115–141, 2000.

[40] G. C. Cawley and N. L. C. Talbot. Gene selection in cancer classification using sparse logistic regression with Bayesian regularization. *Bioinformatics*, 22(19):2348–2355, 2006.

[41] V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. John Wiley & Sons, Inc., New York, NY, 1998.

[42] D. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks is np-hard. Technical report, Microsoft Research, 1994.

[43] D. M. Chickering, C. Meek, and D. Heckerman. Large-sample learning of Bayesian networks is hard. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence,* Acapulco, Mexico, pages 124–133. Morgan Kaufmann, 2003.

[44] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73, 1998.

[45] T. W. S. Chow and D. Huang. Estimating optimal features subsets using efficient estimation of high-dimensional mutual information. *IEEE Trans Neural Networks*, 16:213–224, 2005.

[46] L. S. Cohen, P. F. Escobar, C. Scharm, B. Glimco, and D. A. Fishman. Three-dimensional power doppler ultrasound improves the diagnostic accuracy for ovarian cancer prediction. *Gynecologic Oncology*, 82:40–48, 2001.

[47] G. Cooper. Computational complexity of probabilistic inference using bayesian belief networks (research note). *Artificial Intelligence*, 42:393–405, 1990.

[48] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9:309, 1992.

[49] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. MIT Press, 2001.

[50] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, NY, 1991.

[51] N. Cristianini, J. Shawe-Taylor, and J. Kandola. On kernel target alignment. In *Proceedings of the Neural Information Processing Systems, NIPS'01*, pages 367–373, Cambridge, MA, 2001.

[52] P. Dagum and M. Luby. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial Intelligence*, 60:141–153, 1993.

[53] E. Davidson. *Genomic regulatory systems: Development and evolution*. Academic Press, San Diego, California, 2001.

[54] E. Davidson and M. Levin. Gene Regulatory Networks Special Feature: Gene regulatory networks. *PNAS*, 102(14):4935–, 2005.

[55] E. Davidson, D. McClay, and L. Hood. Regulatory gene networks and the properties of the developmental process. *PNAS*, 100(4):1475–1480, 2003.

[56] E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C.-H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena, O. Otim, C. T. Brown, C. B. Livi, P. Y. Lee, R. Revilla, A. G. Rust, Z. j. Pan, M. J. Schilstra, P. J. C. Clarke, M. I. Arnone, L. Rowen, R. A. Cameron, D. R. McClay, L. Hood, and H. Bolouri. A Genomic Regulatory Network for Development. *Science*, 295(5560):1669–1678, 2002.

[57] H. de Jong, J. Geiselmann, G. Batt, C. Hernandez, and M. Page. Qualitative simulation of the initiation of sporulation in *B. subtilis*. Technical Report 4527, INRIA, 2002.

[58] H. de Jong, M. Page, C. Hernandez, and J. Geiselmann. Qualitative simulation of genetic regulatory networks: Method and application. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 67–73, San Mateo, CA, 2001. Morgan Kaufmann.

[59] J. DeRisi, V. Iyer, and P. Brown. Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale. *Science*, 278(5338):680–686, 1997.

[60] P. D'haeseleer, S. Liang, and R. Somogyi. Genetic networks inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[61] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *CSB '03: Proceedings of the IEEE Computer Society Conference on Bioinformatics*, page 523, Washington, DC, USA, 2003. IEEE Computer Society.

[62] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202, Tahoe City, CA, 1995.

[63] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wily & Sons, Inc., New York, NY, second edition, 2000.

[64] D. J. Duggan, M. Bittner, Y. Chen, P. Meltzer, and J. M. Trent. Expression profiling using cdna microarrays. *Nature Genetics*, 21:10–14, 1999.

[65] S. Dumais, J. Platt, D. Hecherman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th International Conference on Information and Knowledge Management*, pages 148–155, 1998.

[66] T. Eiter, T. Ibaraki, and K. Makino. Decision lists and related boolean functions. *Theor. Comput. Sci.*, 270(1-2):493–524, 2002.

[67] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI-93*, pages 1022–1027, Chambery, France, 1993.

[68] F. Fleuret. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.*, 5:1531–1555, 2004.

[69] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. Witten. Data mining in bioinformatics using Weka. *Bioinformatics*, 20(15):2479–2481, 2004.

[70] D. A. S. Fraser. *Statistics, An Introduction*. John Wiley & Sons, Inc., New York, NY, 1958.

[71] M. Frazier, M. Knotek, B. Wold, E. Uberbacher, E. Branscomb, M. Buchanan, J. Trewhella, L. Makowski, D. Vaughan, A. Reeves, M. Broido, M. Colvin, O. Edward, D. Thomassen, J. Zhou, J. Tiedle, B. Mansfield, D. Casey, J. Wyrick, and A. Adamson. *Genomes to Life: Accelerating Biological Discovery*. 2001.

[72] M. E. Frazier, G. M. Johnson, D. G. Thomassen, C. E. Oliver, and A. Patrinos. Realizing the Potential of the Genome Revolution: The Genomes to Life Program. *Science*, 300(5617):290–293, 2003.

[73] N. Friedman. Inferring Cellular Networks Using Probabilistic Graphical Models. *Science*, 303(5659):799–805, 2004.

[74] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29:131–168, 1997.

[75] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3/4):601–620, 2000.

[76] N. Friedman, I. Nachman, and D. Pe'er. Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proceedings of the 15th Annual Conference*

on Uncertainty in Artificial Intelligence, UAI-99, pages 206–215, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[77] T. Furey, N. Cristianini, N. Duffy, D. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[78] R. Ghosh and C. J. Tomlin. Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. *Lecture Notes in Computer Science*, 2034:232–246, 2001.

[79] L. Glass and S. Kauffman. The logical analysis of continuous non-linear biochemical control networks. *Journal of Theoretical Biology*, 39:103–129, 1973.

[80] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[81] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286(5439):531–537, 1999.

[82] G. J. Gordon, R. V. Jensen, L.-L. Hsiao, S. R. Gullans, J. E. Blumenstock, S. Ramaswamy, W. G. Richards, D. J. Sugarbaker, and R. Bueno. Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer and Mesothelioma. *Cancer Res*, 62(17):4963–4967, 2002.

[83] R. M. Gray. *Entropy and Information Theory*. Springer Verlog, 1991.

[84] M. Gretzer, D. Chan, C. van Rootselaar, J. Rosenzweig, S. Dalrymple, L. Mangold, A. Partin, and R. Veltri. Proteomic analysis of dunning prostate cancer cell lines with variable metastatic potential using seldi-tof. *Prostate*, 60:325–331, 2004.

[85] M. S. Halfon and A. M. Michelson. Exploring genetic regulatory networks in metazoan development: Methods and models. *Physiol Genomics*, 10:131–143, 2002.

[86] M. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, Waikato University, Department of Computer Science, 1999.

[87] M. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, 15:1–16, 2003.

[88] R. Hamming. Error detecting and error correcting codes. *Bell System Technical Jounral*, 9:147–160, 1950.

[89] A. Hartemink, D. Gifford, T. Jaakkola, and R. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Proceedings of Pacific Symposium on Biocomputing '2001*, volume 6, pages 422–433, 2001.

[90] A. Hartemink, D. Gifford, T. Jaakkola, and R. Young. Combining location and expression data for principled discovery of genetic regulatory network models. In *Proceedings of Pacific Symposium on Biocomputing '2002*, volume 7, pages 437–449, 2002.

[91] D. Haussler. Probably approximately correct learning. In *National Conference on Artificial Intelligence*, pages 1101–1108, 1990.

[92] M. Hilario, A. Kalousis, J. Prados, and P.-A. Binz. Data mining for mass-spectra based diagnosis and biomarker discovery. *Drug Discovery Today: BIOSILICO*, 2:214–222, 2004.

[93] M. L. Howard and E. H. Davidson. cis-regulatory control circuits in development. *Developmental Biology*, 271(1):109–118, 2004.

[94] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. TANE: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, 1999.

[95] T. Ideker, V. Thorsson, and R. Karp. Discovery of regulatory interactions through perturbation: Inference and experimental design. In *Pacific Symposium on Biocomputing*, volume 5, pages 302–313, 2000.

[96] H. Issaq, T. Veenstra, T. Conrads, and D. Felschow. The seldi-tof ms approach to proteomics: protein profiling and biomarker identification. *Biochem Biophys Res Commun*, 292:587–592, 2002.

[97] S. Jain, D. Osherson, J. S. Royer, and A. Sharma. *Systems That Learn, 2nd Edition: An Introduction to Learning Theory*. The MIT Press, Cambridge, MA, 1999.

[98] C. Jie and G. Russell. Comparing bayesian network classifiers. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 101–108, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[99] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[100] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning*, pages 121–129, 1994.

[101] M. I. Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1998.

[102] S. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

[103] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[104] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning*, pages 284–292, 1996.

[105] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja. On learning gene regulatory networks under the boolean network model. *Mach. Learn.*, 52(1-2):147–167, 2003.

[106] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *National Conference on Artificial Intelligence*, pages 223–228, 1992.

[107] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistics Society, Series B*, 50(2):157–224, 1988.

[108] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and

R. A. Young. Transcriptional Regulatory Networks in Saccharomyces cerevisiae. *Science*, 298(5594):799–804, 2002.

[109] M. Levine and E. Davidson. From the Cover. Gene Regulatory Networks Special Feature: Gene regulatory networks for development. *PNAS*, 102(14):4936–4942, 2005.

[110] J. Li, H. Liu, J. Downing, A. Yeoh, and L. Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients. *Bioinformatics*, 19(1):71–78, 2003.

[111] J. Li and L. Wong. Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. *Bioinformatics*, 18(5):725–734, 2002.

[112] L. Li, H. Tang, Z. Wu, J. Gong, M. Gruidl, J. Zou, M. Tockman, and R. A. Clark. Data mining techniques for cancer detection using serum proteomic profiling. *Artificial Intelligence in Medicine*, 32:71–83, 2004.

[113] S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithms for genetic network architectures. In *Proceedings of Pacific Symposium on Biocomputing '98*, volume 3, pages 18–29, Maui, HI, 1998.

[114] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2:285–318, 1988.

[115] H. Liu, J. Li, and L. Wong. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, 13:51–60, 2002.

[116] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of the 13th International Conference on Machine Learning*, pages 319–327, 1996.

[117] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, 2002.

[118] W. Luo and O. Schulte. Mind change efficient learning. *Information and Computation*, 204(6):989–1011, 2006.

[119] Y. Maki, D. Tominaga, M. Okamoto, S. Watanabe, and Y. Eguchi. Development of a system for the inference of large scale genetic networks. In *Pacific Symposium on Biocomputing*, volume 6, pages 446–458, 2001.

[120] H. Mannila and K. Raiha. On the complexity of inferring functional dependencies. *Discrete Applied Mathematics*, 40:237–243, 1992.

[121] H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies from relations. *Data Knowl. Eng.*, 12(1):83–99, 1994.

[122] R. J. McEliece. *The Theory of Information and Coding: A Mathematical Framework for Communication*, volume 3 of *Encyclopedia of Mathematics and Its Applications*. Addison-Wesley Publishing Company, Reading, MA, 1977.

[123] D. Mehta and V. Raghavan. Decision tree approximations of boolean functions. *Theor. Comput. Sci.*, 270(1-2):609–623, 2002.

[124] L. Mendoza, D. Thieffry, and E. Alvarez-Buylla. Genetic control of flower morphogenesis in arabidopsis thaliana: A logical analysis. *Bioinformatics*, 15(7-8):593–606, 1999.

[125] T. Mestl, E. Plahte, and S. Omholt. A mathematical framework for describing and analysing gene regulatory networks. *Journal of Theoretical Biology*, 176:291–300, 1995.

[126] J.-P. Mira, V. Benard, J. Groffen, L. C. Sanders, and U. G. Knaus. Endogenous, hyperactive Rac3 controls proliferation of breast cancer cells by a p21-activated kinase-dependent pathway. *PNAS*, 97(1):185–189, 2000.

[127] K. Murphy and S. Mian. Modelling gene expression data using dynamic bayesian networks. Technical report, Computer Science Division, U.C., at Berkeley, Berkeley, CA, 1999.

[128] A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[129] I. M. Ong, J. D. Glasner, and D. Page. Modeling regulatory pathways in e. coli from time series expression profiles. *Bioinformatics*, 18:S241–S248, 2002.

[130] C. P. Paweletz, J. W. Gillespie, D. K. Ornstein, N. L. Simone, M. R. Brown, K. A. Cole, Q.-H. Wang, J. Huang, N. Hu, T.-T. Yip, W. E. Rich, E. C. Kohn, W. M. Linehan, T. Weber, P. Taylor, M. R. Emmert-Buck, L. A. Liotta, and E. F. P. III. Rapid protein display profiling

of cancer progression directly from human tissue using a protein biochip. *Drug Development Research*, 49:34–42, 2000.

[131] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[132] J. Pearl. *Causality: models, reasoning, and inference*. Cambridge University Press, Cambridge, UK, 2000.

[133] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238, 2005.

[134] E. Petricoin, A. Ardekani, B. Hitt, P. Levine, V. Fusaro, S. Steinberg, G. Mills, C. Simone, D. Fishman, E. Kohn, and L. Liotta. Use of proteomic patterns in serum to identify ovarian cancer. *The Lancet*, 359:572–577, 2002.

[135] E. F. Petricoin, D. K. Ornstein, and L. A. Liotta. Clinical proteomics: Applications for prostate cancer biomarker discovery and detection. *Urologic Oncology: Seminars and Original Investigations*, 22:322–328, 2004.

[136] E. F. Petricoin, K. C. Zoon, E. C. Kohn, J. C. Barrett, and L. A. Liotta. Clinical proteomics: translating benchside promise into bedside reality. *Nature Reviews Drug Discovery*, 1:683–695, 2002.

[137] J. Platt. *Fast training of support vector machines using sequential minimal optimization*, chapter 12, pages 185–208. MIT Press, Cambridge, MA, 1999.

[138] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. Kim, L. C. Goumnerova, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, and T. R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415:436–442, 2002.

[139] M. Ptashne and A. Gann. *Genes & Signals*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 2002.

[140] J. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Francisco, CA, 1993.

[141] M. Reich, K. Ohm, M. Angelo, P. Tamayo, and J. P. Mesirov. GeneCluster 2.0: an advanced toolset for bioarray analysis. *Bioinformatics*, 20(11):1797–1798, 2004.

[142] R. L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987.

[143] L. Sanchez and D. Thieffy. A logical analysis of the drosophila gap genes. *Journal of Theoretical Biology*, 211:115–141, 2001.

[144] L. Sanchez, J. van Helden, and D. Thieffy. Establishment of the dorso-ventral pattern during embryonic development of drosophila melanogaster: A logical analysis. *Journal of Theoretical Biology*, 189:377–389, 1997.

[145] M. Schena. *Microarray Analysis*. John Wiley & Sons, Inc., New York, NY, 2003.

[146] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray. *Science*, 270(5235):467–470, 1995.

[147] J. C. Schlimmer and J. Richard H. Granger. Incremental learning from noisy data. *Mach. Learn.*, 1(3):317–354, 1986.

[148] I. Schmulevich, O. Yli-Harja, and J. Astola. Inference of genetic regulatory networks under the best-fit extension paradigm. In *Nonlinear Signal and Image Processing, NSIP 2001*, 2001.

[149] B. Scholkopf, P. Simard, A. Smola, and V. Vapnik. *Advances in Neural Information Processing Systems 10*, chapter Prior Knowledge in Support Vector Kernels, pages 640–646. MIT Press, Cambridge, MA, 1998.

[150] E. Segal, H. Wang, and D. Koller. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, 19(90001):264i–272, 2003.

[151] C. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, IL, 1963.

[152] A. Sharma, F. Stephan, and Y. Ventsov. Generalized notions of mind change complexity. *Information and Computation*, 189(2):235–262, 2004.

[153] S. K. Shevade and S. S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003.

[154] M. Shipp, K. Ross, P. Tamayo, A. Weng, J. Kutok, R. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. Pinkus, T. Ray, M. Koval, K. Last, A. Norton, T. Lister, J. Mesirov, D. Neuberg, E. Lander, J. Aster, and T. Golub. Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature Medicine*, 8:68–74, 2002.

[155] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.

[156] I. Simon, J. Barnett, N. Hannett, C. Harbison, T. Rinaldi, N.J. amd Volkert, J. Wyrick, J. Zeitlinger, D. Gifford, T. Jaakkola, and R. Young. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 166:679–708, 2001.

[157] M. Snyder and M. Gerstein. Defining Genes in the Genomics Era. *Science*, 300(5617):258–260, 2003.

[158] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[159] D. Thieffry and R. Thomas. Qualitative analysis of gene networks. In *Proceedings of Pacific Symposium on Biocomputing '98*, volume 3, pages 77–88, 1998.

[160] R. Thomas and R. d'Ari. *Biological Feedback*. CRC Press, Boca Raton, Florida, 1990.

[161] R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks: I. biological rool of feedback loops and pratical use of the concept of the loop-characteristic state. *Bulletin of Mathematical Biology*, 57(2):247–276, 1995.

[162] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. D. Jong, S. Džeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. V. de Welde, W. Wenzel,

J. Wnek, and J. Zhang. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CS-91-197, Pittsburgh, PA, 1991.

[163] I. Tsamardinos and C. Aliferis. Towards principled feature selection: Relevancy, filters and wrappers. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, FL, 2003.

[164] I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and sample efficient discovery of markov blankets and direct causal relations. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678, New York, NY, USA, 2003. ACM Press.

[165] L. Valiant. Learning disjunctions and conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 560–565, 1985.

[166] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[167] L. van 't Veer, H. Dai, M. van de Vijver, Y. He, A. Hart, M. Mao, H. Peterse, K. van der Kooy, M. Marton, A. Witteveen, G. Schreiber, R. Kerkhoven, C. Roberts, P. Linsley, R. Bernards, and S. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530 – 536, 2002.

[168] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, NY, 1998.

[169] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *Proceedings of the 9th IEEE International Conference on Computer Vision, ICCV 2003*, pages 281–288, Nice, France, 2003. IEEE Computer Society.

[170] Z. Wang, C. Yip, Y. Ying, J. Wang, X.-Y. Meng, L. Lomas, T.-T. Yip, and E. T. Fung. Mass Spectrometric Analysis of Protein Markers for Ovarian Cancer. *Clin Chem*, 50(10):1939–1942, 2004.

[171] D. A. Wigle, I. Jurisica, N. Radulovich, M. Pintilie, J. Rossant, N. Liu, C. Lu, J. Woodgett, I. Seiden, M. Johnston, S. Keshavjee, G. Darling, T. Winton, B.-J. Breitkreutz, P. Jorgenson, M. Tyers, F. A. Shepherd, and M. S. Tsao. Molecular Profiling of Non-Small Cell Lung Cancer and Correlation with Disease-free Survival. *Cancer Res*, 62(11):3005–3008, 2002.

[172] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, CA, 1999.

[173] J. D. Wulfkuhle, L. A. Liotta, and E. F. Petricoin. Early detection: Proteomic applications for the early detection of cancer. *Nature Review Cancer*, 3:267–275, 2003.

[174] E. Xing, M. Jordan, and R. Karp. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 601–608. Morgan Kaufmann Publishers Inc., 2001.

[175] C. H. Yang, L. J. Chen, and Z. R. Sung. Genetic regulation of shoot development in arabidopsis: role of the *EMF* genes. *Dev Biol*, 169:421–435, 1995.

[176] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[177] S. Yaramakala and D. Margaritis. Speculative markov blanket discovery for optimal feature selection. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 809–812, Washington, DC, USA, 2005. IEEE Computer Society.

[178] R. W. Yeung. *A First Course in Information Theory*. Kluwer Academic/Plenum Publishers, New York, NY, 2002.

[179] C.-H. Yuh, H. Bolouri, J. Bower, and E. Davidson. *Computational Modeling of Genetic and Biochemical Networks*, chapter A logical model of cis-regulatory control in eukaryotic system, pages 73–100. MIT Press, Cambridge, MA, 2001.

[180] C.-H. Yuh, H. Bolouri, and E. H. Davidson. Genomic Cis-Regulatory Logic: Experimental and Computational Analysis of a Sea Urchin Gene. *Science*, 279(5358):1896–1902, 1998.

[181] P. D. Zamore and B. Haley. Ribo-gnome: The Big World of Small RNAs. *Science*, 309(5740):1519–1524, 2005.

[182] G. L. Zhang, J. C. Tong, Z. H. Zhang, Y. Zheng, J. T. August, C. K. Kwoh, and V. Brusic. Computational models for identifying promiscuous hla-b7 binders based on information theory and support vector machine. In *Proceedings of the International Conference on Biomedical and Pharmaceutical Engineering 2006 (ICBPE 2006)*, page to appear, 2006.

[183] Y. Zheng, W. Hsu, M. L. Lee, and L. Wong. Exploring essential attributes for detecting microrna precursors from background sequences. In *2006 VLDB Workshop on Data Mining in Bioinformatics*, volume 4316 of *Lecture Notes in Computer Science*, pages 131–145, Seoul, Korea, 2006.

[184] Y. Zheng and C. K. Kwoh. Dynamic algorithm for inferring qualitative models of gene regulatory networks. In *Proceedings of the 3rd Computational Systems Bioinformatics Conference, CSB 2004*, pages 353–362, Stanford, CA, 2004. IEEE Computer Society Press.

[185] Y. Zheng and C. K. Kwoh. Identifying decision lists with the discrete function learning algorithm. In *Proceedings of the 2nd International Conference on Artificial Intelligence in Science And Technology, AISAT 2004*, pages 30–35, Hobart, Australia, 2004.

[186] Y. Zheng and C. K. Kwoh. Improved mdl score for learning of bayesian networks. In *Proceedings of the 2nd International Conference on Artificial Intelligence in Science And Technology, AISAT 2004*, pages 98–103, Hobart, Australia, 2004.

[187] Y. Zheng and C. K. Kwoh. Reconstructing boolean networks from noisy gene expression data. In *Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision, ICARCV 2004*, pages 1049–1054, Kunming, China, 2004.

[188] Y. Zheng and C. K. Kwoh. A feature vector selection method for cancer classification. In *Proceedings of International Conference of Bioinformatics, Bioinfo2005*, pages 23–28, Busan, Korea, 2005.

[189] Y. Zheng and C. K. Kwoh. Identifying simple discriminatory gene vectors with an information theory approach. In *Proceedings of the 4th Computational Systems Bioinformatics Conference, CSB 2005*, pages 12–23, Stanford, CA, 2005.

[190] Y. Zheng and C. K. Kwoh. Cancer classification with microrna expression patterns found by an information theory approach. *Journal of Computers, accepted*, 2006.

[191] Y. Zheng and C. K. Kwoh. Dynamic algorithm for inferring qualitative models of gene regulatory networks. *International Journal of Data Mining and Bioinformatics*, 1:111–137, 2006.

[192] Y. Zheng and C. K. Kwoh. Informative microrna expression patterns for cancer classification. In J. Li, Q. Yang, and A.-H. Tan, editors, *Data Mining for Biomedical Applications, PAKDD 2006 Workshop, BioDM 2006*, volume 3916 of *Lecture Notes in Computer Science*, pages 143–154, Singapore, 2006.

[193] Y. Zheng, C. K. Kwoh, W. Hsu, M. L. Lee, and L. Wong. The discrete function learner for bioinformatics. Technical report, Nanyang Technological University, 2006.

[194] X. Zhou, K.-Y. Liu, and S. T. Wong. Cancer classification and prediction using logistic regression with bayesian gene selection. *Journal of Biomedical Informatics*, 37(4):249–259, 2004.

[195] J. Zhu and T. Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427–443, 2004.