1

PM-1 3 1/2" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Number of copies: _____

Order Number: _____

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

INDUCTIVE LEARNING IN NETWORK FAULT DIAGNOSIS

by

Guang Shi, B. Eng.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements of the degree of

Master of Engineering

Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada, K1S 5B6
Jan 25, 1994

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file / votre référence

Your file / votre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-92930-8

Canada

Name

GILANG SHI

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Electronics and Electrical

SUBJECT TERM

0544

SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0412
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0246
History of 0520
Home Economics 0278
Industrial 0221
Language and Literature 0276
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language
General 0679
Ancient 0284
Linguistics 0290
Modern 0291
Literature
General 0401
Classical 0294
Comparative 0295
Medieval 0297
Modern 0298
African 0316
American 0591
Asian 0305
Canadian (English) 0352
Canadian (French) 0355
English 0593
Germanic 0311
Latin American 0312
Middle Eastern 0315
Romance 0313
Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion
General 0318
Biblical Studies 0321
Clergy 0319
History of 0320
Philosophy of 0322
Theology 0469

SOCIAL SCIENCES

American Studies 0323
Anthropology
Archaeology 0324
Cultural 0326
Physical 0327
Business Administration
General 0310
Accounting 0272
Banking 0770
Management 0454
Marketing 0338
Canadian Studies 0385
Economics
General 0501
Agricultural 0503
Commerce-Business 0505
Finance 0508
History 0509
Labor 0510
Theory 0511
Folklore 0358
Geography 0366
Gerontology 0351
History
General 0578

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0333
United States 0337
History of Science 0585
Law 0398
Political Science
General 0615
International Law and Relations 0616
Public Administration 0617
Recreation 0814
Social Work 0452
Sociology
General 0626
Criminology and Penology 0627
Demography 0938
Ethnic and Racial Studies 0631
Individual and Family Studies 0628
Industrial and Labor Relations 0629
Public and Social Welfare 0630
Social Structure and Development 0700
Theory and Methods 0344
Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture
General 0473
Agronomy 0285
Animal Culture and Nutrition 0475
Animal Pathology 0476
Food Science and Technology 0359
Forestry and Wildlife 0478
Plant Culture 0479
Plant Pathology 0480
Plant Physiology 0817
Range Management 0777
Wood Technology 0746

Biology

General 0306
Anatomy 0287
Biostatistics 0308
Botany 0309
Cell 0379
Ecology 0329
Entomology 0353
Genetics 0369
Limnology 0793
Microbiology 0410
Molecular 0307
Neuroscience 0317
Oceanography 0416
Physiology 0423
Radiation 0821
Veterinary Science 0778
Zoology 0472

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleogeography 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences
General 0566
Audiology 0300
Chemotherapy 0992
Dentistry 0367
Education 0350
Hospital Management 0769
Human Development 0758
Immunology 0982
Medicine and Surgery 0564
Mental Health 0347
Nursing 0569
Nutrition 0570
Obstetrics and Gynecology 0380
Occupational Health and Therapy 0354
Ophthalmology 0381
Pathology 0571
Pharmacology 0419
Pharmacy 0572
Physical Therapy 0382
Public Health 0573
Radiology 0574
Recreation 0575

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry
General 0485
Agricultural 0749
Analytical 0486
Biochemistry 0487
Inorganic 0488
Nuclear 0738
Organic 0490
Pharmaceutical 0491
Physical 0494
Polymer 0495
Radiation 0754
Mathematics 0405
Physics
General 0605
Acoustics 0986
Astronomy and Astrophysics 0606
Atmospheric Science 0608
Atomic 0748
Electronics and Electricity 0607
Elementary Particles and High Energy 0798
Fluid and Plasma 0759
Molecular 0609
Nuclear 0610
Optics 0752
Radiation 0756
Solid State 0611
Statistics 0463

Applied Sciences

Applied Mechanics 0346
Computer Science 0984

Engineering
General 0537
Aerospace 0538
Agricultural 0539
Automotive 0540
Biomedical 0541
Chemical 0542
Civil 0543
Electronics and Electrical 0544
Heat and Thermodynamics 0348
Hydraulic 0545
Industrial 0546
Marine 0547
Materials Science 0794
Mechanical 0548
Metallurgy 0743
Mining 0551
Nuclear 0552
Packaging 0549
Petroleum 0765
Sanitary and Municipal 0554
System Science 0790
Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

The undersigned recommend to the Faculty of Graduate Studies and Research
acceptance of the thesis

INDUCTIVE LEARNING
IN
NETWORK FAULT DIAGNOSIS

submitted by Guang Shi, B. Eng.
in partial fulfillment of the requirements for
the degree of Master of Engineering

Thesis Supervisor

Thesis Supervisor

Chair, Department of Systems and Computer Engineering

Carleton University
Jan 25, 1994

Abstract

Machine learning is a study of computational methods for acquiring knowledge and improving problem solving ability. Among multiple machine learning techniques, Inductive learning techniques have been widely studied and applied into real world applications including the network management domain. However, most inductive learning algorithms have the general problem of overfitting training examples, which makes the rules generated from inductive learning algorithms less reliable and accurate.

This thesis addresses the general overfitting problem and applications of inductive learning techniques in network fault diagnosis. Specifically, this thesis presents the development of a new pruning algorithm which can reduce the overfitting problem. As part of the Pegasus fault diagnosis project, this thesis also investigates the application of the CN2 rule induction algorithm in the development of router normal traffic patterns for the BNR wide area network.

Key words:

Machine learning, Inductive learning, Overfitting problem, Pruning techniques, Network management, Network fault diagnosis, Knowledge acquisition bottleneck

Acknowledgments

I would like to express my special thanks to Professor Bernie Pagurek, Professor Nick Dawes and Dr. Julian Craddock for their consistent guidance and support during the course of my research. Without their supervision, my work would not be of the same quality. I wish to thank Mr. Lyle Zary and Mr. Quyen Bo for their editorial assistance. I also wish to acknowledge the research funding from the Telecommunications Research Institute of Ontario (TRIO). Finally, I wish to thank my wife Ge Lin, for her continuous help, support and love during the course of my studies.

Table of Contents

Abstract	iii
Acknowledgments	iv
Table of Contents	v
List of Figures	ix
List of Tables	xi
List of Acronyms	xii
 Chapter 1: Introduction and Organization	 1
1.1 Introduction	1
1.2 Objectives	2
1.3 Organization	3
 Chapter 2: A Brief Overview of the BNR Network and the Pegasus Project	 4
2.1 Introduction	4
2.2 The BNR Network	4
2.2.1 Network Architecture	4
2.2.2 Network Management	7
2.3 The Pegasus Project	8
2.3.1 Objectives	8
2.3.2 System Architecture	8
2.3.3 Network Simulator	10
2.3.4 Various Diagnostic Tools	10
2.4 Summary	11
 Chapter 3: Machine Learning Techniques: Applications in Network Management	 13
3.1 Introduction	13
3.2 Inductive Learning Techniques	14
3.2.1 Overview	14
3.2.2 Tree Induction Algorithms	15
3.2.3 Rule Induction Algorithms	17
3.2.4 The CN2 Rule Induction Algorithm	19
3.3 Machine Learning in Network Management	21
3.3.1 Overview	21
3.3.2 ITRULE and Its Application in Network Management	22
3.3.3 MERLINO and Its Application in Network Fault Diagnosis	23

3.4	Summary	24
Chapter 4:	The Overfitting Problem and Pruning Techniques	26
4.1	Introduction	26
4.2	The Overfitting Problem	26
4.2.1	Definition of the Overfitting Problem	26
4.2.2	Cause of the Overfitting Problem	27
4.3	Pruning Techniques	28
4.3.1	Pruning as a Solution to the Overfitting Problem	28
4.3.2	CN2's Overfitting Avoidance Approach	29
4.3.2.1	The Rule Evaluation Function	29
4.3.2.2	The Significance Test Pruning Method	31
4.4	The Need for a New Pruning Method	32
4.4.1	Limitations of Current Pruning Techniques	32
4.4.2	A New Pruning Method Would Be Beneficial	33
4.5	Summary	34
Chapter 5:	The Development of a New Pruning Method for Inductive Learning Algorithms	35
5.1	Introduction	35
5.2	The Regression Pruning Algorithm	35
5.3	General Experimental Methods	38
5.3.1	Implementation of RPA and the CN2 Test-bed	38
5.3.2	Types of Noise Experimented in This Thesis	39
5.3.3	Criteria for Evaluations of Pruning Results	40
5.4	Experiments with An Artificial Domain	40
5.4.1	Objectives	40
5.4.2	The Artificial Domain and Data Sets	41
5.4.3	Experimental Results and Analysis	43
5.4.3.1	Effects of Class Noise on the Formation of Classification Rules	43
5.4.3.2	Results of Applying CN2 Significance Test	44
5.4.3.3	Results of Applying RPA	47
5.4.4	Summary	49
5.5	Experiments with the LED Domain	49
5.5.1	Objectives	49
5.5.2	The LED Domain	50
5.5.3	Experimental Methods	51
5.5.4	Experimental Results and Analysis	52
5.5.4.1	Experiments under Normal Attribute Noise Rate	52

5.5.4.1.1	Data Set Size, Noise/Signal Ratio, and the Overfitting Problem	53
5.5.4.1.2	Pruning Results in Data Group A	54
5.5.4.1.3	Pruning Results in Data Group B	58
5.5.4.1.4	Pruning Results in Data Group C	59
5.5.4.2	Experiments under Higher Attribute Noise Rate	60
5.5.5	Summary	62
5.6	Experiments with the Soybean Domain	63
5.6.1	Objectives	63
5.6.2	The Soybean Domain and Data Sets	63
5.6.3	Experimental Methods	64
5.6.4	Experimental Results and Analysis	65
5.6.4.1	Experiments on Split Data Sets	65
5.6.4.2	Experiments on Original Data Sets	66
5.6.5	Summary	69
5.7	Limitations and Improvements	70
5.8	Concluding Remarks	70
Chapter 6: The Application of CN2 Rule Induction Program in Network Fault Diagnosis		73
6.1	Introduction	73
6.2	A General System Model	74
6.3	The Development of Router Normal Traffic Patterns	77
6.3.1	Experimental Conditions	77
6.3.2	Experimental Results and Analysis	78
6.3.2.1	The First Experiment	79
6.3.2.2	The Second Experiment	90
6.3.2.3	The Third Experiment	93
6.3.2.4	The Fourth Experiment	95
6.4	Summary	99
Chapter 7: Conclusions, Contributions and Future Research		101
7.1	Conclusions	101
7.2	Contributions	102
7.3	Future Research	102
References		104
Appendices		109
I --	An example data set created from the artificial domain	109

II -- The unordered rule lists generated from Set_1 under different CN2 significance test levels.	111
--	-----

List of Figures

Chapter 2

- Figure 2.1 Basic Components and Connections in the BNR WAN. 6
 Figure 2.2 The Pegasus System Architecture. 9

Chapter 5

- Figure 5.1 The Regression Pruning Algorithm 36
 Figure 5.2 An example data set from the artificial domain 41
 Figure 5.3 The unordered rule list generated from the example data set. 42
 Figure 5.4 Data Set_1 which contains a Class noise (#36) 43
 Figure 5.5 Data Set_2 which contains a fresh example (#36) 44
 Figure 5.6 The unordered rule list generated from Set_1 with Threshold 0 . . . 45
 Figure 5.7 The unordered rule list generated from Set_1 with Threshold 5 . . . 46
 Figure 5.8 The unordered rule list generated from Set_1 with RPA 48
 Figure 5.9 Results of applying RPA in Data Group A. 56
 Figure 5.10 Results of applying RPA in Data Group B. 58
 Figure 5.11 Results of applying RPA in Data Group C. 59
 Figure 5.12 Results of applying RPA in Data Group D. 61
 Figure 5.13 Performance of RPA on Split Soybean Data Sets 67
 Figure 5.14 Performance of RPA on Original Data Sets 68
 Figure 5.15 The Modified Regression Pruning Algorithm 71

Chapter 6

- Figure 6.1 The General System Model for Applying Machine Learning Techniques
 in Network Fault Diagnosis 75
 Figure 6.2 The System Model for Applying CN2 in the Development of Router
 Normal Traffic Patterns 76
 Figure 6.3 Router bCARc2 and its Connections 80
 Figure 6.4 A Typical Rule for Router bCARc2 79
 Figure 6.5 Line #1 Traffic Variations Over 12 Hours 83
 Figure 6.6 Line #10 Traffic Variations Over One Day 84
 Figure 6.7 Line #12 Traffic Variations Over One Day 85
 Figure 6.8 Rule Accuracy for Different Classes: Evaluation on the Training Set
 88
 Figure 6.9 Rule Accuracy for Different Classes: Evaluation on the Testing Set
 89
 Figure 6.10 Line #1 Traffic Variations Using "10-minute-mean" Traffic Collection
 Function 93
 Figure 6.11 Line #10 Traffic Variations Using "10-minute-mean" Traffic Collection
 Function 94
 Figure 6.12 Line #12 Traffic Variations Using "10-minute-mean" Traffic Collection
 Function 95
 Figure 6.13 Applying RPA on "10-minute-mean" Traffic Data: Training Accuracy
 96
 Figure 6.14 Applying RPA on "10-minute-mean" Traffic Data: Testing Accuracy

Figure 6.15		97
Figure 6.16	Applying RPA in the Fourth Experiment: Training Accuracy.....	98
	Applying RPA in the Fourth Experiment: Testing Accuracy.....	98

List of Tables

Chapter 5

Table 5.1	CN2 Default Parameters	39
Table 5.2	Results of applying CN2 significance test	45
Table 5.3	Results of applying RPA.	47
Table 5.4	Performance of Previous Learning Systems in the LED Domain . .	50
Table 5.5	CN2's performance: 10% normal attribute noise rate; No pruning .	53
Table 5.6	Noise/Signal ratio in Data Group A under 10% normal attribute noise rate.	54
Table 5.7	CN2's performance in Data Group A with the significance test . . .	54
Table 5.8	Results of Applying kPA in Data Group A: Prediction Accuracy. .	55
Table 5.9	Results of applying RPA in Group A: Rule Simplification	57
Table 5.10	Effect of the overfitting problem under different attribute noise rates	61
Table 5.11	The Distribution of Original Soybean Training Examples among All Classes.	65
Table 5.12	CN2's Performance on Split Soybean Data Sets	66
Table 5.13	Performance of CN2 Significance Test on Original Soybean Data Sets	67
Table 5.14	Comparison of Performance on Original Soybean Data Sets.	68

Chapter 6

Table 6.1	Results of Applying the Significance Test	81
Table 6.2	Results of Using Different Star Size.	81
Table 6.3	Rule Evaluation on the Training Set	85
Table 6.4	Rule Evaluation on the Testing Set.	86
Table 6.5	Results of Applying the "15-minute-total" Traffic Collection Function	91
Table 6.6	Results of Applying the "10-minute-mean" Traffic Collection Function	91
Table 6.7	Comparison of Rule Accuracy under Different Traffic Collection Functions.	92
Table 6.8	Comparison of CN2's Performance on the "10-minute-mean" Traffic Data.	94
Table 6.9	Comparison of Results in The Third and Fourth Experiment without RPA.	96
Table 6.10	Effect of Applying RPA in the Third and Fourth Experiment	99

List of Acronyms

ATM	Asynchronous Transfer Mode
BNR	Bell Northern Research
DCD	Data Collector Device
EBL	Explanation Based Learning
KS	Knowledge Source
LAN	Local Area Network
MAN	Metropolitan Area Network
NOC	Network Operations Center
NT	Northern Telecom
PING	Packet Internet Grouper
RPA	Regression Pruning Algorithm
SBL	Similar Based Learning
SNMP	Simple Network Management Protocol
WAN	Wide Area Network

Chapter 1

Introduction and Organization

1.1 Introduction

The primary motivation for this research is the application of machine learning techniques to the development of rule-based expert systems for diagnosing faults in the communications network. Our goal is to apply the inductive learning technique to facilitate the process of constructing a rule base to be used in network fault diagnosis.

Machine learning is a study of computational methods for acquiring knowledge and improving problem solving ability. Among multiple machine learning techniques, the inductive learning technique has been widely studied and applied in real world applications. An inductive learning program is presented with examples of different concepts (classes), known as the training set. For each concept, the program constructs a description (rule) that covers its examples and excludes others as much as possible. However, many inductive learning algorithms produce rules that are overspecialized in the sense that the rules get better accuracy on the training data than on the new testing data. This phenomenon has been termed as *overfitting* of training data by learning algorithms [Feng92].

This thesis addresses the overfitting problem and develops a new pruning method -- the Regression Pruning Algorithm (RPA) to reduce the overfitting problem. Using CN2 rule induction algorithm (which is chosen by this thesis as a representative of inductive learning algorithms) as a test-bed, our experimental results on various task domains, including the network application

domain, verify that the Regression Pruning Algorithm does indeed reduce the overfitting problem significantly, compared with other pruning methods.

1.2 Objectives

The objectives of this thesis are as follows:

1. Conduct research on machine learning techniques and their applications in network management. Literature is reviewed to study various inductive learning techniques. Specifically, three inductive learning algorithms -- ID3 ([Quinlan86a]), AQ ([Michalski80]) and CN2 ([Clark89], [Clark91]) are reviewed and compared in this thesis. Some examples of machine learning applications in network management are also studied.
2. Investigate and identify some of the causes for the overfitting problem which is encountered by most of inductive learning algorithms. Using CN2 rule induction algorithm as a test-bed, develop a new pruning method -- the Regression Pruning Algorithm to reduce the overfitting problem. Test and verify this new pruning method in various task domains.
3. Using a simulation of the Bell Northern Research (BNR) wide area network as a test-bed, investigate the application of machine learning techniques in network fault diagnosis. A general system model for applying machine learning techniques in network fault diagnosis is developed. Specifically, CN2 rule induction program is used to analyze the network router traffic to establish a set of normal traffic patterns, one for each time interval of a day, to account for any time of day traffic variations. These normal traffic patterns can be used to build a rule base to be used in network fault diagnosis. The Regression Pruning Algorithm is also empirically verified in this application domain.

1.3 Organization

The remainder of this thesis is divided into six chapters. Chapter 2 briefly overviews the BNR wide area network and the Pegasus fault diagnosis project ([Pagurek91]) to provide the background information for this thesis. Chapter 3 reviews various machine learning techniques and their applications in network management and fault diagnosis. Chapter 4 investigates and identifies some of the causes for the overfitting problem and reviews state of the art pruning methods for reducing the overfitting problem. Chapter 5 presents the research and development of a new pruning method -- the Regression Pruning Algorithm to reduce the overfitting problem and compares the experimental results on three task domains with those of other pruning methods. Chapter 6 completes another objective of this thesis by applying the CN2 rule induction program in router traffic pattern analysis for network fault diagnosis and further tests the Regression Pruning Algorithm in the network domain. In Chapter 7, this thesis is concluded, contributions and recommendations for future research are discussed

Chapter 2

A Brief Overview of the BNR Network and the Pegasus Project

2.1 Introduction

The Pegasus project, sponsored by the Telecommunications Research Institute of Ontario (TRIO) and Bell Northern Research (BNR), began in 1990 to conduct research on network fault diagnosis. The BNR Wide Area Network (WAN) is being used as a test-bed for the development of new diagnostic methods under the Pegasus project. As one of the multiple approaches deployed in this project, this thesis applies machine learning techniques to alleviate the knowledge acquisition bottleneck in the development of rule-based expert systems to be used in network fault diagnosis. This chapter overviews the attributes of the BNR network and reviews the research work which has been conducted in the Pegasus project.

2.2 The BNR Network

2.2.1 Network Architecture

The BNR WAN is a large, private data communications network ([BNR90a]). It spans four continents - North America, Europe, Asia and Australia and supports over 2000 Macintosh computers, over 1500 engineering workstations in the Ottawa area alone, and a few hun-

dred of other miscellaneous machines such as PCs and DEC VAXs.

The BNR WAN environment is made up of a hierarchy of relatively small Local Area Networks (LANs) interconnected into a Metropolitan Area Network (MAN) in the Ottawa area and further interconnected into a WAN to other BNR/NT locations.

Figure 2.1 shows the basic components and connections in the BNR WAN. All of the workstations in this network are connected using Ethernet protocols. Up to 11 workstations may be connected to each primary hubs and up to 16 primary hubs may be connected to each secondary hubs, which may also provide connectivity to file servers used by all of the workstations under the secondary hubs. The workstations, primary hubs, secondary hubs and file servers form a workgroup LAN in BNR, which may reduce the traffic load of the whole network by consolidating local traffic which need not go to the LAN backbone.

Workgroup LANs are interconnected by each LAN backbones through bridges. Bridges allow workstations below it to communicate with both remote workstations and other workstations below it. The LAN backbones also provide connectivity for some of the special devices such as IBM Mainframes and Fastpaths for Appletalk devices.

LAN backbones are further interconnected through routers to form MANs and WANs. Typically there are two routers connected to each LAN backbones for redundancy in case one of the routers fails. Each router is typically connected to two LAN backbones and at least two serial lines which provide interconnections between routers with transmission speeds ranging from 56 Kb/s to 1.544 Mb/s.

Routers are important devices in the network. They perform routing by picking up packets from LAN backbones destined for other LAN backbones and forward them to the appropriate routers. They also accept packets from other routers and either forward them to next routers along their paths or put them on the destination LAN backbones. An internal routing table is maintained by each of the routers to dynamically choose the best route to for-

To routers at other sites

Figure 2.1 Basic Components and Connections in the BNR WAN

ward the packets. While the dynamic routing approach provides the load balancing, better performance, and error recovery, it does add the complexity of diagnosing network faults because the robustness in the presence of faults tends to mask the existence of the faults. Router traffic pattern analysis is the current approach used in the Pegasus fault diagnosis project to provide secondary symptoms by which hypotheses about failure modes and causes may be indicated.

2.2.2 Network Management

Communications networks are vulnerable because they may fail for a variety of reasons - incorrect working of the technology, accidental or deliberate errors when planning, designing, configuring and using such networks. Thus networks have to be managed. Network management has evolved as a distinct and important aspect of the network architecture.

The typical faults in the BNR network have been summarized into four main categories - LAN Physical Layer Faults, Communication Facility Faults, "Finger Problems" and Protocol Issues ([BNR92]). These faults may be caused by hardware failures, errors in software, or human operator errors.

In BNR, the network management functions are performed at the Network Operations Center (NOC). The NOC performs the network monitoring by using a network management system called MAGNet ([BNR90b]). The MAGNet probes the entire network using two network management protocols - the Simple Network Management Protocol (SNMP) and the Packet Internet Grouper (PING). The workstations which are installed SNMP or PING to perform the network monitoring operations are called Data Collector Devices (DCDs). There are totally eight DCDs in the network, all located in the Ottawa area.

Currently, most of the network management functions are manually undertaken by the human network operators. They look up the status messages generated from the DCDs to make the management decisions. More experienced operators may also use the collected

router traffic loads as a further judgement. But for the most part, these traffic rates are not analyzed by the operators and are simply stored in log files for future reference.

2.3 The Pegasus Project

2.3.1 Objectives

The Pegasus project uses a multiple paradigm approach to network fault diagnosis with several cooperating intelligent diagnostic agents organized along the blackboard architecture ([Pagurek91]). The objectives of the Pegasus project are as follows:

Firstly, automatize the process of fault diagnosis. As the network size becomes larger and the network topology becomes more complex, it is very difficult for a network operator to make his decisions promptly in front of the large amount of network status messages. A fully automated diagnostic system is thus ultimately desired to help the network operators.

Secondly, apply multiple reasoning methods instead of a single method to provide the automated diagnostic system with robustness and comprehensibility. The future diagnostic system should not only response to the faults passively but also should be able to actively or even proactively detect the potential faults. The hypotheses of one diagnostic method should be double checked by other methods as well.

Thirdly, utilize more network information such as router traffic rates or error rates in addition to the status messages to get a broader view of multiple network faults.

2.3.2 System Architecture

Figure 2.2 shows the general architecture of the Pegasus system. This system was developed under UNIX environment running on a SUN SPARC-2 workstation. The languages

Figure 2.2 The Pegasus System Architecture

mainly used in its development are C, C++, and PASCAL. The entire system is based on a blackboard architecture through which various processes communicate. The major components include the Blackboard - a central resource controller ([Iqneibi92]), the PegasusBase - a central database ([Yuan92]), the Dragon simulator - a full scale BNR network simulator ([Dawes92]), and several Knowledge Sources (KSs) which are diagnostic tools developed using different technologies.

The KSs and the simulator can communicate through the Blackboard which controls the access to the different areas in the shared memory using semaphores. All of the information passed through the Blackboard is dynamically stored into the PegasusBase for future reference.

The main advantage of using the blackboard architecture is that it provides the system with more flexibility. Under this architecture, different KSs can be developed and tested independently. Once operational, they can cooperate through the Blackboard or can be activated only when needed.

2.3.3 Network Simulator

One of the key components in the Pegasus system is the Dragon ([Dawes92]) -- a BNR WAN simulator. It was developed at the beginning of the Pegasus project to allow the development and evaluation of various diagnostic methods by inserting faults into the simulated network, which is impossible under the real network environment.

This simulator supports most of the externally observable communications functionality of very large WAN and ATM / Broadband-ISDN networks. It can be integrated into the Pegasus Blackboard system but also can be run stand-alone. It has extensive fault and traffic generation facilities along with many data analysis and control interfaces. It can run at near real time or up to 50 times faster than real time depending on the needs of applications. The simulated network traffic currently can vary with the time of a day. The different types of faults that can be introduced into the simulator are: element break, line break, packet loss rate, router corruption, router line delay, large file transfer, and data storm.

2.3.4 Various Diagnostic Tools

Based on the Blackboard system architecture and the full scale network simulator, various diagnostic tools or Knowledge Sources (KSs) have been developed independently in the Pegasus project. These KSs include:

1. **BANES - Basic Analysis in Nested Evidence Spaces ([Dawes91])**, which uses the Probabilistic Reasoning method to diagnose the network broken faults. It is reactive and acts on the status messages.
2. **END - Expert Network Diagnostician ([Dussault92])**, which uses the Rule-Based Expert System method to verify the BANES output, and to diagnose the network broken faults. It is active and acts on the knowledge from the network operators.

3. **LEAPR - Line Element Anomaly Pattern Recognition ([Viens92])**, which uses the Neural Network technology for router traffic pattern classifications. It is proactive and acts on the traffic statistics.
4. **Detection of Abnormal Router Behavior ([Farrell93])**, which uses the Statistical methodology to detect the router non-broken faults. It can be reactive or proactive and acts on the traffic statistics.
5. **NPAT_{1.1} and MEDIAN - Operational Research Tools ([Kalab93])**, which uses the Operations Research methods for the optimal network design and planning. It is proactive and acts on the network topology.
6. **Case Based Reasoning for Communication Network Fault Diagnosis ([Zary93])**, which uses the Case Based Reasoning techniques to detect the router non-broken faults. It is reactive and acts on the status messages and router traffic statistics.

2.4 Summary

The BNR WAN is very large and complex. The router plays an important role in the network. Current BNR network management functions are undertaken manually and use a single method approach. Only the status messages are used for diagnosis at most of the time. Thus it is highly desired to develop a fully automated, multiple paradigm approach diagnostic system. This is the ultimate goal of the Pegasus project.

The full scale network simulator and the blackboard system architecture developed in the Pegasus project allow the different diagnostic tools to be developed independently and to cooperate each other once integrated along the Blackboard. Various methodologies have been used in the Pegasus project to diagnose broken or non-broken network faults. All have proved very success-

ful. The router traffic pattern analysis is one of the widely used techniques in this project. This thesis applies the machine learning techniques to alleviate the knowledge acquisition bottleneck in the development of router normal traffic patterns, which has not yet been attempted in the Pegasus project.

Chapter 3

Machine Learning Techniques: Applications in Network Management

3.1 Introduction

The task of fault diagnosis, isolation, and repair in such a large data network as the BNR WAN is the responsibility of the network management operation, and is very demanding in terms of human expertise. This expertise however is limited by both lack of skilled personnel, and by the rapid pace of technological development ([Goodman89]).

Rule-based expert systems have been used in other fields to perform fault diagnosis, simulation, testing and monitoring, intelligent user interfaces, planning tools, and other decision intensive tasks in order to relieve the shortage of skilled staff. The technology is now proving an appropriate technology to tackle problems in network operations and management and the use of rule-based expert systems in network management has recently mushroomed ([Goodman89]). However, the major drawback of the rule-based expert system technology -- the well known knowledge acquisition bottleneck in obtaining knowledge or rules directly from human experts ([Goodman92]) limits its applications.

Machine learning techniques have been developed recently to ease the knowledge acquisition bottleneck by automatically generating and refining rules from large databases in the development of rule-based expert systems. This chapter provides an overview on some of the typical

machine learning algorithms including the CN2 rule induction algorithm which is used as a test-bed in this thesis for the development of a new pruning method to reduce the overfitting problem and for the application of machine learning techniques in network fault diagnosis. The most recent applications of machine learning techniques in network management and fault diagnosis are also reviewed in this chapter, which provides the background information for employing the machine learning approach in the Pegasus project.

3.2 Inductive Learning Techniques

3.2.1 Overview

Machine learning is a study of computational methods for acquiring knowledge and improving problem solving ability. Because of the breadth of this chapter, machine learning includes a wide range of methods such as Learning concepts from examples, Genetic algorithms, Case-based reasoning, Connectionist techniques, Language acquisition, and Conceptual clustering, just named a few. These methods have been applied to a variety of task domains including: Classification and recognition, Problem solving and planning, Reasoning and inference, Natural language processing, Design and diagnosis, Vision and speech perception, Robotics and motor control, etc.

Among the multiple methods, the task of Learning concepts from examples is one of the widely studied problems in the machine learning area ([Carbonell87]). The basic definition for concept learning is: Given examples and counterexamples of some concepts (classes), generate an intensional description of that concept. This description should cover all the examples but none of the counterexamples, and it should correctly classify future instances ([Carbonell87]). Although the definition is straightforward, there exist multiple approaches in the concept learning area.

Inductive learning method is a major branch in the concept learning area ([Carbonell87]). Inductive learning algorithms carry out search for concepts from the examples with little or no other domain knowledge. Each example is described by a fixed set of attributes and a class to which the example belongs. The algorithm is presented with an example set (known as the training set, with each example represented as a list of attribute-value pairs) from which it learns a set of rules or descriptions for classifying the examples in the data set from their attribute values. The algorithm should then be able to apply the rules or descriptions to a new set of data, i.e. the testing set, and classify each example correctly.

Inductive learning methods move from specific data to some general rules or descriptions, and in this sense they are clearly doing induction, whether they search the description space in a general-to-specific or a specific-to-general direction. There are two major families of systems in the inductive learning area. They are Tree induction algorithms and Rule induction algorithms.

3.2.2 Tree Induction Algorithms

Tree induction algorithms represent acquired concepts in the form of decision trees. In addition, these algorithms construct their trees in a top-down fashion. A representative of tree induction systems is the ID3 algorithm ([Quinlan86a]).

The ID3 tree induction algorithm constructs a decision tree from a set of training examples. The input to ID3 is a list of positive and negative examples of some concepts or classes. The output is a decision tree with tests at each node for sorting examples down alternative branches. Terminal nodes contain the class of objects that have been sorted by all earlier decisions in the tree, and which are not further discriminable.

ID3 induces a decision tree by repeatedly specializing leaf nodes of an initially single-node tree. This is a general-to-specific approach. The specialization operation involves replac-

ing a leaf node with an attribute test, and adding new leaves to that node corresponding to the possible results of that test. An information theoretic measure, e.g. entropy is used to select the attribute whose values improve prediction of class membership above the accuracy expected from a random guess. The training set is recursively decomposed in this manner until no remaining attribute improves prediction in a statistically significant manner by a user-supplied parameter of "confidence" (e.g., 90%).

To classify a new example, a path from the root of the decision tree to a leaf node is traced. At each internal node reached, one follows the branch corresponding to the value of the attribute tested at that node. The class at the leaf node represents the class prediction for that example.

One of the major advantages of tree induction algorithms is that the Top-Down Induction of Decision Tree (TDIDT) schema can be easily adapted to handle noisy data. In real-world classification tasks, the description of an object often contains errors. Some sources of these errors are faulty measurement, ill-defined thresholds (e.g., when is a person "tall"?), and subjective interpretation of a multitude of inputs (e.g., what criteria are used when describing a person as "athletic"?). The examples which contain error descriptions have been termed as noisy data ([Quinlan86b]). Since the classification rule is couched in terms of the description of an object, the noisy data can be expected to affect the formation and use of classification rules incorrectly.

Tree induction algorithms can be easily adapted to handle noisy data by virtue of their TDIDT approach to tree generation. During induction, all possible attribute tests are considered when "growing" a leaf node in the tree, and entropy is used to select the best one to place at that node. Overfitting of the noisy data thus can be avoided by halting tree growth when no more significant information can be gained. Tree pruning techniques (e.g., [Quinlan87a]; [Niblett87]), used for example in ID3's descendants C4 ([Quinlan87a]) and ASSITANT ([Cestnik87]), have proved effective against noisy data.

However the TDIDT approach does suffer from a different set of limitations. For instance, the nonincremental nature of TDIDT systems makes them quite inefficient at incorporating new instances since they must recompute their trees from scratch when new data are encountered. Another major problem is that decision trees are essentially sequential decision algorithms which are quite different to the data-driven nature of expert systems. Rule bases are data-driven in the sense that any set of input data can potentially be used to begin the inference. Trees however must always begin with the attribute associated with the root node. In addition, rule bases can accommodate missing attribute information, whereas trees are not designed to do so. Trees can also be difficult to understand for the user, a problem which should not be underestimated in light of the overall advantages of explicit knowledge representation inherent to production rules which are generated from rule induction algorithms. So, Quinlan described a scheme ([Quinlan87b]) whereby ID3-induced trees are transformed into production rules.

3.2.3 Rule Induction Algorithms

Unlike tree induction algorithms, the rule induction algorithms generate a concept description stated as a disjunction of conjunctions which is equivalent to a set of "if... then..." classification rules rather than a decision tree. A well known rule induction system is the AQ algorithm ([Michalski80]).

Given a set of training examples of different concepts or classes, the AQ algorithm generates a set of decision rules for each class in turn. It starts by selecting some seed object from the positive instances and finding an optimal rule that covers this instance but none of the negative instances. The algorithm then removes all positive instances that are covered by the rule, selects a new seed from the remaining set, and generates another rule based on the new seed. This process continues until all positive instances have been covered by at least one of the generated rules. This whole process is repeated for each concept or class.

The original AQ algorithm carries out a beam search through the space of descriptions. Like tree induction method, it uses an evaluation function to constrain its search. The default preference criterion favors conjunctive descriptions that cover the maximum number of positive instances. Therefore, if the algorithm cannot find a single conjunctive concept description, it generates a disjunctive description with a small number of conjunctions. The AQ method carries out considerably more search than ID3 and its relatives, but it can handle production rules.

In the AQ algorithm, a new example is classified by finding which of the induced rules have their conditions satisfied by the example. If the example satisfies only one rule, then one assigns the class predicted by that rule to the example. If the example satisfies more than one rule, then one predicts the most common class of training examples that were covered by those rules. If the example is not covered by any rule, then it is assigned by default to the class that occurred most frequently in the training examples.

The major advantage of rule induction algorithms over tree induction algorithms is that rule induction algorithms generate classification rules which provide a much more flexible concept representation than the tree structures: Firstly, rules are much easier to understand for the user than the decision tree especially when the tree is large; Secondly, rule bases are data-driven in the sense that any set of input data can potentially be used to begin the inference while trees must always begin with the attribute associated with the root node. The flexibility makes rule induction algorithms more easy to be applied to the rule-based expert system domain than tree induction algorithms.

However, rule induction algorithms do have some limitations. One of the major problems is that the dependence on specific training examples during search makes them less easy to handle noise. The reason is that the AQ algorithm involves an implicit assumption that the Bayes error rate for the problem is zero, i.e., "perfect" classification of each attribute is possible in terms of the other attributes, which is usually not true to the real-world domain that

might contain noisy data ([Goodman92]). In addition, the rule induction algorithms carry out considerably more search than tree induction algorithms, which is computationally unattractive.

3.2.4 The CN2 Rule Induction Algorithm

More recently, Clark et. al. ([Clark89], [Clark91]) have developed CN2, a new rule induction algorithm which combines the efficiency and ability to cope with noisy data of ID3 with the if-then rule form and flexible search strategy of the AQ algorithm.

In designing the CN2 algorithm, Clark et. al. modified the AQ algorithm in ways that removed its dependence on specific training examples, increased the space of descriptions searched, and applied statistical techniques which are analogous to those used for tree pruning in the generation of if-then rules. As a result, CN2 can be viewed as a generalization of the AQ algorithm.

The AQ algorithm, when generating a complex (a conjunction of attribute tests), performs a general-to-specific beam search for the best complex. But the method only considers specializations that exclude some particular covered negative example from the complex while ensuring some particular seed positive example remains covered, iterating until all negative examples are excluded. As a result, AQ searches only the space of complexes that are completely consistent with the training data.

The CN2 algorithm retained the beam search method of the AQ algorithm but removed its dependence on specific examples during search and extended its search space to include rules that do not perform perfectly on the training data. This is achieved by broadening the specialization process to examine all specializations of a complex, in much the same way that ID3 considers all attribute tests when growing a node in the tree. This top-down search for complexes lets CN2 apply a cutoff method similar to decision-tree pruning to halt

specialization when no further specializations are statistically significant.

During the search procedure, CN2 employs a rule evaluation function (Entropy or Laplace error estimate) and a pruning method (the Significance test) to determine whether a rule found is both good (has high accuracy when predicting the majority class covered) and reliable (the high accuracy on training data is statistically significant). The evaluation functions and the pruning method are discussed in more detail in Chapter 4.

CN2 produces two types of rules: an Ordered rule list or an Unordered rule set. The original CN2 algorithm ([Clark89]) generates an ordered rule list only. The last rule in the list is a default rule, which simply predicts the most commonly occurring class in the training data for all new examples. To use the induced rules to classify new examples, CN2 tries each rule in order until one is found whose conditions are satisfied by the example being classified. The class prediction of this rule is then assigned as the class of the example. If no induced rules are satisfied, the final default rule assigns the most common class to the new example. To understand a rule, all the previous rules in the list must also be taken into consideration since the meaning of any single rule is dependent on all the other rules which precede it in the rule list. The ordering of the rules is then important. From this point of view, the ordered rule list is a form of decision tree ([Goodman92]). As a result, the ordered rule list representation suffers from the same limitations mentioned earlier with respect to tree induction algorithms.

In [Clark91], the original CN2 algorithm was extended to generate unordered rule sets like AQ-based systems by modifying only its control procedure, leaving the beam search procedure unchanged. The main modification to the algorithm is to iterate the search for each class in turn, removing only covered examples of that class when a rule has been found. Unlike for ordered rules, the negative examples remain because now each rule must independently stand against all negatives. The covered positives must be removed to stop CN2 repeatedly finding the same rule. With an unordered rule list, all rules are tried and those whose conditions are satisfied by the example being classified are collected. If a clash occurs (i.e.

more than one class predicted), some probabilistic method is needed to resolve clashes. The method used by [Clark91] is to tag each rule with the distribution of covered examples among classes, and then to sum these distributions to find the most probable class should a clash occur.

The same experiments as performed for the earlier experiments on ordered rules in [Clark89] were carried out in [Clark91] again and the results show that the CN2 unordered algorithm has an even higher accuracy than that of CN2 ordered algorithm, with a small (2%) but significant (at the 95% level) higher average accuracy. Clark et. al. claimed that one possible explanation for this high performance is that, with unordered rules, several rules may contribute to the classification of one example thus reducing effects of noise and occasional poorly performing rules. Spreading of the classification decision over several rules has been termed “using multiple knowledge” in [Gams91]. Clark et. al. claimed that it seems likely a similar phenomenon is occurring in CN2 unordered algorithm.

3.3 Machine Learning in Network Management

3.3.1 Overview

Network management is a relatively new area but the number of methodologies in this areas has grown very fast. Rule-based expert systems are among the many successful approaches to network management ([Patel89]). However, rule-based expert systems suffer from the well known knowledge acquisition bottleneck problem when obtaining rules directly from human experts ([Goodman92]). As a result, there is a demand for tools which can solve the knowledge transformation problem.

Machine learning techniques have been recently used in the network management area

and have proved efficient in easing the bottleneck of knowledge acquisition in the development of rule-based expert systems for network management ([Patel89]). Here we show two examples of applications of machine learning techniques in network management and fault diagnosis.

3.3.2 ITRULE and Its Application in Network Management

ITRULE ([Goodman91], [Goodman92]) is a generalized rule induction algorithm developed by Goodman and Smyth to automatically generate useful rules from trouble ticket and alarms databases. The goodness of a rule is measured by a hypothesis preference criterion which is called J-measure to trade-off simplicity and goodness-of-fit. The rules produced by ITRULE can be used either as a human aid to understand the inherent model embodied in data, or as a tentative input set of rules to an expert system. In this case, ITRULE can ease the knowledge acquisition bottleneck by presenting the expert with a tentative rule set, or, in cases where no human expert exists, it may directly transform data into rule-based systems.

The ITRULE algorithm was implemented on a network trouble ticket database ([Goodman91]) and was found to be a useful data summarizing tool, recording how specific problems should be closed out and how long it should take to do so. It was also implemented on an alarm database, outputting a real-time prioritized list of the most important network alarms.

Goodman et. al. have made valuable contributions to the area of network management. Their work shows that machine learning techniques can be used to ease the bottleneck of knowledge acquisition by making information available in the convenient format of rules and machine learning tools are useful for automated rule induction in large network management databases.

3.3.3 MERLINO and Its Application in Network Fault Diagnosis

More recently, Bisio et. al. ([Bisio92]) have described their research of using a machine learning tool -- MERLINO ([Bisio91]) in the diagnosis of network overload.

MERLINO is a learning system for incremental, semi-automated knowledge base construction. MERLINO philosophy foresees a man-machine interaction conducted by the human expert, who writes a tentative knowledge base and gradually refines it by means of an automated learning module. The tentative knowledge base is logically divided into a set of fully specified rules and a set of partially defined rules, expressed using an extension of first order predicate logic suitable to declare partial definitions. MERLINO learning module is an integrated inductive / deductive system, devoted to solving the partial definitions present in the knowledge base, provided there is suitable set of examples and counterexamples. The solutions proposed, if validated by the human expert, are inserted into a new knowledge base version, until a good level of performance is reached. Other MERLINO tools, such as the rule editor, the knowledge base maintenance module and the knowledge base testing module facilitate the gradual construction of the knowledge base.

The studied network in this paper ([Bisio92]) is a 20 node - hierarchical 2 levels - circuit-switched network. The behavior of the described network is simulated by a real-time simulator. The performance parameters are computed on the basis of measures collected at intervals of 15 minutes of simulated time. The performance parameters are stored in a relational data base. The problem under their diagnosis is network overload. MERLINO is used to find a set of rules from the collected network parameters that detect a focussed overload situation and identify the focus. The inductive capability of MERLINO helps to determine, which are the best parameters to use, the simplest correlation of them, the effective thresholds. The deductive capability makes the system very flexible, allowing to exploit the power of a rule based formalism and to take into account the rules proposed by the traffic engineer.

This paper has made some valuable contributions to network fault diagnosis. Firstly, it shows that machine learning systems can be helpful tools in network fault detection and it provides us with a system model for applying machine learning techniques in network fault diagnosis. Secondly, it combines inductive and deductive learning methods for generating new rules and refining pre-existing ones.

3.4 Summary

In this chapter, we reviewed machine learning techniques and their applications in network management. [Goodman91] and [Bisio92] are two examples of applying machine learning techniques in network management and fault diagnosis. These two papers have made valuable contributions to both machine learning and network management areas. They have shown that machine learning techniques can help ease the bottleneck of knowledge acquisition in the development of rule-based expert systems to be used in network management and fault diagnosis. However, none of these two papers have addressed the overfitting problem ([Feng92]), which is general to most of inductive learning algorithms, in their applications. This motivates us to develop a new pruning method in this thesis to ease the overfitting problem while applying inductive learning methods to network fault diagnosis in the Pegasus project.

In the inductive learning area, Tree induction algorithms suffer from their concept description and interpretation structures which are not data-driven and difficult to understand for the user. Rule induction algorithms tend to be favored because of their inherent advantage of explicit knowledge representation. They have been applied to ease the bottleneck of knowledge acquisition when building rule-based expert systems for network management and fault diagnosis. However, the original rule induction algorithms like the AQ families can not handle noisy data efficiently. CN2 is an improved rule induction algorithm that was developed recently to combine the efficiency and ability to cope with noisy data of the tree induction algorithm with the if-then

rule form and flexible search strategy of the rule induction algorithm. As a result, CN2 is chosen as a representative of inductive learning algorithms in this thesis for the application of inductive learning methods in network fault diagnosis.

Chapter 4

The Overfitting Problem and Pruning Techniques

4.1 Introduction

This chapter completes another objective of this thesis by investigating the general overfitting problem encountered by most of inductive learning algorithms especially when they are applied to real-world application domains. This chapter reviews current pruning techniques used to ease the overfitting problem including the CN2 significance test pruning method and identifies their limitations. The need of a new pruning method for CN2 and other inductive learning algorithms is then discussed at the end of this chapter.

4.2 The Overfitting Problem

4.2.1 Definition of the Overfitting Problem

It has been widely noted by researchers studying inductive learning algorithms that performance on training data may be a misleading indication of classification descriptions' predictive accuracy. Most recently, Feng et. al. ([Feng92]) have described their research on comparison of classification algorithms in inductive learning, statistics and neural networks

on data sets from large-scale, real-world applications, including medicine, finance, image analysis and engineering design. The experimental results show that almost all the inductive learning algorithms could generally perform very well on the training set, getting relatively high accuracy, but their accuracy drops considerably on the testing set. This phenomenon then was termed as “overfitting” of training data by the learning algorithms, i.e. classification descriptions derived from the training data reflect not only true underlying relationships, but also patterns arising purely by chance.

4.2.2 Cause of the Overfitting Problem

One of the possible causes of the overfitting problem is the existence of noise in the training data ([Schaffer93]). Quinlan ([Quinlan86b]) has defined noise in his research as the data which contain error descriptions and stated that the training data are likely to contain noise when they are taken from the real-world application domain. As classification descriptions are derived from the descriptions of the training data, the noisy training data are expected to affect the formation and use of the classification descriptions incorrectly.

More specifically, during the concept formation procedure, some of the classification patterns to be found in noisy training data may reflect true underlying structure, but others may arise merely by chances as a consequence of noise. An induction procedure that simply strives to achieve optimum performance on the training data may construct a model that captures both kinds of patterns. It might go too far, fitting noise as well as structure and attempting to extract more information from the data than the data really contains. As a result of the overfitting, the performance of classification descriptions on the new testing data may drop down in comparison to their previous performance on the noisy training data. The level dropped down may vary with the noise rate in the training data.

4.3 Pruning Techniques

4.3.1 Pruning as a Solution to the Overfitting Problem

Since overfitting in this sense decreases prediction accuracy, a great deal of effort has been expended in developing overfitting avoidance methods for inductive learning algorithms, generally in the form of pruning strategies. These methods have been reported widely ([Breiman84], [Cestnik91], [Mingers87], [Niblett86], [Quinlan87]) and they have been compared empirically in [Mingers89]. Few researchers would now undertake decision tree or rule induction without relying on some form of overfitting avoidance ([Schaffer93]).

Pruning techniques have been applied to tree induction algorithms to produce trees of optimal size. Many decision tree algorithms apply pruning: pre-pruning, post-pruning or both. In pre-pruning, the algorithm stops testing when the number and/or mixture of examples belonging to different classes falls below a critical level. In practice, when such a situation occurs, the minority class of examples are treated as noise or are given a probability. In post-pruning, a complete tree is built which classifies all the training data correctly, and then successive nodes are removed from the tree, which changes the ratio of examples classified into different classes at the leaves, until the tree is optimal according to a certain criterion.

Rule induction algorithms also apply pruning techniques to increase the simplicity and prediction accuracy of rules. For example, AQ11 ([Michalski83]) and AQ15 ([Michalski86]) handle noise with pre-pruning and post-pruning techniques, leaving the basic AQ algorithm intact. The CN2 algorithm improves the noise handling ability of the AQ algorithm further by employing a rule evaluation function and a statistical significance test pruning method.

4.3.2 CN2's Overfitting Avoidance Approach

4.3.2.1 The Rule Evaluation Function

The CN2 algorithm carries out a pruned search in the way that it uses a rule evaluation function to justify the predictability of the generated rules, and it also uses a significance test to measure the reliability of the highly predictive rules and terminate the search when no more significant rules can be found.

The original CN2 algorithm ([Clark89]) uses an entropy rule evaluation function which behaves very similarly to apparent accuracy which tends to select very specific rules covering only a few examples, as the likelihood of finding rules with high accuracy on the training data increases as the rules become more specific. In the extreme case, a maximally specific rule will just cover one example and hence have an unbeatable score using the metrics of accuracy (scores 100% accuracy) or entropy (scores 0.00, a perfect score). The problem is that rules covering few examples are unreliable, especially with noise in the domain. Their accuracy on the training data does not adequately reflect their true predictive accuracy on the new test data which may appear.

To avoid selecting highly specific rules, the original CN2 algorithm uses a significance test which ensures that the distribution of examples among classes covered by the rule is significantly different from that which would occur by chance. In this way, many rules covering only a few examples are eliminated, as the significance test deems their apparent high accuracy likely to be simply due to chance.

However, while a significance test eliminates rules which are below a certain threshold of significance, there is still the problem that rules which just pass the significance test might tend to be preferred over more general and reliable but less apparently accurate rules.

Consider a domain with two equally likely classes C1 and C2, and consider three rules

R1, R2, and R3, where:

R1 covers 1000 examples of class C1 and 1 of class C2 (denoted by [1000, 1]);

R2 covers 5 examples of C1 and 0 of C2 (denoted by [5, 0]);

R3 covers 1 example of C1 and 0 of C2 (denoted by [1, 0]).

In this example, the algorithm should ideally prefer R1 as its accuracy on new test data is likely to be the best. Rules R2 and R3 only cover a few examples and their apparent accuracies of 100% are not fully reflective of performance on new test data. However, although a 99% significance test eliminates R3, R2 will just pass and be selected in preference to R1. Raising the significance level further does not solve the problem as a rule R1.5(say) may exist which again just passes the raised significance threshold.

[Clark91] describes the metrics of apparent accuracy/entropy as having undesirable "downward" bias, i.e., preference for rules low down in the general (top) to specific (bottom) search space. Raising the significance threshold causes the level of specificity at which the search terminates to raise, but does not eliminate the downward bias itself.

To solve this downward bias problem, [Clark91] introduced a new rule evaluation function - Laplace expected accuracy estimate. This expected accuracy estimate is given by the following formula:

$$\text{LaplaceAccuracy} = (n_c + 1) / (n_{\text{tot}} + k)$$

where:

k is the number of classes in the domain

n_c is the number of examples in the predicted class covered by the rule

n_{tot} is the total number of examples covered by the rule

Back to the previous example, the Laplace accuracy estimates for predicting the class with the most covered examples in are 99.8% for R1, 85.7% for R2, and 66.6% for R3. Thus,

Laplace rule evaluation function can avoid the undesirable downward bias of the entropy function, and which significance testing can only partly overcome.

4.3.2.2 The Significance Test Pruning Method

As described earlier, the significance test is employed by the original CN2 algorithm, which uses the entropy rule evaluation function, as a pruning method to avoid selecting highly specific rules. The significance test ensures that the distribution of examples among classes covered by the rule is significantly different from that which would occur by chance. In this way, many rules covering only a few examples are eliminated, as the significance test deems their apparent high accuracy likely to be simply due to chance.

To assess significance, CN2 compares the observed distribution among classes of examples satisfying the rule with the expected distribution that would result if the rule selected examples randomly. Some differences in these distributions will result from random variation. The issue is whether the observed differences are too great to be accounted for purely by chance. If so, CN2 assumes that the rule reflects a genuine correlation between attributes and classes. To test significance, the system uses the likelihood ratio statistic ([Kalbfleisch79]). This is given by:

$$2 \sum_{i=1}^n f(i) \log \left(\frac{f(i)}{e(i)} \right)$$

where the distribution $F = (f(1), \dots, f(n))$ is the observed frequency distribution of examples among classes satisfying a given rule and $E = (e(1), \dots, e(n))$ is the expected frequency distribution of the same number of examples under the assumption that the rule selects examples randomly. This is taken as the $N = \sum f(i)$ covered examples distributed among classes with the same probability as that of examples in the entire training set. This statistic provides an information-theoretic measure of the (noncumulative) distance between the two

distributions. Under suitable assumptions, one can show that this statistic is distributed approximately as χ^2 with $n - 1$ degree of freedom. This provides a measure of indicates significance - the lower the score, the more likely that the apparent regularity is due to chance.

But the significance test can only partly overcome the downward bias of the entropy function. The Laplace rule evaluation function can avoid the undesirable downward bias of the entropy function. However, it changes the role of significance testing as a pruning method. The behavior of significance testing with Laplace is qualitatively different to that with Entropy. If entropy is used, raising the significance threshold causes CN2 to select a smaller number of more general rules in preference to a large number of highly specific rules. Using Laplace, the evaluation function is sufficient on its own to bias the search towards those general rules with higher predictive accuracy, tending to find rules of highest predictive accuracy (and thus also high significance) first. The significance testing instead alters the point at which CN2 stops searching for further rules. In other words, with Laplace the significance testing acts solely as a termination criterion for the algorithm. Increasing the significance threshold will cause the algorithm terminate earlier with the number of rules decreasing and the overall accuracy also decreasing.

4.4 The Need for a New Pruning Method

4.4.1 Limitations of Current Pruning Techniques

There are a number of limitations in current pruning techniques including the CN2 significance test pruning method. One of the major problems is that current pruning techniques generally attempt to improve performance by strictly statistical means, using the training data alone in some sophisticated fashion to distinguish between real and spurious structure, which arises by chance as a consequence of noise, in an induced model. However, as stated by

[Schaffer93], the training data itself can hardly tell us which of the patterns in them are real. To distinguish between real and spurious structure in an induced model, we must either have fresh data or independent information indicating which models are inherently more plausible.

Another problem is that some of the pruning methods such as the CN2 significance test are limited to prune classification descriptions only, based on their statistical significance on the training data. A result of this limitation is that the pruning would increase the simplicity of classification descriptions to get reduced number of rules. But it could not change the rule structure in order to get increased prediction accuracy as well. In CN2, for example, with Laplace rule evaluation function, raising the significance threshold only reduces the number of rules. The CN2 would still select the same rules early on during the search but would just terminate earlier. This is observed with the same early rules tending to appear in the rule set but with the number of rules decreasing and the overall accuracy also slightly decreasing.

Another minor problem with some pruning techniques is that the level of pruning can not be automatically controlled. For example, in CN2 the significance threshold must be set up individually for each new data set. One has to manually try a number of times before a better level can be reached and it is usually hard to decide when a threshold level is optimal.

4.4.2 A New Pruning Method Would Be Beneficial

Since current pruning methods have such limitations, a new pruning method would be valuable and beneficial if it could extend current approaches to overcome their limitations. Ideally, such a new pruning method should apply statistical means to prune the training data as well as classification descriptions by using fresh data. Still further, an automated pruning procedure is certainly more preferable. A result of such a pruning method would be the improvement of both the simplicity and prediction accuracy of classification descriptions.

4.5 Summary

Overfitting is a general problem with the applications of inductive learning algorithms to large-scale real-world domains where noise may exist. Pruning techniques have been widely applied to inductive learning algorithms to ease the overfitting problem. However, current pruning methods such as the CN2 significance testing generally suffer from a certain limitations. The task of developing a new pruning method that can overcome the limitations of current pruning approaches would thus be significant and valuable to machine learning and its applications. This is the motive for the development of a new pruning method in this thesis.

Chapter 5

The Development of a New Pruning Method for Inductive Learning Algorithms

5.1 Introduction

In Chapter 4, we discussed the general overfitting problem encountered in most of inductive learning algorithms and reviewed state of the art pruning techniques applied to solve this problem. The limitations of current pruning techniques motivate us to develop a new pruning method in this thesis. In this chapter, we use the CN2 rule induction algorithm as a test-bed to show our development of a new pruning method which can be applied to inductive learning algorithms to improve their performance against noisy data. The new pruning method -- the Regression Pruning Algorithm (RPA) is presented. As tests on a single domain are not sufficient to draw reliable conclusions about the relative performance of the new pruning method, experiments on three different task domains are conducted and the performance of the new pruning method is empirically compared with that of the CN2 significance test and other pruning techniques.

5.2 The Regression Pruning Algorithm

The new pruning method developed in this thesis is called the Regression Pruning Algorithm. The objective in developing this new pruning algorithm is to improve current pruning methods in ways that: Firstly, remove their dependence on the training data by adding in fresh

data for pruning; Secondly, prune the training data with the help of the fresh data to overcome the influence of noise on the formation of classification descriptions so that optimum descriptions can be obtained. Thirdly, automate the pruning procedure. The Regression Pruning Algorithm is shown in Figure 5.1 below.

Step 1: Divide set of examples into a training set and a testing set.

Step 2: Divide the training set equally into a training subset and a pruning subset.

Step 3: Learn CN2 rules from the training subset.

Step 4: Test CN2 rules (from step 3) against the pruning subset, gaining accuracy A_4 .

Step 5:

a) Remove one example $[i]$ from the training subset, relearn CN2 rules from the "leave-one-out" training subset and retest (relearned) CN2 rules against the pruning subset, gaining accuracy A_{5i} .

b) If A_{5i} is higher than any other A_{5j} ($j \neq i$) then remember i and A_{5i} as being the least useful example in the training subset.

c) Repeat Step 5 a) and b) for all examples in the training subset.

Step 6: If $A_{5i} > A_4$, remove example i (the least useful example) from the training subset and return to step 3. Otherwise, the pruning is complete.

Figure 5.1 The Regression Pruning Algorithm

As can be seen from Figure 5.1, the RPA works in an iterative fashion by repetitively pruning the training set to remove noisy (worst) examples with the help of an additional data set - the pruning set. Since the pruning data consists of fresh cases, they can be used to distinguish between real and apparent structures in the original training set as we discussed in Chapter 4.

In each iteration to find a worst case, every example in the current training set is removed once and each time new rules (structures) are reconstructed from the "leave-one-out" new training

set and their prediction accuracy is tested on the pruning set. The new accuracy on the pruning set is compared with the best accuracy (also on the pruning set) which has been found so far during this iteration. If the new accuracy is better, the best accuracy is replaced by the new accuracy and this particular removed example is marked as the worst case before it is put back to the current training set for the next round. On the other hand, if the new accuracy is not better than the current best accuracy, the removed example is simply put back to the current training set to keep the iteration going.

At the end of each iteration, if a worst case has been found, it is removed from the current training set (which used to be the original training set at the beginning of the pruning algorithm) and the pruned "current training set" is passed over to the next iteration as the "current training set" in the new iteration. The whole pruning process is stopped automatically, leaving us the pruned optimum training set, when no more worst cases can be found after a new iteration.

As seen above, the major difference between the RPA and current pruning methods such as the CN2 significance test is that the RPA prunes the training data with the help of the fresh data while the CN2 significance test prunes the existing rules simply according to their statistical significance on the same training data. Since removing noisy examples from the training set allows the inductive learning algorithm to reconstruct different rules from the pruned training set, it is possible for the RPA to get more general rules in the sense that rules are both simple and highly predictive on new testing data. In contrast, CN2 tends to generate the same early rules from the training data but just terminates earlier under higher level significance threshold ([Clark91]). Thus, the CN2 significance test is restricted by the simplification / prediction accuracy tradeoff, i.e., pruning decreases the number of rules but the overall accuracy also decreases ([Clark91]). So, the RPA has an obvious advantage over the CN2 significance test in pruning methodology.

5.3 General Experimental Methods

In the remainder of this chapter, we show our experiments with the RPA on three task domains to test its pruning ability. Firstly, we apply the RPA to an artificial domain to demonstrate the new pruning method. Secondly, we test the RPA in the LED domain ([Breiman84]) where varying levels of noise can be controlled. Finally, we apply the RPA to the well known Soybean domain ([Michalski80]) and compare the performance with other pruning methods.

This section describes the implementation of the RPA and presents the assumptions and conditions generally involved in all domains. Other assumptions or conditions which are specific to a particular domain are described in the corresponding section.

5.3.1 Implementation of RPA and the CN2 Test-bed

The RPA was implemented in C language under the UNIX environment running on a SUN SPARC-2 workstation. The CN2 rule induction learning program version 6.1 ([Boswell90]) is integrated with the RPA program so that we can test the RPA's pruning ability and compare its performance with the CN2 significance test pruning method, which represents current pruning methods.

There are four user-specified parameters with CN2 program version 6.1. They are: (1) Algorithm: specify whether CN2 is to produce Ordered or Unordered sets of rules. (2) Error estimate: specify whether CN2 is to use the Laplace or Entropy/Apparent accuracy estimate to assess the rule accuracy. (3) Star size: specify the beam search width. (4) Threshold: specify the significance test level. The default parameters of the CN2 program are shown in Table 5.1 below.

Table 5.1 CN2 Default Parameters

Algorithm	Error estimate	Star size	Threshold
Unordered	Laplace	5	0

We choose the default CN2 parameters in our experiments: Firstly, the Unordered algorithm combined with the Laplace accuracy estimate generate more general rules which are also easier to understand and interpret for the user. Secondly, we use the default Star size as it is an heuristic parameter and is less important for our experiments. Finally, since the objective of our experiments is to show the pruning ability of the RPA, we switch off the significance test (i.e., Threshold = 0) when we apply the RPA. We apply the significance test (e.g., raise the threshold level) only when we need to compare its performance with that of the RPA.

5.3.2 Types of Noise Experimented in This Thesis

Since our research work involves dealing with noise, a way to simulate and control noise must be included.

Specifically, two different types of noise are considered in this thesis: *Attribute noise* and *Class noise*. The definition for Attribute noise is as follows: Suppose a noise level of n percent has been applied to some attribute A in the training examples. Whenever an object's description is generated, the true value of A for that object will be replaced with n percent probability by an randomly chosen incorrect value ([Quinlan86b]). This definition models Attribute noise as the occasional, nonrepeatable substitution of a possibly incorrect value for the true attribute value. Similarly, Class noise is defined as follows: Given a n percent noise rate to each of the classes in the training examples, a Class noise is introduced by replacing the true value of a class name, under n percent probability, with a randomly chosen incorrect

value.

5.3.3 Criteria for Evaluations of Pruning Results

The RPA is a new pruning method. To evaluate its pruning ability, we have to compare its performance on the noisy training data with other pruning methods including the CN2 significance testing.

Two objective evaluation criteria are employed in this thesis. They are *prediction accuracy* on new testing data and the *simplification* of classification rules. A rule is termed as *general* in the sense that it is both simple and predictive. The prediction accuracy of a rule is evaluated according to its performance on new testing data. To compare the simplification between different rule sets, both the number of rules and the number of conjunctions or conditions per rule are taken into account.

5.4 Experiments with An Artificial Domain

5.4.1 Objectives

The objective of this experiment is to demonstrate the new pruning method by experimenting the RPA in an artificial domain. Specifically, in this experiment, we create an artificial domain and introduce Class noise into it. By experimenting with this particularly noisy domain, we show that how the Class noise leads the CN2 rule induction program to overfit the training data and why the CN2 significance test pruning method can not solve the overfitting problem. Then we apply the RPA to this domain to demonstrate that this new pruning method can overcome the limitations of the CN2 significance test pruning method and improve the CN2's performance against the noisy training data.

5.4.2 The Artificial Domain and Data Sets

This artificial domain was created for experimental purposes. It contains two classes (*A* and *B*) while each class has two attributes (*X* and *Y*) with continuous numeric values. The two classes were chosen to be linearly separable, which makes the experiments easy to clarify. Appendix I shows an example data set which was created from this artificial domain and Figure 5.2 shows the 2-D plot for this data set.

Figure 5.2 An example data set from the artificial domain

As can be seen from Figure 5.2, this data set contains 35 examples: 17 of class *A* and 18 of class *B*. Class *A* and *B* are linearly separable along attribute *Y*. All of the examples of class *A* have $Y < 9.5$ while all of the examples in class *B* have $Y > 9.5$. Trained with this data set under default parameters, CN2 generated an unordered rule list which is shown in Figure 5.3. As can be seen from Figure 5.3, there are three unordered rules in this list and the last one is a default rule. The first rule covers all of the examples in class *A* and the second rule covers

```

IF      Y < 9.50
THEN    class = A      [17 0]

IF      Y > 9.50
THEN    class = B      [0 18]

(default) class = B      [17 18]

```

Figure 5.3 The unordered rule list generated from the example data set

all of the examples in class B. These two rules are general because they are very simple and also reflects the linearly separable nature of this domain. The default class was chosen to be class B even though the number of examples in class B are just one more than the number of examples in class A.

Then we introduced a Class noise into this domain. We created a new data set, Set_1, by adding one more example(#36) into the above 35-example set. The new example #36 fell into the domain of class A but had the incorrect class name of B. So, it was a Class noise in Set_1. We used this noisy data set in our experiments to study the effect of Class noise to the formation of classification rules. The 2-D plot of Set_1 is shown in Figure 5.4 below.

Similarly, we created another new data set, Set_2, by repeating the same steps for creating Set_1 (adding one more example into the original 35-example data set). However, we ended up with a normal situation this time, i.e., the new example #36 was a normal class A example in Set_2. As a result, Set_2 can be treated as a new data set to Set_1 as it contains fresh data. We used Set_2 in our experiments as a pruning set for the RPA. Figure 5.5 shows Set_2.

Figure 5.4 Data Set_1 which contains a Class noise (#36)

5.4.3 Experimental Results and Analysis

5.4.3.1 Effects of Class Noise on the Formation of Classification Rules

In the first experiment, we applied CN2 with its default parameters to Set_1 to show how the Class noise (example #36) affects CN2 during its rule generation. Trained with Set_1 under CN2 default parameters (i.e., without the significance test), CN2 generated six rules including a default one. These unordered rules are shown in Figure 5.6.

Comparing the rules in Figure 5.6 with those in Figure 5.3, we can observe the effects of the Class noise to the CN2's rule generation. Due to the presence of one Class noise (example #36), the number of rules increase from two to five (the default rule is not taken into account) and the five rules are more specific, e.g., two of the five rules are so specific that each

Figure 5.5 Data Set_2 which contains a fresh example (#36)

rule only covers one particular example. The performance of these rules was perfect (100% overall) on the training set (Set_1) itself. However, when these rules were applied to the new data set (Set_2, e.g., a testing set), the performance dropped to 94.4% for class A and 97.2% overall because one class A example in Set_2 (#36) was misclassified (by rule #5) to class B. The reason that rule #5 misclassified the new example is that this rule was generated to fit the Class noise (example #36 in Set_1).

Through this example, we can see that the CN2 rule induction algorithm tends to overfit the training data as it treats every training example including the noisy case as normal examples.

5.4.3.2 Results of Applying CN2 Significance Test

Since the CN2 significance test was intended to prune out overfitting rules, we applied

```

IF      5.50 < Y < 9.50
THEN    class = A      [11 0]

IF      Y < 4.50
THEN    class = A      [5 0]

IF      12.75 < X < 13.50
THEN    class = A      [1 0]      (example #17)

IF      Y > 9.50
THEN    class = B      [0 18]

IF      10.50 < X < 11.75
and     Y < 5.50
THEN    class = B      [0 1]      (example #36)

(default) class = B      [17 19]

```

Figure 5.6 The unordered rule list generated from Set_1 with Threshold 0

it in the second experiment and tried to improve CN2's performance. Specifically, we changed the Threshold parameter step by step in this experiment. In each step, the threshold level was raised by five and rules were re-generated and their prediction accuracy was tested on Set_1 (the training set) and Set_2 (the testing set) respectively. Table 5.2 shows the results from each step.

Table 5.2 Results of applying CN2 significance test

Threshold	# of Rules (including the default rule)	Overall Accuracy on Set_1 (%)	Overall Accuracy on Set_2 (%)
0	6	100	97.2

Table 5.2 Results of applying CN2 significance test

Threshold	# of Rules (including the default rule)	Overall Accuracy on Set_1 (%)	Overall Accuracy on Set_2 (%)
5	4	97.2	94.4
10	3	83.3	80.6
15	3	83.3	80.6
20	2	52.8	50.0
25	1 (default)	52.8	50.0

As seen in Table 5.2, raising the Threshold from zero to five reduces the number of rules from six to four but the overall accuracy also slightly decreases. The new rules generated from Set_1 with Threshold 5.00 are shown in Figure 5.7 below.

IF	$5.50 < Y < 9.50$	
THEN	class = A	[11 0]
IF	$Y < 4.50$	
THEN	class = A	[5 0]
IF	$Y > 9.50$	
THEN	class = B	[0 18]
(default)	class = B	[17 19]

Figure 5.7 The unordered rule list generated from Set_1 with Threshold 5

As seen in Figure 5.7, the two specific rules in Figure 5.6 which only cover a single example have been eliminated with Threshold 5.00 but other rules remain unchanged. When we evaluated these rules on Set_1 and Set_2, one class A example in Set_1 (#17) was misclassified by the default rule to class B and two class A examples in Set_2 (#17 and #36) were

misclassified into class B. As a result of these misclassifications, the overall accuracy on Set_1 and Set_2 both slightly decreased. Continually raising the threshold level, we got the same results, i.e., the same early rules tended to appear in the rule list and the prediction accuracy also decreased. Appendix II shows the unordered rule lists which were generated with other threshold levels.

As seen in this experiment, the CN2 significance test pruning method acts solely as a termination criterion. Raising the threshold level reduces the number of rules but also decreases the prediction accuracy. We can not get the same general rules from this noisy domain as those in Figure 5.3 with the CN2 significance test pruning method.

5.4.3.3 Results of Applying RPA

In the third experiment, we switched off the CN2 significance test (Threshold = 0) and applied the RPA to this noisy domain. Specifically, we used Set_1 as the training set and Set_2 as the pruning set. The RPA was applied to prune the training data in Set_1 with the help of the fresh data in Set_2.

At the end of the first iteration, the RPA pruned out the example #36 from Set_1 as removing this example led CN2 to generate new rules which had better prediction accuracy (100% overall) on the pruning set (Set_2) which contains fresh data. The RPA stopped pruning Set_1 at the end of the second iteration as it could not find any more “worst” examples in Set_1 that were “harmful” to the rule generation. The pruning results are shown in Table 5.3 below.

Table 5.3 Results of applying RPA

# of Iterations	The Example Removed from Set_1	# of Rules (including the default rule)	Overall Accuracy on Set_1 (%)	Overall Accuracy on Set_2 (%)
0	none	6	100	97.2

Table 5.3 Results of applying RPA

# of Iterations	The Example Removed from Set_1	# of Rules (including the default rule)	Overall Accuracy on Set_1 (%)	Overall Accuracy on Set_2 (%)
1	#36	3	97.2	100

As seen in Table 5.3, the accuracy on Set_1 slightly decreases as the new rules generated from the pruned training set misclassified the example (#36) in Set_1. However, the prediction accuracy on the pruning set (Set_2) which is noise free increases to 100% overall. The number of rules are also reduced from six to three. The new rules generated after the pruning are shown in Figure 5.8.

IF	Y < 9.50	
THEN	class = A	[17 0]
IF	Y > 9.50	
THEN	class = B	[0 18]
(default)	class = B	[17 18]

Figure 5.8 The unordered rule list generated from Set_1 with RPA

As can be seen in Figure 5.8, these rules are general to this noisy domain as they reflect the linearly separable nature of this domain even though they do not cover one particular example (#36) in the training set (Set_1). In this particular experiment, we know that the removed example (#36) is indeed a Class noise and should not be treated as a normal case by the rules.

As seen in this experiment, applying the new pruning method -- the RPA can lead CN2 to generate general and reliable rules, which is impossible to achieve by applying the current pruning approach (the CN2 significance test).

5.4.4 Summary

In this section, we used a small artificial example to show the influence of Class noise on the performance of CN2 rule induction algorithm. The experiments show that CN2 tends to overfit the noisy training set, generating complicated and specific rules instead of simple and general rules. As a result, the number of rules increases but the prediction accuracy on new examples decreases.

The experiments also show the limitations of the current pruning approach -- the CN2 significance test pruning method. The significance test acts solely as a termination criterion as CN2 tends to generate the same early rules in the rule list but just terminate earlier under higher level of threshold. As a result, raising the threshold level reduces the number of rules but also decreases the overall prediction accuracy both on the training set and on the testing set.

The RPA, however, shows its strength as a new pruning method. In our experiments, the RPA pruned the noisy training set with the help of the pruning set which contained fresh data. By removing noisy training data, the RPA led CN2 to generate different rules which were not in the previous rule list. This is, however, impossible with the CN2 significance test. As a result of applying the RPA, CN2 can generate general rules. The rules may not classify all the training examples correctly, but they perform well on new data. The pruning process with the RPA is also automated.

5.5 Experiments with the LED Domain

5.5.1 Objectives

The objective of experimenting with the LED domain is to test the RPA in a small real

world domain which also contains Attribute noise. Specifically, we want to investigate how the Noise/Signal ratio in a data set affects the degree of the overfitting problem. We apply the RPA on different-sized data sets under the normal attribute noise rate and also on data sets with higher attribute noise rate. The results are compared with those from other pruning methods including CN2 significance test.

5.5.2 The LED Domain

The LED display domain was introduced by Breiman ([Breiman84]). This simple domain contains seven Boolean attributes, one for each light-emitting diodes of the LED display. All attribute values are either 0 or 1, according to whether the corresponding light is on or off for the decimal digit. There are none missing attribute values in this domain. However, each correct attribute value has the 10% probability of having its value inverted, i.e., each element of a display is subject to a 10% random error. There are ten classes in this domain, one for each decimal digit in the set of {0 to 9}. Each class (digit) has the same theoretical distribution probability which is 10%. The task is to recognize the correct digit in a LED display with faulty elements. This requires that the learning system be able to handle noisy attributes in the training and testing cases.

[Breiman84] has described that for this 10% attribute noise rate, the upper bound of the performance of any learning system in the LED domain is 74%. Table 5.4 shows the performance of different learning systems on this domain.

Table 5.4 Performance of Previous Learning Systems in the LED Domain

Learning System	# of Training Cases	# of Test Cases	Classification Rate	Reference
Optimal Bayes Classification Rate	200	5000	74%	[Breiman84]
CART Decision Tree Algorithm	200	5000	71%	[Breiman84]

Table 5.4 Performance of Previous Learning Systems in the LED Domain

Learning System	# of Training Cases	# of Test Cases	Classification Rate	Reference
Nearest Neighbor Algorithm	200	5000	71%	[Breiman84]
C4 Decision Tree Algorithm	2000	500	72.6%	[Quinlan87]
IWN System (Add-OR)	400	500	73.3%	[Tan88]

5.5.3 Experimental Methods

[Tan88] shows that the performance of the IWN learning system ([Tan88]) on the LED domain varies with the number of training examples. When the number of training examples increases, the performance increases and gradually reaches the upper bound of 74%. In our experiments, we did experiments on different sizes of data sets under the normal 10% attribute noise rate. We also did experiments on the same size of data sets but under different noise rates.

In order to evaluate the pruning results, a pure data set which contains 50 examples was firstly generated under zero attribute noise rate. Trained with this pure data set, CN2 got 10 pure rules, one for each class (digit). These rules were used in the experiments to evaluate the pruning results. Comparing the rules generated from the pruned training set with the pure rules, we could find out whether the new pruning method actually removed noise and led CN2 to generate simple and general rules. This is one of the advantages of experimenting with the LED domain.

During the experiments with the LED domain, we relaxed the stop condition of the RPA. In the previous experiments with the artificial domain, the RPA was stopped when the

new accuracy of removing another training example was less than or equal to the current best accuracy (corresponding to the condition "if New_accuracy > Best_accuracy" on line 25 of the RPA in Figure 5.1). In the experiments with the LED domain, we changed line 25 of the RPA to "if New_accuracy >= Best_accuracy", i.e., the RPA would be stopped only when the new accuracy of removing another training example was less than the current best accuracy. As a result of relaxing the stop condition, we pruned out some "redundant" training examples (those examples whose removal led to the "=" relation in the condition "if New_accuracy >= Best_accuracy", i.e., the new accuracy of removing a "redundant" training example is equal to the current best accuracy) besides noise, however, we achieved better performance in the LED domain as a result of this change.

Another difference between the experiments with the LED domain and the previous experiments with the artificial domain is that the pruning set in the LED domain also contains noise, which is ordinary in real world applications. Since the RPA uses both the training set and the pruning set for pruning, we need a third data set which consists of new testing data to verify that the RPA does not lead CN2 to overfit the pruning set. In the experiments, the three data sets were taken from the LED domain independently but under the same conditions. The size of each data set was chosen to be the same, which made the comparison unbiased.

5.5.4 Experimental Results and Analysis

5.5.4.1 Experiments under Normal Attribute Noise Rate

In the first set of experiments, we introduced 10% normal attribute noise to each data set generated from the LED domain. Specifically, three groups of data sets were generated: Group A, B, and C. These Groups differed by the size of each data set in the group. The three different sizes of data sets were 50, 100, and 200 (examples per data set) for Group A, B, and C respectively. Each group consisted of three data sets: a training set, a pruning set and a testing set.

5.5.4.1.1 Data Set Size, Noise/Signal Ratio, and the Overfitting Problem

In this experiment, we studied the effect of the overfitting problem on each of the three data groups. In each group, CN2 was trained with the training set and the rules which were generated from the training set without any pruning were tested on all three data sets. Table 5.5 shows CN2's performance in each group. The "drop of accuracy" in the table was calculated as the difference between the training accuracy and the average of the two testing accuracies (i.e., the accuracy on the pruning set and the accuracy on the testing). The drop of accuracy reflected the effect of the overfitting problem.

Table 5.5 CN2's performance: 10% normal attribute noise rate; No pruning

Group	# of examples per data set	Training Accuracy	Accuracy on the Pruning Set	Accuracy on the Testing Set	Drop of Accuracy
A	50	90%	52%	42%	43%
B	100	79%	68%	67%	11.5%
C	200	79.5%	73.5%	71.5%	7.5%

As seen in Table 5.5, the effect of the overfitting problem on CN2's performance varies with the size of the data set, under the normal attribute noise rate. The effect of the overfitting problem is greater in smaller data sets than in larger data sets. For example, the drop of accuracy in Group A (50-example per data set) is very high compared with the drop of accuracy in Group C (200-example per data set). In other words, under the normal attribute noise rate, rules generated from a smaller training set tend to be more specific and overfit the training set. The reason behind is that the (Class) Noise/Signal ratio varies with the size of the data set under the same attribute noise rate: the smaller is a data set, the higher is the Noise/Signal ratio. Table 5.6 shows the Noise/Signal ratio in Group A under the 10% normal attribute noise rate.

Table 5.6 Noise/Signal ratio in Data Group A under 10% normal attribute noise rate

Data Set	# of Data per set	# of Class Noise	Noise/Signal Ratio
The training set	50	29	58%
The pruning set	50	22	44%
The testing set	50	25	50%

As seen in the table, the Noise/Signal ratio in each data set is much higher than the 10% Attribute noise rate. As a result, rules generated from the high Noise/Signal ratio training set are more likely to overfit the noisy data, which shows the lower prediction accuracy on new testing data. However, when the data set size increases, the Noise/Signal ratio decreases as the same amount of attribute noise is spread over a larger number of examples. As a consequence, the effect of the overfitting problem is lighter on large-sized data sets.

5.5.4.1.2 Pruning Results in Data Group A

In this experiment, we firstly applied the significance test pruning method to improve CN2's performance in Group A, which contains small-sized data sets. Table 5.7 shows the experimental results.

Table 5.7 CN2's performance in Data Group A with the significance test

Threshold	Accuracy on the Training Set	Accuracy on the Pruning Set	Accuracy on the Testing Set
0	90%	52%	42%
10	88%	52%	42%
15	82%	52%	42%
20	84%	50%	42%
25	66%	44%	34%

As seen in the table, raising the threshold level can not improve CN2's performance in

terms of the accuracy on new test data.

Secondly, we applied the RPA on the same data sets without the significance test. The training set was continually pruned with the help of the pruning set. In each pruning iteration, new rules were generated from the pruned training set. The accuracy was tested on all three data sets. The RPA was stopped when further pruning would deteriorate the prediction accuracy on the pruning set. Table 5.8 shows the experimental results. The same results are drawn

Table 5.8 Results of Applying RPA in Data Group A: Prediction Accuracy

# of Iterations	Type of Examples Removed	Accuracy on the Training Set (%)	Accuracy on the Pruning Set (%)	Accuracy on the Testing Set (%)
0	-	90	52	42
1	noise	92	58	52
2	noise	90	64	66
3	redundant	90	68	64
4	redundant	90	68	64
5	redundant	90	68	66
6	noise	90	68	64
7	noise	90	68	64
8	noise	90	68	64
9	noise	90	68	64
10	noise	90	68	66
11	noise	90	74	66
12	noise	88	76	66
13	redundant	88	76	66
14	redundant	88	76	66
15	redundant	88	76	66
16	redundant	88	74	66

on a 2-D plot in Figure 5.9.

Figure 5.9 Results of applying RPA in Data Group A

As can be seen from the above results, the RPA stabilizes the performance after 12 iterations. The training accuracy slightly drops from 90% to 88%. However, both the pruning and testing accuracy increase about 24%.

Comparing the rules generated before and after applying the RPA with the pure rules, we found that the new rules generated from the pruned training set were also simpler and more general. There are 10 pure rules describing the ten classes (digits) in this domain. CN2 generated 14 rules from the original training set before applying the RPA. After applying the RPA, CN2 generated 12 less complicated rules from the pruned training set. For example, CN2 generated 2 rules to describe Class One (digit "1") and Class Four (digit "4") before applying the RPA. After 12 pruning iterations, CN2 generated only 1 simple rule to cover all

of the examples in each of the two classes (One and Four). More than this, the number of conjunctions in each of the rules were also reduced, which reflected the down-sized rule complexity. Table 5.9 shows the comparison of rule complexity before and after applying the RPA for each of the ten classes (digits).

Table 5.9 Results of applying RPA in Data Group A: Rule Simplification

Classes (Digits)	# of Rules Before Pruning	# of Rules After Pruning	# of Conjunctions Before Pruning	# of Conjunctions After Pruning
0	1	1	4	3
1	2	1	2	2
			3	-
2	2	2	3	3
			3	3
3	1	1	5	4
4	2	1	3	2
			3	-
5	1	1	4	3
6	1	1	2	3
7	1	1	3	2
8	2	2	5	4
			3	3
9	1	1	4	4

For some of the classes in Table 5.9, although the number of rules or the number of conjunctions are not changed, we found out that the contents of the conjunctions was changed to be similar to those in the pure rules after our investigation of the rule contents in detail.

Also seen in Table 5.8, eight out of the first twelve training examples removed during the pruning process are real noise and the rest 4 are redundant examples. After removing the first two noisy examples, both the pruning and the testing accuracy increase significantly. This

shows that Firstly, the noise affects the rule generation which is a problem that the significance test can not overcome. Secondly, the RPA can prune out the noise from the training set to lead CN2 to generate simpler and more general rules from the pruned training set. These rules thus may not classify all the original training examples correctly, but they perform well on new data. Thirdly, the RPA also removes redundant training examples besides noise. It shows that redundant data may also have negative influence on the rule generation. This is worth while conducting future research.

5.5.4.1.3 Pruning Results in Data Group B

In this experiment, we applied the RPA in Group B which contains middle-sized data sets among the three data groups. The results are shown in Figure 5.10 below.

Figure 5.10 Results of applying RPA in Data Group B

As seen in Figure 5.10, the RPA also performed well on Group B data sets. The train-

ing accuracy dropped 5% (from 79% to 74%). The accuracy on the pruning set increases 9% (from 68% to 77%) and the accuracy on the testing set also increased 6% (from 67% to 73%). As a result, the drop between the accuracy on the training set and the accuracy on the testing set decreased from 12% (79% to 67% before applying the RPA) to only 1% (74% to 73% after applying the RPA). The accuracy on the testing set (73%) almost reached the upper bound (74%) in this domain. This experiment shows that the RPA improved CN2's performance against middle-sized noisy data sets.

5.5.4.1.4 Pruning Results in Data Group C

Similarly, we applied the RPA in Data Group C which contains the largest size data sets among the three groups. The results are shown in Figure 5.11 below.

Figure 5.11 Results of applying RPA in Data Group C

As seen in the results, the RPA's performance in Data Group C which has 200 cases in

each data set, is not so successful as the results in Group A and B. The accuracy on the testing set slightly increased from 71.5% to 72.5% and then slightly decreased to less than 70%. Although the accuracy on the pruning set continuously increased from 73.5% up to 79.5% which was even better than the accuracy on the training set at this level, the decreased accuracy on the testing set shows that the RPA might have led CN2 overfit the pruning set. From this experiment, we can understand the importance of using the testing set to verify the pruning results.

The reason that the RPA is not successful in Group C is that the effect of the overfitting problem is actually very light in this group. As can be seen from Table 5.5, the drop from the training accuracy to the testing accuracy is much less in Group C than in Group A or B. The testing accuracy in Group C is also already close to the upper bound accuracy that any learning systems can achieve in this domain (73.5% or 71.5% compared with 74%) before applying any pruning. This experiment shows that the RPA is more useful on high Noise/Signal ratio data sets such as the data sets in Group A and B, where the effect of the overfitting problem is great. This is further supported by our experiments on data sets with higher level attribute noise rate, which is shown below.

5.5.4.2 Experiments under Higher Attribute Noise Rate

In the last set of experiments, we created a new data group, Group D. Each data set in Group D contains 100 examples with 15% attribute noise rate. The experimental results are compared with those in Data Group B which has same data set size but with lower attribute noise rate (10%). Table 5.10 compares CN2's performance in these two data groups before applying any pruning techniques.

Table 5.10 Effects of the overfitting problem under different attribute noise rates

Data Group	# of Examples in Each Data Set	Attribute Noise Rate	Training Accuracy	Accuracy on the Pruning Set	Accuracy on the Testing Set	Drop of Accuracy
B	100	10%	79%	68%	67%	11.5%
D	100	15%	77%	56%	54%	22%

As seen in Table 5.10, the effect of the overfitting problem in Data Group D is greater than that in Data Group B. The reason is that the attribute noise rate in Group D is higher than that in Group B. With the same data set size, the higher attribute noise rate leads to the higher Noise/Signal ratio which in turn causes the greater overfitting problem. Figure 5.12 shows the results of applying the RPA in Data Group D.

Figure 5.12 Results of applying RPA in Data Group D

Comparing Figure 5.12 with Figure 5.10, the RPA performs slightly better in Group D than in Group B. The accuracy on the testing set increases 10% in Group D compared with 5% in Group B. This experiment verifies that the RPA performs better on high Noise/Signal ratio data sets.

5.5.5 Summary

In this section, we applied the RPA to the LED display domain which contains Attribute noise. Previous experimental results by other researchers have shown that the upper bound accuracy can be achieved by any learning systems in this domain is 74%. However this upper bound can be achieved only on fairly large data sets. [Tan88] has shown that the performance of the IWN learning system was poor on small data sets. This has been supported by our experiments, applying CN2 on different-sized data sets. We found that the effect of the overfitting problem varies with the data set size and the attribute noise rate. As remarked earlier, under the same attribute noise rate, the overfitting problem varies as the Noise/Signal ratio which in turn varies as the data set size. In other words, the smaller is the data set, the higher is the Noise/Signal ratio and thus the greater is the effect of the overfitting problem. On the other hand, with the same data set size, the overfitting problem varies as the attribute noise rate, i.e., the higher is the attribute noise rate, the greater is the effect of the overfitting problem.

In our experiments, the RPA showed its strength as a pruning method in the noisy LED domain especially when the effect of the overfitting problem is great. We showed that the RPA performed well in Data Group A, B, and D where the effect of the overfitting problem was obvious. We showed that after applying the RPA to prune the training set with the help of the pruning set (which also contains noise), CN2's performance had been improved. The accuracy on the testing set increased. The drop from the training accuracy to the testing accuracy was also reduced. the RPA did prune out noisy training data and generate simpler rules which were

similar to the pure rules. In the experiments of applying the RPA on Group C data sets, we showed that the importance of using the testing set to verify the pruning results, which can prevent the RPA from overfitting the pruning set. This is more important in real world applications where the pruning set contains noise as well.

5.6 Experiments with the Soybean Domain

5.6.1 Objectives

The objective of experimenting the RPA with the Soybean domain is to test the RPA on another small real world domain which also contains unknown attribute values besides noise. As the Soybean domain is well known and has been widely used in the machine learning area, we also want to compare the RPA's pruning results with the performance of other learning systems in this domain.

5.6.2 The Soybean Domain and Data Sets

The Soybean disease database was first used by Michalski to develop and evaluate the AQ11 learning algorithm ([Michalski80]). Now, it has been widely adopted to evaluate various learning systems. For example, it has been used in ([Tan88]) to compare the performance of the IWN learning system with ID3 ([Quinlan86a]) and AQ11/15 ([Michalski83], [Michalski86]). Clark et.al. also used it to compare CN2 ([Clark89], [Clark91]) with C4.5 ([Quinlan87]). We obtained this database from the UCI Repository of Machine Learning Databases.

There are 19 classes in the original database. However, only the first 15 of which have been used in prior work. The reason seems to be that the last four classes are unjustified by the data as they have so few examples. There are 35 categorical attributes, some nominal and

some ordered. There are also unknown attribute values in this domain. The original database contains a training set with 290 training examples and a testing set with 340 testing examples.

5.6.3 Experimental Methods

Since there are only two available data sets in the Soybean domain, the training set and the testing set, we have two approaches to apply the RPA in this domain. First, we can use the training set to train CN2 and use the testing set to help the RPA prune the training set and to test the pruning results. As a result, the testing set in this approach is not wholly independent as we also use it for pruning. This approach has been adopted by Mingers in [Mingers89]. Second, we can split the training set into two halves, one used for training and another used for pruning. In the second approach, the testing set is thus wholly independent as it is used for testing only. However, in this case, the number of training examples left is only one half of the original training set, which makes the pruning results hard to be compared with the results of other learning systems which have conducted learning on the original training set.

In our experiments, we combined the above two approaches: Firstly, we adopt the second method. We randomly split the original training set into two data sets, one is used as the training set and the other is used as the pruning set. The pruning results are verified on the independent testing set. Secondly, we adopt the other method. We use the testing set both as the pruning set and the testing set. By adopting this approach, we want to compare our pruning results with the performance of other learning systems.

We also relax the stop condition of the RPA in this domain as we did in the LED domain.

5.6.4 Experimental Results and Analysis

5.6.4.1 Experiments on Split Data Sets

In this experiment, we split the original training set into two data sets, the training set and the pruning sets. The way we split the original training set is to randomly split it into two equal halves, i.e., 50% of the original training data are in the new training set and another 50% in the pruning set. However, the original training examples are not equally distributed among the 15 classes (the distribution is shown in Table 5.11). As we hope that either of the two new data sets may contain examples from all 15 classes, we randomly pick up 50% examples from each of the 15 classes to construct the training set and leave the rest as the pruning set.

Table 5.11 The Distribution of Original Soybean Training Examples among All Classes

Class Number	Class Name	# of Examples
1	diaporthe-stem-canker	10
2	charcoal-rot	10
3	rhizoctonia-root-rot	10
4	phytophthora-rot	40
5	brown-stem-rot	20
6	powdery-mildew	10
7	downy-mildew	10
8	brown-spot	40
9	bacterial-blight	10
10	bacterial-pustule	10
11	purple-seed-stain	10
12	anthracnose	20
13	phyllosticta-leaf-spot	10
14	alternarialeaf-spot	40
15	frog-eye-leaf-spot	40

After we had created the three data sets, the CN2 was trained with the new training set and the rules were tested on both the pruning set and the testing set. The results are shown in Table 5.12 below.

Table 5.12 CN2's Performance on Split Soybean Data Sets

Data Set	# of Examples	Accuracy
The Training Set	145	97.9%
The Pruning Set	145	82.1%
The Testing Set	340	81.8%

As seen in the table, the rules performed very well on the training set but their accuracy on the pruning set and the testing set dropped 15%. It shows that CN2 overfit the new training set. However, the accuracy on the pruning set and the testing set are very close. This partly shows that the new pruning set, which was randomly picked out from the original training set, still reflects the nature in this domain.

In order to improve CN2's performance on split data sets, we applied the RPA to prune the new training set with the help of the pruning set and tested the pruning results on the independent testing set. Figure 5.13 shows the pruning results on a 2-D plot.

As can be seen from Figure 5.13, the RPA improved CN2's performance on split Soybean data sets. Training accuracy slightly decreased. However, the accuracy on the independent testing set increased from 81.8% to 86.8% and the accuracy on the pruning set also increased from 82.1% to 89.7%.

5.6.4.2 Experiments on Original Data Sets

In this experiment, we applied CN2 significance test on the original Soybean data sets firstly. Table 5.13 shows the pruning results with different significance test levels. As can be seen in Table 5.13, the significance test could not improve CN2's performance. Raising the

Figure 5.13 Performance of RPA on Split Soybean Data Sets

threshold level slightly decreased both the training accuracy and the testing accuracy.

Table 5.13 Performance of CN2 Significance Test on Original Soybean Data Sets

Threshold	Training Accuracy	Testing Accuracy
0	96.6%	90.3%
5	96.2%	90.3%
10	94.8%	89.7%
15	94.5%	89.4%

In order to improve CN2's performance in this domain, we applied the RPA on the original Soybean data sets. Specifically, the original training set, which contains 290 examples, was used to train CN2 and the testing set, which contains 340 examples, was used to help the RPA prune the training set. The results of applying the RPA on the original Soybean data sets are shown in Figure 5.14 below.

Figure 5.14 Performance of RPA on Original Data Sets

As seen from Figure 5.14, the testing accuracy has been increased significantly from 90.3% to 96.2%. The difference between the training accuracy and the testing accuracy was also reduced, i.e., the overfitting problem was reduced.

Table 5.14 shows the comparison of the performance of different learning systems on the original Soybean data sets including the CN2's performance with the RPA and with the significance test. Some of the results are extracted from [Tan88].

Table 5.14 Comparison of Performance on Original Soybean Data Sets

Learning Systems	Testing Accuracy
AQ11/15	98.0%
IWN (Max-OR)	97.1%
IWN (Add-OR)	96.6%
CN2 (with the RPA)	96.2%

Table 5.14 Comparison of Performance on Original Soybean Data Sets

Learning Systems	Testing Accuracy
ID3	92.4%
CN2 (with the Significance Test)	90.3%

As seen in Table 5.14, before applying the RPA, CN2's performance on the Soybean data sets was much lower than other learning systems. However, after applying the RPA to prune the training set, CN2's performance was greatly improved. The testing accuracy now is much better than both the accuracy of ID3 and the accuracy of CN2 with the significance test. CN2's performance with the RPA is now also at the top level and can be compared with those of IWN and AQ11/15 systems. This experiment shows that the RPA has proven efficient in reducing the overfitting problem.

5.6.5 Summary

In this section, we showed the successfulness of the RPA on the Soybean disease database which is another small real world domain that also contains noise and unknown attribute values.

The experimental results showed that the RPA improved CN2's performance on both the split Soybean data sets and the original Soybean data sets in terms of the testing accuracy. Comparing CN2's performance with other learning systems in the Soybean domain, we showed that the performance of CN2 with the significance test was much lower than the performance of other systems such as ID3 ([Quinlan86a]), IWN ([Tan88]), and AQ11/15 ([Michalski83], [Michalski86]). It was the RPA that improved CN2's performance and also reduced the overfitting problem.

5.7 Limitations and Improvements

Experimenting with the RPA, we found that the RPA consumes a fairly amount of computing time especially when the training set contains a large number of examples. In each of the RPA iteration, the program has to remove every training examples from the current training set once and trains CN2 to generate new rules from the "leave-one-out" training set and tests the rules on the pruning set to find out the worst case. Under large number of training cases, it would take a lot of computing time for every iteration. Thus, the pruning speed becomes the bottleneck for applying the RPA on large data sets.

However, we also found out that during a particular pruning iteration, there may exist several least useful examples (i.e., removing them one at a time gains the same accuracy against the pruning set), especially when the training set is large. The original RPA algorithm only removes one of them (the first one) in an iteration. However, the remained cases tend to appear as being the least useful examples again in subsequent iterations. So, it wastes a lot of computing time to run additional iterations to remove them one at a time. Thus, we can speed up the RPA by modifying the original algorithm to remove all of the least useful examples found in one particular iteration. The modified RPA algorithm is shown in Figure 5.15.

We have done a small trial on an artificial data set. This modification greatly improved the RPA's pruning speed without any influence on the pruning result. This improvement let us apply the modified algorithm on large data sets to achieve better performance quickly.

5.8 Concluding Remarks

In this chapter, we developed and implemented a new pruning algorithm, the Regression Pruning Algorithm, using CN2 rule induction program as a test-bed. Although we used CN2 as a

Step 1: Divide set of examples into a training set and a testing set.

Step 2: Divide the training set equally into a training subset and a pruning subset.

Step 3: Learn CN2 rules from the training subset.

Step 4: Test CN2 rules (from step 3) against the pruning subset, gaining accuracy A_4 .

Step 5:

a) Remove one example $[i]$ from the training subset, relearn CN2 rules from the "leave-one-out" training subset and retest (relearned) CN2 rules against the pruning subset, gaining accuracy A_{5i} .

b) If A_{5i} is higher than any other A_{5j} ($j \neq i$), then remember i and A_{5i} as being the ONLY least useful example in the training subset.

Else If A_{5i} is among several other same (best) accuracies, then remember i , A_{5i} and all the other examples as being the MULTIPLE least useful examples in the training subset.

c) Repeat Step 5 a) and b) for all examples in the training subset.

Step 6: If $A_{5i} > A_4$, then remove example i (the least useful example) from the training subset if only one least useful example exists or remove all of the least useful examples including i if there are multiple least useful examples, and return to step 3.

Otherwise, the pruning is complete.

Figure 5.15 The Modified Regression Pruning Algorithm

test-bed, the RPA was not built into CN2. The CN2 learning system was just an independent learning algorithm to the RPA. Thus, the RPA can be applied to any other learning systems, too.

Our experiments on three task domains showed that the RPA is an useful pruning tool for rule induction learning systems to overcome the overfitting problems especially when the learning systems are applied to real world domains which may contain various types of noise. We showed that the RPA can deal with Class noise as well as Attribute noise. It can work with various domains which may have Continuous numeric attribute value, Boolean attribute value, Categori-

cal attribute value, and also Unknown or Missing attribute value. It can automatically pick out noise and lead the learning systems regenerate more general and reliable rules from the pruned training examples to achieve better predictive performance on the new test data. This algorithm has proven more useful in the high Noise/Signal ratio domain where the traditional learning algorithms can not achieve good results. The slow pruning speed on large number of training examples is a bottleneck for the applications of the RPA on large real world domains. However, the modification to the original algorithm can greatly accelerate the pruning speed to ease the bottleneck.

With the help of the pruning algorithm, the rule induction learning systems can be more easily applied on the real world domain. In the next chapter, we show our applications of rule induction learning systems on the network fault diagnosis and further test the RPA.

Chapter 6

The Application of CN2 Rule Induction Program in Network Fault Diagnosis

6.1 Introduction

Network fault diagnosis in general is based on the detecting and distinguishing normal versus abnormal network behavior ([Maxion90]). To be able to notice abnormal network behavior suggests that some internal representation or model of normal network behavior exists, against which observed events in the network can be compared. The models of normal network behavior can be acquired through learning ([Bisio92]).

The Pegasus project ([Pagurek91]), which was discussed in Chapter 2, employs a multiple paradigm approach for network fault diagnosis. As one of the diagnostic tools developed in this project, [Viens92] used neural network techniques to establish normal and abnormal traffic patterns for router fault diagnosis. The diagnosis in [Viens92] is to compare the observed router traffic pattern with the normal and abnormal models to detect possible router faults. Another diagnostic tool in this project, [Ferrell93], on the other hand, used statistical techniques to calculate the traffic contribution of a router to its surrounding area and compare the result with the normal contribution pattern. If a router's contribution varies from its normal pattern by a specific threshold level, an alarm will be generated to indicate the abnormal status of this router. These diagnostic tools have proven efficient in diagnosing various router faults. However, one of the problems with [Viens92] is that the neural network only takes as input the top two and bottom two

lines of a router. It is possible that in some cases particular lines are always the top performers and other particular lines are always the lowest performers. As a result, the system can not always flag the changes in traffic patterns correctly. In [Ferrell93], the base traffic patterns were established using the mean traffic over 24 hours. However, in real network, the traffic varies with the time of day and days of week. In order to flag the traffic variations correctly, it is desired to establish a set of base traffic patterns to account for any time of day or days of week traffic variations.

As one of the multiple approaches employed in the Pegasus project, this thesis applies machine learning techniques in network fault diagnosis. We show that machine learning techniques is useful in solving the problems experienced by [Viens92], and [Ferrell93]. Specifically, in this chapter, we build a general system model for applying machine learning techniques in network fault diagnosis. The CN2 rule induction program is applied to the traffic statistics of all the lines around a router to establish a set of normal traffic patterns to account for any time of day traffic variations. The traffic patterns are described by easily understandable rules generated by CN2. These CN2 rules predict the time of day for the traffic. If the time of day predicted is not the actual time of day, this indicates a fault. These rules thus can be used to build rule-based expert systems for the detection of router faults. We also show that the Regression Pruning Algorithm (RPA) developed in Chapter 5 is useful when we apply the CN2 rule induction program in the network domain.

6.2 A General System Model

In this thesis, we developed a general system model for applying machine learning tools in network fault diagnosis. It is shown in Figure 6.1. As seen in the figure, a network simulator is required in this model to substitute the real network, allowing the generation of anomalies in experiments. Observations of the network behavior are recorded in the network information database which constitutes the input for machine learning tools. The input to machine learning tools

Figure 6.1 The General System Model for Applying Machine Learning Techniques in Network Fault Diagnosis

may also contain domain knowledge which is extracted from domain experts interactively as initial rules and directives for deductive learning tools (if applicable). Machine learning tools then automatically synthesize diagnostic rules from the network information database for the corresponding network behavior. These diagnostic rules can be used later to build rule-based expert systems to help network operators monitor the performance of real networks.

Figure 6.2 shows the specific system model tailored from the general model to fit the specific application in this thesis. As seen in Figure 6.2, the network under experiment is the BNR Wide Area Network. The DRAGON ([Dawes92]) is the network simulator used to substitute the BNR network in our experiments. As remarked earlier in Chapter 2, the router is an important device in the BNR network. Network traffic is a network parameter that can be used to indicate the router's performance. In this thesis, we apply the CN2 rule induction program to help establish router normal line traffic patterns for diagnosing router abnormalities. A knowledge source

Figure 6.2 The System Model for Applying CN2 in the Development of Router Normal Traffic Patterns

called "traffic collector" was developed in this thesis to collect router line traffic parameters from the DRAGON simulator into the router line traffic database which constitutes the input to CN2. The rules generated from the router line traffic by CN2 are pruned by applying the Regression Pruning Algorithm developed in Chapter 5 to improve rules' prediction accuracy. The established router normal traffic patterns can be used in the future development of rule-based expert systems which can be used to monitor the real network.

In the next section, we apply this general system model to direct our development of router normal traffic patterns to account for any time of day network traffic variations.

6.3 The Development of Router Normal Traffic Patterns

The objective of this experiment is to obtain a set of rules to account for any time of day router traffic variations under normal network conditions. The reason that we only account for time of day traffic variations but not for days of week is that the current DRAGON network simulator only supports time of day traffic variations. However, the method applied in this experiment would be general enough to be used in other traffic variation conditions.

6.3.1 Experimental Conditions

As the objective is to establish normal router traffic patterns which vary with the time of day, we set up the following conditions for the Dragon simulator.

1. The network topology was configured by three BNR supplied standard text files: The NM2.NO file, which lists the devices and their connections in an object oriented description; The NM2.SY file, which describes the data collector names and locations; and the NM2.LK file, which describes how the serial lines are interconnected.
2. The ATM network was not set up and no workstations were randomly added in to generate extra network traffic load.
3. The traffic varies (as in real life) with the time of day.
4. A packet loss rate of 2 was chosen which is real and can reflect the normal network traffic load [Dawes92].
5. The simulator was set to run not as exactly the real time but to run as fast as possible so that we can get experimental results quickly.
6. The simulator function which can simulate the situation that network elements fail

randomly and then are repaired later was turned off.

A knowledge source called the Traffic-collector was developed that could poll the Dragon simulator through the Blackboard ([Iqneibi92]) to collect the line traffic data for a chosen router. The traffic collector polled the chosen router at a certain time interval (for example, every 1 minute) during a certain time period (for example, 1 day) for the traffic coming in and going out of each line connected to this particular router. The out-going traffic was recorded into a log file in the CN2 input file format.

The attribute in this domain is the line traffic. The number of attributes for a router depends on the number of lines connected to this particular router. For instance, if there are ten lines connected to router A, then each data file for router A contains ten attributes. In order to account for the time of day traffic variations, we assigned 24 classes for this domain, each for an one-hour period in a day. An example was assigned to one of the 24 classes according to the time when it was collected. For example, the example that was collected during 8:00am - 9:00am was considered as an example of class Eight.

In the experiments, we generated two independent data sets: a training set and a testing set. The training set was used to train CN2 to generate rules while the testing set was used to empirically verify the performance of the obtained rules. However, when we applied the Regression Pruning Algorithm to improve CN2's performance, we used three data sets: a training set, a pruning set and a testing set. Each data set was independently generated from the simulator under the same conditions and their sizes were the same.

6.3.2 Experimental Results and Analysis

The router bCARc2 was randomly chosen from all possible routers in the BNR network to be used in our experiments. There are 15 lines connected with this router. Figure 6.3 shows the connections of bCARc2 with other routers and LANs. The line number in Figure

6.3 corresponds to the attribute number in router traffic files

6.3.2.1 The First Experiment.

In the first experiment, the Traffic-collector polled the router bCARc2 every 1 minute for 24 hours to collect the line traffic data. So each of the training and testing data set contains 1,440 examples, 60 examples for each class. CN2 was trained with the training set and it generated a set of rules for each of the 24 classes. These rules were generated to describe normal traffic patterns on router bCARc2 in a day according to the traffic rate on each of the lines connected with bCARc2. Figure 6.4 shows a sample rule.

```

IF line1 < 22080.00
  AND line10 < 70.00
  AND 1330.00 < line13 < 1681.00
  AND 11845.00 < line14 < 14273.00
  AND 2655.00 < line15 < 3248.00

THEN class = one [8 2 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

```

Figure 6.4 A Typical Rule for Router bCARc2

As seen in Figure 6.4, one of the traffic patterns during 0:00am - 1:00am in a day (i.e., class One) is described in line traffic rate. When this rule was evaluated on the training set, 8 of the 60 class One training examples were covered by this pattern. This pattern also covered 2 training examples in class Two and 1 training example in class Eight. This overlapping may lead to misclassifications when this rule is evaluated on new testing data.

Generally, there exists a trade-off between the rule generality and the prediction accuracy. If a rule is too general, it will cover examples of other classes and lead to higher degree of misclassifications and lower prediction accuracy on both the training set and the testing set. On the other hand, however, if a rule becomes too specific, it will overfit the training examples and lead to lower prediction accuracy on the testing examples. Our goal is to make CN2 gen-

Figure 6.3 Router bCARc2 and its Connections

erate general and predictive rules, i.e., both the training accuracy and the testing accuracy are better to be high and at the same level.

In this experiment, the CN2 Unordered learning algorithm was used and the rule evaluation function was the Laplace Error Estimate. Table 6.1 shows the experimental results of applying the significance test to improve CN2's performance. Table 6.2 shows the results of

Table 6.1: Results of Applying the Significance Test

Star Size	Threshold	Training Accuracy (%)	Testing Accuracy (%)	Accuracy Drop (%)
5	0	56.0	47.1	8.9
5	10	55.1	46.4	8.7
5	20	53.9	45.9	8.0
5	30	51.0	43.3	7.7
5	40	48.3	41.6	6.7
5	50	41.8	36.2	5.6

using different Star Size without the significance test.

Table 6.2: Results of Using Different Star Size

Star Size	Threshold	Training Accuracy (%)	Testing Accuracy (%)	Accuracy Drop (%)
5	0	56.0	47.1	8.9
10	0	62.8	52.4	10.4
15	0	74.2	62.4	11.8
20	0	74.4	61.2	13.2

As seen in Table 6.1, raising the threshold level only terminates CN2 at earlier stage but can not change rules. Both the training and testing accuracies drop when more pruning is applied. Thus, the significance test can not improve CN2's performance in this domain. In order to get more general and reliable rules, we have to apply the Regression Pruning Algo-

rithm which was developed in Chapter 5.

As can be seen in Table 6.2 that both the training and testing accuracies are getting better when the Star Size increases. However, the drop from the training accuracy to the testing accuracy is also getting larger. It shows that CN2 tends to overfit the training set when the Star Size increases. There exists a trade-off between the accuracy and the overfitting problem. In this experiment, we got relatively good accuracy and less degree of the overfitting problem with the Star Size 15. So the set of parameters we adopt to use for the rest of experiments are: Unordered learning algorithm; Laplace Error Estimate; No Significance Test (Threshold = 0); and Star Size of 15.

As seen from Table 6.1 and Table 6.2, the results from the first experiment are not successful. Both training and testing accuracy is low. With Star Size 15, the training accuracy is only 74.2% and the testing accuracy drops 11.8% to 62.4%. Although we could apply the Regression Pruning Algorithm to reduce the overfitting problem, the low training accuracy might not be improved in this case. There are two possibilities that might have caused the problem. Either the representation of router line traffic is not fit in machine learning tools or there may exist an upper bound of the classification accuracy in this domain.

To investigate the first possibility, we analyzed the line traffic statistics in the training data set. We found out that there are no traffic on line #2 to #9 throughout the whole day. Line #1 and line #12 to #15 always have traffic during the day. Line #10 and line #11 have low and burst traffic over a day. Traffic on line #12 to #15 have similar variations during a day and so do line #10 and line #11. As can be seen in Figure 6.3, among those lines which have traffic, line #1 is connected to a LAN and all the others are connected to Routers. So there might have different traffic variations for these two kinds of connections. Figure 6.5 shows the traffic variations on line #1 during a day. For brevity, only the first-half of a day traffic is drawn. The second-half has the same distribution. Figure 6.6 shows line #10 traffic variations over a day which is same as the variations on line #11. Similarly, Line #12's traffic variations over a day

2 OF 2

PM-1 3 1/2" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

is drawn in Figure 6.7 to represent the traffic variations on line #12 to #15.

Figure 6.5 Line #1 Traffic Variations Over 12 Hours

As can be seen from those figures, the traffic on the line which connects to a LAN does not apparently vary with the time of day. Line #1 is connected to the LAN e0.bCAR, so its traffic does not apparently vary with the time of day. However traffic patterns on lines connected to other routers apparently vary with the time of day. As seen in Figure 6.7, line #12's traffic is low at night except during the time of large file transfers (the large file transfer occurred around 4:00am) and high during the day time except during the lunch time. But the traffic on line #10 and line #11 is a little special. Although the traffic volume varies with the time of day, the traffic fluctuates significantly over the whole day. As can be seen from Figure 6.6, lots of traffic falls on the X-axis which represents zero traffic volume. The absolute traffic volume is also lower than the volumes on other lines. Thus, when we collected the traffic at one-minute interval, we got fluctuating traffic on line #10 and line #11. Trained with these fluctuating traffic data, CN2 was difficult to catch the real traffic patterns. It is one of the rea-

Figure 6.6 Line #10 Traffic Variations Over One Day

sons why the first experiment was not successful.

Key observations from the traffic statistics: Lines connected to LANs and lines connected to other Routers have different traffic variations during a day. The traffic on lines connected to other routers apparently vary with the time of day. Some of the lines connected to other routers have fluctuating traffic and the traffic is also low and for most of the time the traffic volume is zero. Lines connected to LANs have high traffic volume but the traffic does not apparently vary with the time of day. The lines which have fluctuating traffic make the generation of traffic patterns less predictive. So we should try to minimize the influence of the fluctuating traffic to the rule generation of the learning system. We show our approach to collect better traffic information in the next two experiments.

Then we investigated the second possibility which is whether there exists an upper bound for the task of classification in this domain. We evaluated the rules generated from the

Figure 6.7 Line #12 Traffic Variations Over One Day

training examples against both the training set and the testing set and show the results in Table 6.3 and Table 6.4 below.

Table 6.3: Rule Evaluation on the Training Set

Actual Classes	Predicted Classes																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	38	1	1		8	3	5	1													1		1	1
2	4	41	1		7	1	2	1													2		1	0
3	4	3	31	3	5	5		2	1									1			1	1		
4		2	1	46	4		1		2				4											
5	3	1	1		47	4	3	1																
6	4	2			11	33	6	4																
7	2	4			9	5	34	5																1
8	6	2		4	5	1	4	36											1		1			

Table 6.3: Rule Evaluation on the Training Set

Actual Classes	Predicted Classes																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
9				4					48		1		4			1	1		1						
10										45	5				5	3	1	1							
11											54	3					2	1							
12				1					1	1	6	46	1		2	2									
13				8					2				46	1	2				1						
14										3	4	3	1	43	2	1	2	1							
15										2	1	3		1	50	1	1	1							
16										3	1	1		1	4	49	1								
17										1	2	2	1	1	3		50								
18								1			4			1	3	3	1	41	2	2	1	1			
19				1					1										50	5	2	1			
20													1						5	49	2	1	1	1	
21									1										3	1	48	5	2		
22																			2	1	1	49	6	1	
23																			1	1	2	1	54	1	
24	2	1		1	1	2	1												2	1		1	7	41	

Table 6.4: Rule Evaluation on the Testing Set[illegible]

Table 6.4: Rule Evaluation on the Testing Set

Actual Classes	Predicted Classes																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
5	7	3	1		35	5	5	3												1				
6	4	3			10	30	8	4														1		
7	5	5			9	10	27	3																1
8	11	2		3	5	4	7	26					1						1					
9	2			7				1	41		2	1	5						1					
10	2								1	35	5	2		1	5	3	5	1						
11	3									3	50	1					2	1						
12	2								1	1	8	40	2	2	2	2								
13	4			8					4			1	38	1	3				1					
14	1									3	3	3	2	41	2	1	3	1						
15	3								1	4	3	1		2	39	3	1	3						
16	4								1	2	1	1		1	4	41	4	1						
17	3									1	4	2		1	4	2	43							
18	2							1		1	3		1	2	3	2	4	35	2	1	1	1		
19	2								1				1						2	47	5	2		
20	2												1							7	41	4	2	1
21	4								1											5	4	39	5	2
22	1		1																2	1	4	40	7	4
23	2				1														2	1	2	3	48	1
24	5	1		1		3	1	1											2	1	1	1	8	35

As seen in Table 6.3, the rules were evaluated against the training set itself. The header contains the classes predicted by these rules. The first column contains the actual classes. Each row shows that how many examples of the actual class were predicted by these rules and how many examples were misclassified into other classes. Similarly, Table 6.4 shows the evaluation results on the testing set. Figure 6.8 and Figure 6.9 also show the comparisons between

the rule accuracy for each of the 24 classes against the training set and the testing set respectively.

Figure 6.8 Rule Accuracy for Different Classes: Evaluation on the Training Set

As can be seen from the results, there exist similarities among several classes, i.e., rules (traffic patterns) are overlapping among these classes. The overlapping is a general problem. However, it shows more frequently among small classes (classes under Eight). The rule accuracy for those classes is relatively lower as can be seen in Figure 6.8 and 6.9. The reason seems to be that the smaller classes represent the traffic after midnight and early in the morning. During that time, the network traffic tends to be low and stable. So these traffic patterns are similar to each other and overlapping. They are difficult to be distinguished by the learning algorithm. As a result, the prediction accuracy for the traffic patterns in small classes are lower.

Since this problem is determined by the nature of the network, there must exist an upper bound for the overall rule accuracy that any learning system can achieve. One possible

Figure 6.9 Rule Accuracy for Different Classes: Evaluation on the Testing Set

explanation for this upper bound problem is that the traffic data are multi-dimensional and non-linearly separable. But the machine learning algorithms are linear classifiers. They represent classifications of this domain in hyperplanes. Examples within or near a hyperplane are classified into one class, and different hyperplanes may have different classes. In the traffic example, the hyperplanes for different classes are overlapping. Expanding one hyperplane will cover more examples in the predicted class but inevitably examples in other classes will be misclassified. Using a number of smaller hyperplanes jointed together to describe the pattern is better than using a larger hyperplane but the number of rules will get bigger and the rules tend to be specific and overfit the training examples. So there is an limit to the accuracy which machine learning algorithms can achieve no only in this network traffic domain but also in other domains. In the following experiments we show our approach to improve this performance limitation.

6.3.2.2 The Second Experiment

The objective of the second experiment is to improve CN2's performance by finding a better way to collect traffic data.

In the first experiment, the line traffic of router bCARc2 was collected at an interval of one minute. Figure 6.6 showed that the traffic on line #10 and #11 fluctuates significantly. So we tried to reduce the influence of traffic fluctuation in this experiment by applying different traffic collection functions.

The definitions of the different traffic collection functions employed in this thesis are as follows:

1-minute-total: The 1-minute-total traffic on each links (lines) of a particular router is defined as the total traffic sent out by the router in last one minute.

10-minute-mean: The 10-minute-mean traffic on each links (lines) of a particular router is defined as the mean traffic sent out by the router in last 10 minutes. It is calculated by dividing the total traffic in last 10 minutes by ten.

15-minute-total: The 15-minute-total traffic on each links (lines) of a particular router is defined as the total traffic sent out by the router in last 15 minutes.

Firstly, we tried to increase the traffic collection time interval from one minute to fifteen minutes. The Traffic-collector was modified to poll all lines every 15 minutes for a day. Each time the line traffic data were stored in the traffic data file. Two such files were generated, one for each independent day. We did two sets of experiments to test the results. In the first set, traffic set Day-1 was used as the training set and Day-2 was used as the testing set. In the second set, Day-2 was used as the training set and Day-1 as the testing set. We did the second set of experiments to verify our results in the first set. For each set of the experiments the CN2 Unordered learning algorithm was used and the parameters

were set as same as those in the first experiment, i.e., Threshold=0; Star Size=15; Laplace Error Estimate. The results are shown in Table 6.5 below.

Table 6.5: Results of Applying the “15-minute-total” Traffic Collection Function

Set of Experiments	Training Set	Testing Set	Training Accuracy	Testing Accuracy	Accuracy Drops
1	Day-1	Day-2	91.7%	33.3%	46.9%
2	Day-2	Day-1	89.6%	33.3%	55.3%

As seen in the table, both sets of experiments got similar results. The training accuracy is high and the testing accuracy is very low. The accuracy drop is then very high. It shows that the “15-minute-total” traffic collection function can not moderate the influence of router traffic fluctuation. The training accuracy is high because the number of examples is very small when polling the simulator every 15 minutes instead of every 1 minute. CN2 may have overfit the small training set.

Since the 15-minute-total representation was not successful, we used another approach. We modified the Traffic-collector to poll bCARc2 every 10 minutes. Instead of using the traffic total, the Traffic-collector calculated the mean traffic in the 10-minute time interval and stored the mean traffic in the traffic data file. Similarly, two traffic data sets were generated. Each data set contained 144 examples with 15 attributes. Two sets of experiments were conducted under the same conditions in last experiment. The results are shown in Table 6.6 below.

Table 6.6: Results of Applying the “10-minute-mean” Traffic Collection Function

Set of Experiments	Training Set	Testing Set	Training Accuracy	Testing Accuracy	Accuracy Drops
1	Day-1	Day-2	93.8%	82.6%	11.2%
2	Day-2	Day-1	93.1%	84.0%	9.1%

As seen in the table, the “10-minute-mean” traffic collection function is better

than the “1-minute-interval” and “15-minute-total” traffic collection functions. The training accuracy is high and the rules also perform well on new examples in the testing set. Table 6.7 compares the results of applying different traffic collection functions. The accuracy used corresponds to the accuracy of using Day-1 as the training set and Day-2 as the testing set.

Table 6.7: Comparison of Rule Accuracy under Different Traffic Collection Functions

Collection Function	Training Accuracy	Testing Accuracy	Accuracy Drops
1-minute-interval	74.2%	62.4%	11.8%
15-minute-total	91.7%	33.3%	46.9%
10-minute-mean	93.8%	82.6%	11.2%

Key observations from the second experiment: The “10-minute-mean” is a good traffic collection function. Traffic patterns generated from the traffic data collected by using this function are general and reliable in the sense that both the training accuracy and the testing accuracy are high and the accuracy drop is relatively small. When the time interval for the traffic collection gets smaller, the collected traffic will fluctuate significantly and so it is hard for CN2 to generate reliable rules from the fluctuating traffic data. It is better to use a relatively longer time interval to collect traffic and also it is better to collect the mean traffic to smooth the line traffic fluctuation and reduce the influence of the traffic fluctuation on the classifications of normal traffic patterns. Figure 6.10 - 6.12 show the variations of the traffic on line #1, line #10 and line #12 respectively where the traffic was collected using the 10-minute-mean function.

Compared with previous figures (Figure 6.5-6.7), the line traffic variations are changed after using the 10-minute-mean collection function. Especially the traffic fluctuation on line #10 has been efficiently smoothed.

Figure 6.10 Line #1 Traffic Variations Using “10-minute-mean” Collection Function

6.3.2.3 The Third Experiment

Although we had good results in the second experiment, the overfitting problem is still existing. As seen in Table 6.6, the Accuracy Drop when applying the “10-minute-mean” traffic collection function is still around 10%.

In this experiment, we applied the Regression Pruning Algorithm developed in Chapter 5 to reduce the overfitting problem. The experiment was carried out in the following way: Day-1 traffic was used as the training set and the training examples were pruned with the help of Day-2, the testing set. Table 6.8 shows the results of applying the RPA pruning method

Figure 6.11 Line #10 Traffic Variations Using "10-minute-mean" Collection Function

Table 6.8: Comparison of CN2's Performance on the "10-minute-mean" Traffic Data

CN2 Status	Training Accuracy (%)	Testing Accuracy (%)	Accuracy Drop (%)	Number of Rules
Without RPA	93.8	82.6	11.2	42
With RPA	95.1	87.5	7.6	41

Figure 6.13 and 6.14 show the prediction accuracy on 2-D plots.

The results show that after applying RPA, both the training accuracy and the testing accuracy increase. However, the testing accuracy increased more than the training accuracy. So the accuracy drop decreased. The number of rules also slightly reduced. The Regression Pruning Algorithm improved CN2's performance in this domain.

Figure 6.12 Line #12 Traffic Variations Using "10-minute-mean" Collection Function

6.3.2.4 The Fourth Experiment

In the first experiment, we investigated the limitations of applying machine learning techniques in the multi-dimensional non-linearly separable domain. As remarked before, there exists an upper bound for the performance of any learning systems. In the third experiment, even though we applied the RPA pruning method, the testing accuracy still dropped 7.6% to 87.5%. The objective of this experiment is thus to investigate the way to solve the traffic pattern overlapping problem so as to overcome the upper bound limitation.

As we have studied before, most line traffic is low and stable from 1:00am to 8:00am except during the large file transfer period which occurs at 4:00am. The traffic pat-

Figure 6.13 Applying RPA on “10-minute-mean” Traffic Data: Training Accuracy

terms of classes One, Two, Three, Five, Six, Seven, and Eight overlap, which leads to the low prediction accuracy among all these classes. In this experiment, we combined these overlapped classes into one super-class. Then in each data set, we had 18 classes instead of the original 24 classes. We repeated the third experiment with the same data except combined classes. The results and the comparison between the fourth and the third experiments are shown in Table 6.9 below.

Table 6.9: Comparison of Results in The Third and Fourth Experiment without RPA

Experiment Number	Number of Classes	Training Set	Testing Set	Training Accuracy (%)	Testing Accuracy (%)	Accuracy Drops (%)
3rd	24	Day-1	Day-2	93.8	82.6	11.2
		Day-2	Day-1	93.1	84.0	9.1

Figure 6.14 Applying RPA on "10-minute-mean" Traffic Data: Testing Accuracy

Table 6.9: Comparison of Results in The Third and Fourth Experiment without RPA

Experiment Number	Number of Classes	Training Set	Testing Set	Training Accuracy (%)	Testing Accuracy (%)	Accuracy Drops (%)
4th	18	Day-1	Day-2	97.9	92.4	5.5
		Day-2	Day-1	96.5	91.0	5.5

The results show that the performance improved after we combined the overlapping classes. Both the training and the testing accuracy increased and the accuracy-drop was very low. We further applied the RPA on these new data sets. Figure 6.15 and 6.16 show the results from applying RPA on the first set of experiment which used Day-1 as the training set and Day-2 as the pruning set. Table 6.10 also compares the effect of applying RPA in the third and fourth experiments.

Figure 6.15 Applying RPA in the Fourth Experiment: Training Accuracy

Figure 6.16 Applying RPA in the Fourth Experiment: Testing Accuracy

Table 6.10: Effect of Applying RPA in the Third and Fourth Experiment

Experiment Sequence	CN2 Status	Training Accuracy (%)	Testing Accuracy (%)	Accuracy Drops (%)
3rd	without RPA	93.8	82.6	11.2
	with RPA	95.1	87.5	7.6
4th	without RPA	97.9	92.4	5.5
	with RPA	97.2	94.4	2.8

The results show that with the RPA pruning, the difference between the training accuracy and the testing accuracy became even smaller. Since the accuracy was already very high after we combined the overlapping classes, the accuracy did not increase very much after applying RPA and the training accuracy dropped 0.7%.

Key observations from the fourth experiment: Combining the overlapping classes improves CN2's performance despite of the upper bound limitation.

6.4 Summary

In this chapter, we showed our application of machine learning techniques in network fault diagnosis. The network fault diagnosis requires normal network model be built, against which abnormal network behavior can be compared. Our application showed that the models or rules of normal network behavior can be acquired through inductive learning.

Specifically, in this thesis, a general system model was developed for the applications of machine learning techniques in the network domain. This model directed our research to apply the CN2 rule induction program to help establish normal traffic patterns for routers in the BNR WAN to account for the time of day network traffic variations. A network traffic collection func-

tion which collects the mean traffic in 10-minute time interval was developed and proved efficient in providing CN2 with more accurate traffic data for rule classifications. The Regression Pruning Algorithm developed in Chapter 5 was also applied in this application to test its prune ability.

In this chapter we also discussed the upper bound limitations of applying machine learning tools to the multi-dimensional non-linearly separable domain. We provided a partial solution to this problem by combining similar traffic patterns into a super-class and got very good results. The performance improvement by using this approach shows that we should pay more attentions to investigate the domain characteristics in our future research work on machine learning.

Chapter 7

Conclusions, Contributions and Future Research

7.1 Conclusions

This thesis addresses the three objectives described in Chapter 1:

- 1) Various machine learning techniques and their applications in network management are reviewed in Chapter 3. The BNR network and the Pegasus fault diagnostic project are briefly overviewed in Chapter 2.
- 2) The Overfitting problem encountered in most of inductive learning algorithms is identified and investigated in Chapter 4. A new pruning technique -- the Regression Pruning Algorithm is developed and tested in three different task domains in Chapter 5.
- 3) This thesis also applies machine learning techniques in network fault diagnosis. In Chapter 6, the CN2 rule induction program is used to help establish normal router traffic patterns which can be used in the development of rule-based expert systems for network fault diagnosis. The Regression Pruning Algorithm is also tested in this domain.

7.2 Contributions

This thesis has the following contributions:

- 1) In machine learning area: A new heuristic pruning method -- the Regression Pruning Algorithm is researched and developed in this thesis. It is also applied in three machine learning task domains including the artificial domain, the LED display domain, and the Soybean disease domain. The Regression Pruning Algorithm proves efficient in reducing the overfitting problem compared with other pruning methods.
- 2) In machine learning application area: A general system model for applying machine learning techniques in network fault diagnosis is developed. This thesis shows that machine learning techniques can help ease the knowledge acquisition bottleneck in the development of rule-based expert systems for network fault diagnosis. Specifically, the CN2 rule induction program is applied to generate useful classification rules from large volumes of network traffic data, which is a difficult work for human operators to accomplish.
- 3) In network fault diagnosis area: CN2 with the new pruning method is applied to the establishment of normal router traffic patterns to account for any time of day traffic variations. The effective traffic collection function "10-minute-mean" is developed by analyzing the line traffic variations on the router bCARc2. Research is also conducted on how to present the network characteristics to machine learning tools to overcome the limitations by their linear classification nature.

7.3 Future Research

There are a number of future research directives that can be considered by people who are interested in extending this thesis's work:

- 1) The traffic patterns generated from the CN2 rule induction program can be used to build an expert system knowledge source which in turn can monitor the network at different time of day to report any router traffic abnormalities.
- 2) In this thesis, only normal router traffic patterns are established to account for any time of day traffic variations. One of the possible future extensions is to establish abnormal router traffic patterns for each of the possible faults related to a router.
- 3) Normal network traffic conditions are defined as an assumption in this thesis. The actual network however is highly dynamic. So it is necessary to establish different normal traffic patterns to account for any network load variations.
- 4) The "10-minute-mean" is a good traffic collection function in this thesis. However, other traffic collection functions such as "Running Window Smoothing" technique (Running windows average traffic rates over a specified number of intervals) can be employed and compared with this thesis's approach.
- 5) In this thesis, we assigned 24 classes to represent each one-hour period of a day. There are other possible ways for the class assignment according to the need of specific applications.
- 6) The stability of RPA in converging on the optimal training set by use of the pruning set is not guaranteed theoretically. Further examination of this and the relationship between the rate of convergence and rate of reduction in overfitting is appropriate.

References

- [Bisio91] "Incremental Knowledge Base Construction through Integrated Deductive and Inductive Learning", R. Bisio, et. al., CSELT Technical Reports, Vol. 19, No. 5, Nov., 1991.
- [Bisio92] "Using a Machine Learning Tool in Diagnosis of Network Overload", R. Bisio, R. Gemello, and E. Montariolo, ICC '92, Chicago, June, 1992.
- [BNR90a] "BNR Local Area Network Environment: An Introduction for the Network User", BNR Information Technology, Ottawa, Canada., July 1990.
- [BNR90b] "MAGNet: Network Management Tools, Release 3.0", Bell Northern Research, Ottawa, Canada, June. 1990.
- [BNR92] "Classical Faults in BNR Networks (Oral Presentation by Bernie Murphy)", Information Technology Division, Bell Northern Research, Ottawa, Canada, September, 1992.
- [Boswell] "Manual for CN2 Version 6.1", R. Boswell, Reference: TI/P2154/RAB/4/1.5, The Turing Institute Limited, 1990.
- [Breiman84] "Classification and Regression Trees", L. Breiman et. al., Belmont, California: Wadsworth International Group, 1984.
- [Carbonell87] "Learning Machine", J. Carbonell and P. Langley, Encyclopedia of Artificial Intelligence, Volume 1, pp464-488, John Wiley & Sons, Inc., 1987.
- [Cestnik87] "ASSISTANT 86: A knowledge-elicitation tool for sophisticated users", B. Cestnik, I. Kononenko, & I. Bratko, Proceedings of the Second European Working Session on Learning, pp31-45, Sigma Press, 1987.

- [Cestnik91] "On Estimating Probabilities in Tree Pruning", B. Cestnik & I. Bratko. Machine Learning, EWSL-91, Berlin: Springer-Verlag, 1991.
- [Clark89] "The CN2 Induction Algorithm", P. Clark and T. Niblett, Machine Learning, 3(4): pp261-283, 1989.
- [Clark91] "Rule Induction with CN2: Some Recent Improvements", P. Clark and R. Boswell, In EWSL '91. Machine Learning: Proceedings of the European Working Session on Learning, pp151-163, 1991.
- [Dawes91] "Network Diagnosis by Reasoning in Uncertain Nested Evidence Spaces", N. W. Dawes, J. Altoft, B. Pagurek. Carleton University Technical Report, SCE-91-26, 1991.
- [Dawes92] "DRAGON: A Fast Network Simulator: Version 3.4", N. W. Dawes, Carleton University Technical Report, SCE-92-14, June, 1992.
- [Dussault92] "A Diagnostic Expert System for Wide Area Networks", R. Dussault, Master of Engineering Thesis, Carleton University, March, 1992.
- [Farrell93] "Detection of Abnormal Router Behavior in a Wide Area Network", A. D. Farrell, Masters of Engineering Thesis, Dept. of Systems and Computer Engineering, Carleton University, 1993.
- [Feng92] "Comparison of Classification Algorithms in Machine Learning. Statistics and Neural Networks (draft)", C. Feng, et. al., 1992.
- [Fisher89] "Concept Simplification and Prediction Accuracy", D. H. Fisher and J. C. Schlimmer, 1989.
- [Gams91] "A Schema for Using Multiple Knowledge", M. Gams, M. bohanec, and B. Cestnik, Submitted to IJCAI-91, 1991.

- [Goodman89] "Real Time Autonomous Expert Systems in Network Management", R. M. Goodman, et. al., Integrated Network Management, I, pp599-624, 1989.
- [Goodman91] "Automated Knowledge Acquisition from Network Management Database", R. M. Goodman and H. Latin, Integrated Network Management II, pp541-549, 1991.
- [Goodman92] "An Information Theoretic Approach to Rule Induction from Database", R. M. Goodman and P. Smyth, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 4, August 1992, pp301-316.
- [Iqneibi92] "A Blackboard Architecture to Support Network Fault Diagnosis", S. Iqneibi, Master of Engineering Thesis, Department of Systems and Computer Engineering, Carleton University, 1992.
- [Kalab93] "Operations Research Techniques Applied to Network Fault Management", P. Kalab, Master of Engineering Thesis, Dept. of Systems and Computer Engineering, Carleton University, 1993.
- [Kalbfleish79] "Probability and Statistical Inference Vol. 2", J. Kalbfleish, New York: Springer-Verlag, 1979.
- [Michalski80] "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis", R. S. Michalski and R. L. Chilausky, International Journal of Policy Analysis and Information Systems, Vol. 4, No. 2, 1980.
- [Michalski83] "Incremental generation of VL1 hypotheses: The underlying methodology and the description of the program AQ11", R. S. Michalski, & J. Larson, Technical Report ISG 83-5, Computer Science Department, University of Illinois as

Urbana, 1983.

- [Michalski86] "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains", R. S. Michalski, et. al., Proceedings of the Fifth National Conference on Artificial Intelligence, pp1041-1045, Morgan Kaufmann, 1987.
- [Mingers87] "Expert Systems -- Rule Induction with Statistical Data", J. Mingers, Journal of the Operational Research Society, 38, pp39-47, 1987.
- [Mingers89] "An Empirical Comparison of Pruning Methods for Decision Tree Induction", J. Mingers, Machine Learning, 4, pp227-243, 1989.
- [Niblett86] "Learning Decision Rules in Noisy Domain", T. Niblett & I. Bratko, Research and Development in Expert Systems III, Cambridge: Cambridge University Press, 1986.
- [Niblett87] "Constructing Decision Trees in Noisy Domains", T. Niblett, Proceedings of the Second European Working Session on Learning, pp67-78, Bled, Yugoslavia: Sigma Press, 1987.
- [Pagurek91] "A Multiple Paradigm Diagnostic System For Wide Area Communication Networks", B. Pagurek, N. W. Dawes, A. R. Kaye, Carleton University Technical Report, SCE-91-37, 1991.
- [Patel89] "Integrating Network Management and Artificial Intelligence", A. Patel et. al., pp647-660, Integrated Network Management I, North-Holland, 1989.
- [Quinlan86a] "Induction of Decision Trees", J. R. Quinlan, Machine Learning, 1, pp81-106, 1986.
- [Quinlan86b] "The Effect of Noise on Concept Learning", J. R. Quinlan, Machine Learning:

An Artificial Intelligence Approach, Vol. 2, pp149-166, 1986.

- [Quinlan87a] "Simplifying Decision Trees", J. R. Quinlan, International Journal of Man-Machine Studies, 27, pp221-234, 1987.
- [Quinlan87b] "Generating Production Rules from Examples", J. R. Quinlan, Proc. Tenth Int. Joint Conf. on Artificial Intelligence, pp304-307, 1987.
- [Schaffer93] "Overfitting Avoidance as Bias", C. Schaffer, Machine Learning, 10, pp153-178, 1993.
- [Tan88] "Using Weighted Network to Represent Classification Knowledge in Noisy Domains", M. Tan & L. Eshelman, Proceedings of the 5th International Conference on Machine Learning, pp121-134, Ann Arbor, Michigan: Morgan Kaufman.
- [Viens92] "A Neural Network Approach to Detect Traffic Anomalies in a Communication Network", F. Viens, Master of Engineering Thesis, Dept. of Systems and Computer Engineering, Carleton University, 1992.
- [Yuan92] "PegasusBase - A Run Time Schema Extendible System", B. Yuan, Master of Engineering Thesis, Department of Systems and Computer Engineering, Carleton University, 1992.

Appendix I

An example data set created from the artificial domain which is described in Chapter 5.4.

Table 7: An example data set created from the artificial domain

Example	Attribute: X	Attribute: Y	Class
1	7.0	9.0	A
2	10.0	8.5	A
3	15.0	9.0	A
4	20.0	8.5	A
5	12.5	8.0	A
6	8.0	7.0	A
7	17.0	7.5	A
8	5.0	6.0	A
9	10.0	6.0	A
10	15.0	6.0	A
11	20.0	6.0	A
12	15.0	4.0	A
13	7.0	4.0	A
14	18.0	4.0	A
15	10.0	4.0	A
16	12.5	2.0	A
17	13.0	5.0	A
18	12.0	13.0	B
19	10.0	10.0	B
20	17.0	11.0	B

Table 7: An example data set created from the artificial domain

Example	Attribute: X	Attribute: Y	Class
21	20.0	10.0	B
22	12.5	11	B
23	5.0	10.5	B
24	7.5	11.0	B
25	10.0	12.0	B
26	15.0	12.0	B
27	20.0	12.0	B
28	15.0	14.0	B
29	10.0	14.0	B
30	7.0	14.0	B
31	17.5	14.5	B
32	12.0	15.0	B
33	20.0	16.0	B
34	5.0	16.0	B
35	14.0	10.0	B

Appendix II

The unordered rule lists generated from data set Set_1 (in Chapter 5.4) under different CN2 significance test levels.

```

IF      5.50 < Y < 9.50
THEN   class = A    [11 0]

IF      Y > 9.50
THEN   class = B    [0 18]

(default) class = B    [17 19]

```

Figure 7 The unordered rule list generated from Set_1 with Threshold 10.00

```

IF      5.50 < Y < 9.50
THEN   class = A    [11 0]

IF      Y > 9.50
THEN   class = B    [0 18]

(default) class = B    [17 19]

```

Figure 8 The unordered rule list generated from Set_1 with Threshold 15.00

IF Y > 9.50
THEN class = B [0 18]

(default) class = B [17 19]

Figure 9 The unordered rule list generated from Set_1 with Threshold 20.00

(default) class = B [17 19]

Figure 10 The unordered rule list generated from Set_1 with Threshold 25.00

END

0 8-1 1-94

FIN