

Spring 2008

# K-Means+ID3 and dependence tree methods for supervised anomaly detection

Kiran S. Balagani  
*Louisiana Tech University*

Follow this and additional works at: <https://digitalcommons.latech.edu/dissertations>



Part of the [Artificial Intelligence and Robotics Commons](#)

---

## Recommended Citation

Balagani, Kiran S., "" (2008). *Dissertation*. 508.  
<https://digitalcommons.latech.edu/dissertations/508>

This Dissertation is brought to you for free and open access by the Graduate School at Louisiana Tech Digital Commons. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of Louisiana Tech Digital Commons. For more information, please contact [digitalcommons@latech.edu](mailto:digitalcommons@latech.edu).

**K-MEANS+ID3 AND DEPENDENCE TREE**  
**METHODS FOR SUPERVISED ANOMALY DETECTION**

by

Kiran S. Balagani, M. S.

A Dissertation Presented in Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE  
LOUISIANA TECH UNIVERSITY

*March 2008*

UMI Number: 3298935

### INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

**UMI**<sup>®</sup>

---

UMI Microform 3298935

Copyright 2008 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company  
300 North Zeeb Road  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

23 January 2008

Date

We hereby recommend that the dissertation prepared under our supervision  
by Kiran SVRV Balagani  
entitled K-Means+ID3 and Dependence Tree Methods for Anomaly  
Detection

be accepted in partial fulfillment of the requirements for the Degree of  
Doctor of Philosophy in CAM

Vis Venands Ploha

Supervisor of Dissertation Research

Ulizhong Shi

Head of Department

Department

Recommendation concurred in:

Ulizhong Shi

Ahri Ban

Jenko Kanno  
Dale E. Truitt

Advisory Committee

Approved:

Bala Subrahmanyan

Director of Graduate Studies

Approved:

William M. Conarty

Dean of the Graduate School

Tom Wynn

Dean of the College

## ABSTRACT

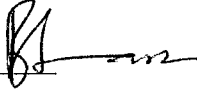
In this dissertation, we present two novel methods for supervised anomaly detection. The first method “K-Means+ID3” performs supervised anomaly detection by partitioning the training data instances into  $k$  clusters using Euclidean distance similarity. Then, on each cluster representing a density region of normal or anomaly instances, an ID3 decision tree is built. The ID3 decision tree on each cluster refines the decision boundaries by learning the subgroups within a cluster. To obtain a final decision on detection, the k-Means and ID3 decision trees are combined using two rules: (1) the nearest neighbor rule and (2) the nearest consensus rule. The performance of the K-Means+ID3 is demonstrated over three data sets: (1) network anomaly data, (2) Duffing equation data, and (3) mechanical system data, which contain measurements drawn from three distinct application domains of computer networks, an electronic circuit implementing a forced Duffing equation, and a mechanical mass beam system subjected to fatigue stress, respectively. Results show that the detection accuracy of the K-Means+ID3 method is as high as 96.24 percent on network anomaly data; the total accuracy is as high as 80.01 percent on mechanical system data; and 79.9 percent on Duffing equation data. Further, the performance of K-Means+ID3 is compared with individual k-Means and ID3 methods implemented for anomaly detection.

The second method “dependence tree based anomaly detection” performs supervised anomaly detection using the Bayes classification rule. The class conditional probability densities in the Bayes classification rule are approximated by dependence trees, which represent second-order product approximations of probability densities. We derive the theoretical relationship between dependence tree classification error and Bayes error rate and show that the dependence tree approximation minimizes an upper bound on the Bayes error rate. To improve the classification performance of dependence tree based anomaly detection, we use supervised and unsupervised Maximum Relevance Minimum Redundancy (MRMR) feature selection method to select a set of features that optimally characterize class information. We derive the theoretical relationship between the Bayes error rate and the MRMR feature selection criterion and show that MRMR feature selection criterion minimizes an upper bound on the Bayes error rate. The performance of the dependence tree based anomaly detection method is demonstrated on the benchmark KDD Cup 1999 intrusion detection data set. Results show that the detection accuracies of the dependence tree based anomaly detection method are as high as 99.76 percent in detecting normal traffic, 93.88 percent in detecting denial-of-service attacks, 94.88 percent in detecting probing attacks, 86.40 percent in detecting user-to-root attacks, and 24.44 percent in detecting remote-to-login attacks. Further, the performance of dependence tree based anomaly detection method is compared with the performance of naïve Bayes and ID3 decision tree methods as well as with the performance of two anomaly detection methods reported in recent literature.

## APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author KIRAN S. BALAGANI 

Date 02/23/2008

## **DEDICATION**

To my beloved grandparents Sreemathi Sarojini Konakalla, late Sri Narasimha Rao Konakalla, Sreemathi Anasuya Devi Balagani, and Sri Venkateshwarlu Balagani.

To my dearest grandmothers Sreemathi Subhadra Devi Konakalla and Sreemathi Eshwari Mekapothula.



## TABLE OF CONTENTS

ABSTRACT.....	iii
DEDICATION.....	vi
LIST OF TABLES.....	ix
LIST OF FIGURES.....	x
ACKNOWLEDGMENTS.....	xiii
CHAPTER 1: INTRODUCTION.....	1
1.1 K-Means+ID3 Anomaly Detection Method.....	3
1.2 Dependence Tree Based Anomaly Detection Method.....	5
1.3 Contributions of the Dissertation.....	6
CHAPTER 2: RELATED RESEARCH.....	9
2.1 Anomaly Detection Research in Computer Networks.....	9
2.2 Anomaly Detection Research in Other Domains.....	13
CHAPTER 3: THE K-MEANS+ID3 ANOMALY DETECTION METHOD.....	16
3.1 Anomaly Detection with Individual K-Means Clustering Method.....	19
3.2 Anomaly Detection with ID3 Decision Tree Learning.....	20
CHAPTER 4: TRAINING AND TESTING THE K-MEANS+ID3 ANOMALY DETECTION METHOD.....	22
4.1 The Candidate Selection Phase.....	23
4.2 The Candidate Combination Phase.....	25
4.2.1 The Nearest Consensus Rule.....	26
4.2.2 The Nearest Neighbor Rule.....	26
CHAPTER 5: DATASETS FOR TESTING THE K-MEANS+ID3 METHOD.....	27
5.1 Network Anomaly Data.....	28
5.2 Duffing Equation Data.....	29
5.3 Mechanical Systems Data.....	30
CHAPTER 6: RESULTS OF K-MEANS+ID3 ANOMALY DETECTION METHOD.....	32
6.1 Results on the NAD-1998 Dataset.....	33

6.2	Results on the NAD-1999 Dataset.....	35
6.3	Results on the NAD-2000 Dataset.....	36
6.4	Results on the Duffing Equation Dataset.....	38
6.5	Results on the Mechanical Systems Dataset.....	40
CHAPTER 7: ANOMALY DETECTION USING		
	DEPENDENCE TREES .....	42
7.1	Problem Formulation .....	43
7.2	Dependence Trees.....	44
7.3	Steps for Building Dependence Trees.....	45
7.4	Optimality of Dependence Tree Approximation .....	46
7.5	Estimating Mutual Information.....	47
CHAPTER 8: RELATIONSHIP BETWEEN DEPENDENCE TREE		
	CLASSIFICATION ERROR AND BAYES ERROR RATE .....	49
8.1	Derivation Relating Bayes Error Rate ToDependence Tree	
	Classification Error .....	53
CHAPTER 9: MAXIMUM RELEVANCE MINIMUM REDUNDANCY		
	FEATURE SELECTION.....	56
9.1	Maximum Relevance Feature Selection .....	57
9.2	Maximum Relevance Minimum Redundancy (MRMR) .....	58
9.3	Relationship Between MRMR Feature Selection and Bayes Error Rate.....	59
CHAPTER 10: THE KDD CUP 1999 INTRUSION DETECTION DATASET.....		
10.1	Features in KDD Cup 1999 Dataset.....	69
CHAPTER 11: EXPERIMENTS AND RESULTS.....		
11.1	Results of MRMR Feature Selection on KDD Cup 1999 Dataset.....	73
11.2	Dependence Tree Results.....	74
	11.2.1 Dependence Tree Results with Supervised MRMR Selection .....	75
	11.2.2 Dependence Tree Results with Unsupervised MRMR Selection .....	79
11.3	Comparison with Naïve Bayes and ID3 Anomaly Detection Methods .....	83
	11.3.1 Results with Supervised MRMR Feature Selection.....	84
	11.3.2 Results with Unsupervised MRMR Feature Selection .....	85
11.4	Comparison with Other Studies on KDD Cup 1999 Datasets .....	87
CHAPTER 12: CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS .....		
		90
REFERENCES .....		
		94

## LIST OF TABLES

Table 5.1	Characteristics of the NAD, DED and MSD datasets used in the anomaly detection experiments.....	28
Table 10.1	Basic features of KDD Cup 1999 dataset. ....	69
Table 10.2	Content features of KDD Cup 1999 dataset. ....	69
Table 10.3	Temporal features of KDD Cup 1999 dataset.....	70
Table 10.4	Distribution of normal and attack connections in the KDD Cup 1999 dataset .....	71
Table 11.1	The first eight features in KDD Cup 1999 datasets selected through MRMR supervised feature selection. ....	73
Table 11.2	The first eight features in KDD Cup 1999 datasets selected through MRMR unsupervised feature selection. ....	74
Table 11.3	Results of Dependence Tree based anomaly detection (DTree), Naïve Bayes (NB) anomaly detection, and ID3 anomaly detection on KDD Cup 1999 features selected by the supervised MRMR feature selection method.....	84
Table 11.4	Results of Dependence Tree based anomaly detection (DTree), Naïve Bayes (NB) anomaly detection, and ID3 anomaly detection on KDD Cup 1999 features selected by the unsupervised MRMR feature selection method.....	86
Table 11.5	Comparison of class-wise percentage detection accuracies and false positive rates of the dependence tree based anomaly detection method with the network attack detection methods reported in Bouzida et al. and Song et al.....	87

## LIST OF FIGURES

Figure 4.1	Procedure of Candidate Selection.....	23
Figure 4.2	Extraction of k-Means and ID3 decision tree scores from $f = 3$ candidate clusters for the test instance $Z_i$ .....	24
Figure 4.3	An example anomaly score matrix for test instance $Z$ . The anomaly scores of the k-Means method are hardened using the Threshold rule.....	26
Figure 6.1	Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the NAD-1998 test dataset.....	34
Figure 6.2	ROC Curves and AUCs of k-Means, ID3, and K-Means+ID3 with NN-Rule and NC-Rule over the NAD-1998 test dataset.....	34
Figure 6.3	Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the NAD-1999 test dataset.....	35
Figure 6.4	ROC Curves and AUCs of k-Means, ID3, and K-Means+ID3 with NN-Rule and NC-Rule over the NAD-1999 test dataset.....	36
Figure 6.5	Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the NAD-2000 test dataset.....	37
Figure 6.6	ROC Curves and AUCs of k-Means, ID3, and K-Means+ID3 methods over the NAD-2000 test dataset.....	38
Figure 6.7	Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the DED test dataset.....	39

Figure 6.8	ROC Curves and AUCs of k-Means, ID3 and K-Means+ID3 methods over the DED test dataset. ....	39
Figure 6.9	Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the MSD test dataset .....	40
Figure 6.10	ROC Curves and AUCs of k-Means, ID3 and K-Means+ID3 methods over the MSD test dataset.....	41
Figure 7.1	Dependence trees approximating the joint probability distribution $P(x_1, x_2, x_3, x_4)$ . The dependence tree in (a) approximates $P(x_1, x_2, x_3, x_4)$ as $P(x_1)P(x_2 x_1)P(x_3 x_2)P(x_4 x_1)$ and the dependence tree in (b) approximates $P(x_1, x_2, x_3, x_4)$ as $P(x_1)P(x_2 x_1)P(x_3 x_2)P(x_4)$ .....	45
Figure 11.1	Dependence trees with two features (5, 14) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	75
Figure 11.2	Dependence trees with three features (5, 14, 6) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	76
Figure 11.3	Dependence trees with four features (5, 14, 6, 32) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	76
Figure 11.4	Dependence trees with five features (5, 14, 6, 32, 23) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	77
Figure 11.5	Dependence trees with six features (5, 14, 6, 32, 23, 12) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	77
Figure 11.6	Dependence trees with seven features (5, 14, 6, 32, 23, 12, 37) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	78
Figure 11.7	Dependence trees with eight features (5, 14, 6, 32, 23, 12, 37, 31) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	79
Figure 11.8	Dependence trees with two features (23, 24) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	79

Figure 11.9	Dependence trees with three features (23, 24, 5) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	80
Figure 11.10	Dependence trees with four features (23, 24, 5, 33) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	80
Figure 11.11	Dependence trees with five features (23, 24, 5, 33, 3) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	81
Figure 11.12	Dependence trees with six features (23, 24, 5, 33, 3, 35) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	82
Figure 11.13	Dependence trees with seven features (23, 24, 5, 33, 3, 35, 34) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	82
Figure 11.14	Dependence trees with eight features (23, 24, 5, 33, 3, 35, 34, 36) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.....	83

## ACKNOWLEDGMENTS

I express my sincere and utmost gratitude to my advisor Dr. Vir V. Phoha. He patiently provided me the vision, encouragement, advice and support to progress through my research endeavors and doctoral study at Tech. Dr. Phoha has been crucial in ensuring my academic, professional, financial and moral well-being ever since I became his student. In every sense, none of this work would have been possible without him.

I express special thanks to my committee members Dr. Weizhong Dai, Dr. Jinko Kanno, Dr. Andrei Paun and Dr. Galen Turner for their encouragement, guidance and helpful suggestions. I sincerely thank our research collaborators Dr. Asok Ray (Penn State University), Dr. Sitarama Iyengar (Louisiana State University), Dr. Shashi Phoha (Penn State University), and Dr. Rastko Selmic (Louisiana Tech University). Their encouragement and support have been invaluable. I thank several anonymous reviewers for their help and comments that improved various published papers.

I express gratitude to all my dear friends and colleagues for their constant encouragement and support. A penultimate thank-you goes to my family members for always being there when I needed them the most. My final and most heartfelt acknowledgment goes to my wife Prashanthi.

# CHAPTER 1

## INTRODUCTION

Anomaly detection is the process of identifying *unusual events* occurring in a system by analyzing the audit data generated from monitoring the system's activities. Some examples of a "system" include a computer host running several software applications, a computer network comprising hundreds of nodes, an electronic circuit implementing arithmetic operations, and a mechanical mass beam structure under fatigue stress. Some examples of "unusual events" include unexpected behaviors of software applications (e.g., sudden shutdowns, memory and buffer overflows, etc.) in the case of a computer host, unexpected bursts in TCP/IP traffic passing through a computer network, unexpected response measurement from an electronic circuit, and the occurrence of an evolving crack in a mass beam structure.

Based on the past research activity in anomaly detection, anomaly detection can be broadly classified into (1) supervised anomaly detection, and (2) unsupervised anomaly detection. In supervised anomaly detection, the types of anomalies that may occur in a system are known *a priori*. Therefore, the problem of supervised anomaly detection becomes that of identifying whether an event is an anomaly, and if it is, then the specific type of anomaly to which it belongs. On the other hand, in unsupervised anomaly detection, the types of anomalies that may occur are largely unknown. Therefore, a



typical approach to solve the unsupervised anomaly detection problem is to build a profile of normal behavior of the system and then identify all events that significantly deviate from the normal profile as anomalies.

Recently, new forms of anomaly detection employing data mining techniques [1], called Anomaly Detection using Data Mining (ADDM) methods, have emerged in the literature. ADDM methods perceive the anomaly detection problem as a data classification problem in which data instances, representing events, are classified as normal or as anomalies. To classify data instances, ADDM methods employ a wide range of data mining and machine learning techniques like neural networks [2][3], decision trees [4], support vector machines [5], fuzzy logic [6], symbolic dynamics [7][8], self-organizing maps [9], Markov chain models [10], discrete Markov models [11], and association rules [12]. ADDM methods have gained popularity in both supervised and unsupervised anomaly detection fields because of their abilities to (1) automatically extract anomaly signatures, (2) detect new anomalies, (3) maintain high detection accuracies with low false positive rates, and (4) scale on large distributed datasets.

In this dissertation, we present two supervised anomaly detection methods: (1) the K-Means+ID3 anomaly detection method and (2) the dependence tree based anomaly detection method. Both methods build on existing data mining methods (i.e., k-Means clustering, ID3 decision tree learning, and dependence tree approximations of joint probability densities) and therefore can be classified as ADDM methods. However, the two anomaly detection methods differ in their approach to solve the supervised anomaly detection problem. The first method, K-Means+ID3, is designed to identify data instances as either “normal” or “anomaly”. That means, from a classification point-of-view, the K-

Means+ID3 method performs a two-category classification. Applications of such “two-category” anomaly detection approaches exist in domains where data instances may be known to originate due to an anomaly but the class or type of anomaly itself is unknown. For example, in the case of detecting an evolving crack in a mechanical system under fatigue stress, it may be known that the data instances generated during the evolution of the crack corresponds to an anomaly; however, to what specific anomaly type the data instance belongs is unknown. The second method, dependence tree based anomaly detection, is designed to identify specific types of anomalies. That is, the dependence tree based anomaly detection method performs multi-category classification. Applications of such “multi-category” approaches to anomaly detection exist in domains where data instances are known to belong to normal or to specific types of anomalies. An example of such an application domain is the detection of computer network attacks, where the type of anomaly is known to belong to one of the several attack types like denial-of-service attack, probing attack, user-to-root access attack, etc. Next, we present a brief overview of the K-Means+ID3 anomaly detection method and the dependence tree based anomaly detection method.

### **1.1 K-Means+ID3 Anomaly Detection Method**

The K-Means+ID3 anomaly detection method cascades two data mining algorithms: (1) k-Means clustering [13] and (2) the ID3 decision tree learning [14] for classifying normal and anomaly data instances. In the first stage of K-Means+ID3, k-Means clustering is performed on training instances to obtain  $k$  disjoint clusters using Euclidean distance similarity. The k-Means algorithm is used to organize the training instances into disjoint subsets or “clusters,” where each member in a cluster is more

closely related to other members in its associated cluster than to members in other clusters. In the second stage of K-Means+ID3, each cluster of training instances is further subject to ID3 decision tree learning. In ID3 learning, the ID3 algorithm builds a decision tree from the cluster of training instances. The leaf nodes of the ID3 decision tree contain the class name whereas a non-leaf node is a decision node. Each leaf node contains one of the two classes: (1) normal or (2) anomaly.

Once the training set is organized into clusters and associated ID3 decision trees, test data is compared to the classification system established by the training data set. Using this classification system, an unknown test instance is (1) examined for closeness to the clusters, and (2) for the closest clusters (i.e., the clusters that are closest by Euclidean distance between the test data instance and the clusters' centroids), the decision on the test instance as normal or anomaly is given by each cluster's ID3 decision tree. The ID3 decision tree's decision is contrasted with the k-Means cluster's decision, and the first conformity between the two decisions is the decision assigned to the unknown data instance.

Experiments were performed on three datasets: (1) Network Anomaly Data (NAD), (2) Duffing Equation Data (DED), and (3) Mechanical System Data (MSD), which contain measurements from three distinct application domains of computer networks, an electronic circuit implementing a forced Duffing equation, and a mechanical mass-beam system respectively. Anomaly detection performance of the K-Means+ID3 method was gauged using six performance measures: (1) detection accuracy, (2) false positive rate, (3) area under Receiver Operating Characteristic (ROC) curve, (4) precision, (5) total accuracy (or recall), and (6) F-measure. Results show that the

detection accuracy of the K-Means+ID3 method is as high as 96.24% at a false positive rate of 3.66% on NAD; the total accuracy is as high as 80.01% on MSD and 79.9% on DED.

## 1.2 Dependence Tree Based Anomaly Detection Method

Dependence trees approximate an  $n$ -dimensional joint probability distribution as a product of second-order component distributions, that is, probability distributions conditioned on at most one variable. The component probability distributions are selected such that they maximize the mutual information [15] between features. For the purpose of anomaly detection, we use dependence trees to approximate the class conditional probability density “ $P(X | \omega)$ ”, which is the probability that an unknown data instance  $X$  occurs given that it belongs to a class variable  $\omega$ . The class variable  $\omega$  is one of normal, a denial-of-service attack, a probing attack, a user-to-root attack, or a remote-to-login attack. Once the class conditional probability densities are obtained, through Bayes formula [13], the class conditional probability density is transformed to posterior probability  $P(\omega | X)$ , which is the probability of occurrence of a class  $\omega$  given an unknown data instance  $X$ . The data instance  $X$  is then assigned to a class with the highest posterior probability.

Because dependence tree construction is an optimization procedure that maximizes mutual information between features, the features with high correlation tend to have high mutual information (see [16]). However, when features are highly correlated, the respective class-discriminative power changes little if some of the features are removed. Therefore, to reduce correlation within the features in dependence trees, we perform feature selection using the supervised and unsupervised versions of the

Maximum Relevance Minimum Redundancy (MRMR) feature selection method. Further, we theoretically show that the MRMR feature selection criterion minimizes an upper bound on the Bayes error rate.

To demonstrate the performance of the dependence tree based anomaly detection method, experiments were performed on KDD Cup 1999 benchmark intrusion detection datasets [41]. The KDD Cup 1999 datasets contain labeled instances of normal and attack traffic originating from the MIT-DARPA simulated computer network testbed [17]. Results show that the dependence tree based anomaly detection has category-wise detection accuracy as high as 99.76% for normal, 93.88% for denial-of-service attacks, 94.88% for probe attacks, 86.40% for user-to-root attacks, and 24.44% for remote-to-login attacks. The dependence tree based anomaly detection method is further compared with two classifier based anomaly detection methods: (1) the naïve Bayes [13] anomaly detection and (2) the ID3 decision tree based anomaly detection as well as with intrusion detection models reported in Bouzida et al. [18] and Song et al. [19].

### **1.3 Contributions of the Dissertation**

The contributions of the dissertation are enumerated as follows:

- The dissertation presents “K-Means+ID3,” a novel method to cascade the k-Means clustering and ID3 decision tree learning methods for mitigating the *Forced Assignment* and *Class Dominance* problems of the k-Means method for classifying data originating from normal and anomalous behaviors in a computer network, an active electronic circuit, and a mechanical mass beam system under fatigue stress.

- The dissertation evaluates the performance of K-Means+ID3 classifier and compares it with the individual k-Means clustering and ID3 decision tree methods using six performance measures.
- From a classification perspective, the dissertation presents a novel method for cascading two successful data partition methods for improving classification performance. From an anomaly detection perspective, the dissertation presents a high performance anomaly detection method.
- The dissertation presents a dependence tree based anomaly detection method for detecting attacks on a computer network. Further, the Maximal Relevance Minimum Redundancy (MRMR) feature selection method is used to obtain an optimal set of features for attack detection using dependence trees.
- The dissertation presents a theoretical relationship between dependence tree classification error and Bayes error rate and shows that the dependence tree approximation procedure minimizes an upper bound on Bayes error rate.
- The dissertation presents a theoretical relationship between MRMR feature selection and Bayes error rate and shows that the MRMR feature selection criterion minimizes an upper bound on the Bayes error rate.
- The dissertation evaluates the performance of the dependence tree based anomaly detection method in detecting attacks on a computer network, and compares it with two popular classification methods, namely, the naïve Bayes method and the ID3 decision tree method.

The rest of the dissertation is organized as follows. Chapter 2 discusses related research in anomaly detection. Chapter 3 introduces the K-Means+ID3 anomaly detection

method and discusses anomaly detection with individual k-Means and ID3 methods. Chapter 4 details the training and testing phases of the K-Means+ID3 method. Chapter 5 discusses the datasets used to evaluate the K-Means+ID3 method. Chapter 6 presents the results of the K-Means+ID3 method. Chapter 7 presents the dependence tree based anomaly detection method. Chapter 8 presents theoretical relationship between the dependence tree classification error and Bayes error rate. Chapter 9 introduces the MRMR feature selection method and presents the relationship between MRMR feature selection criterion and the Bayes error rate. Chapter 10 discusses the KDD Cup 1999 dataset used for evaluating the performance of the dependence tree based anomaly detection method. Chapter 11 presents the results of the MRMR feature selection method and the dependence tree based anomaly detection method. We conclude the dissertation and identify future research directions in Chapter 12.

## CHAPTER 2

### RELATED RESEARCH

In this chapter, we present recent research work related to anomaly detection. This chapter is divided into two sections. The first section presents related research on anomaly detection in computer networks. The second section presents related research on anomaly detection in other domains (e.g., mechanical fatigue-crack detection, anomaly detection in computer hosts, anomaly detection in electronic circuits, etc).

#### **2.1 Anomaly Detection Research in Computer Networks**

Sarasamma et al. in [9] presented a multilevel hierarchical Kohonen network to implement a network anomaly detection system. Their motivation for implementing a multilevel Kohonen network was that the single-layered Kohonen network, though effective in grouping similar input vectors into clusters, does not guarantee an optimal separation of resulting clusters. Further, the experiments with single-layered Kohonen network on KDD Cup 1999 network intrusion detection data have resulted in unacceptably high false positive rates. On the other hand, an anomaly detection system based on a multilevel hierarchical Kohonen network combined with domain knowledge based grouping of features in KDD Cup 1999 dataset has resulted in detection accuracy as high as 97.19% for denial-of-service attacks, 87.88% for probe attacks, 66.52% for



user-to-root attacks, and 0.37% for remote-to-login attacks. These detection accuracies were achieved at a false positive rate of 2.92%, which was the lowest false positive rate achieved by the hierarchical Kohonen network.

Sarasamma et al. in [20] presented a hyperellipsoidal clustering technique for supervised anomaly. The hyperellipsoidal clustering technique is implemented as a self-organizing map in which the winning cluster is decided based on the Mahalanobis distance between the input vector and the cluster mean. A new cluster is initiated if the Mahalanobis distance between the input vector and the winning cluster center is greater than a predefined threshold. The motivation for using Mahalanobis distance as a transfer function in the self-organizing map comes from the assumption that each cluster originates from a multivariate Gaussian distribution function. The locus of points of constant density for a multivariate Gaussian distribution function geometrically forms a hyper ellipsoid with constant radius given by Mahalanobis distance. By applying hyperellipsoidal clustering on KDD Cup 1999 network traffic datasets, Sarasamma *et al.* [20] achieved a detection accuracy of 83.97% for denial-of-service attacks, 91.31% for probe attacks, 84.56% for user-to-root attacks, and 33.78% for remote-to-login attacks at a false positive rate of 2.68%.

Song et al. in [19] presented a genetic programming approach to supervised anomaly detection. The Random Subset Selection–Dynamic Subset Selection (RSS-DSS) algorithm was proposed for dynamically filtering large datasets for subsequent classification in genetic programming paradigm. The RSS-DSS algorithm initially divides the entire training set into small blocks to allow incremental loading on to the main memory. Next, a block is selected with uniform probability and a subset of training

patterns are selected from the block based on two parameters: (1) age and (2) difficulty. Training is performed on the selected subset of patterns through an evolutionary phase which involves “crossover” and “mutation” operations of genetic programming. Three different fitness functions: (1) equal class cost, (2) variable class cost, and (3) hierarchical cost are used to determine the best evolutionary phases. In [19], experiments were conducted on KDD Cup 1999 datasets and only the first eight “basic” features were utilized. The RSS-DSS genetic programming approach achieved a detection accuracy of 95.6% for denial-of-service attacks, 48.5% for probe attacks, 10.1% for user-to-root attacks, and 0.2% for root-to-login attacks. These attack detection accuracies were reported at a false positive rate of 0.27%.

Qu et al. in [21] introduced a new correlation measure to select features for classification and data mining tasks. Their correlation measure, measured between any two features, is known as Decision Dependent Correlation (DDC). The DDC measures correlation in terms of two components: (1) the correlation between features and the class variable, calculated as sum of mutual information between two features and the class variable, and (2) the correlation between features, calculated as the mutual information between the two features. The first component quantifies the relevance of features with the class variable and the second component quantifies the redundancy within features. Feature selection is performed by maximizing the first component and minimizing the second component of the DDC measure. Qu et al. [21] performed experiments on the KDD Cup 1999 dataset. By incrementally using DDC measure to select a set of features and by using a linear discriminant function to perform classification, Qu et al. achieved 99.93% detection accuracy with 0.07% false positive rate for denial-of-service attacks,

99.91% detection accuracy with 0.09% false positive rate for probe attacks, 91.13% detection accuracy with 9.258% false positive rate for user-to-root attacks, and 92.47% detection accuracy with 8.35% false positive rate for remote-to-login attacks. It is to be noted here that Qu et al. performed two-category attack detection (i.e., identifying a single attack type in the presence of normal traffic) as opposed to multi-category attack detection, as performed by earlier mentioned works of Sarasamma et al. [9][20], Song et al. [19], and the dependence tree based anomaly detection method of this dissertation.

Bouzida et al. in [18] performed anomaly detection in network traffic using nearest neighbor classification and C4.5 decision trees. The use of nearest neighbor classification for anomaly detection was motivated by the fact that its classification error is bounded by twice the Bayes error rate, the least possible classification error that can be achieved by any classification method (see [22]). The use of C4.5 decision trees for anomaly detection in network traffic was motivated by the robustness of C4.5 decision trees in handling noisy data and by their high classification performance in various application domains such as automated patient classification, image classification, etc. (see [23]). Further, to reduce the dimensionality of data, Bouzida et al. performed feature extraction using principal component analysis. Experiments performed on the KDD Cup 1999 datasets using the nearest neighbor classification rule and four principal components resulted in 97.14% detection accuracy for denial-of-service attacks, 74.4% detection accuracy for probe attacks, 7.91% detection accuracy for user-to-root attacks, and 0.80% for root-to-login attacks, at a false positive rate of 0.5%. Experiments with C4.5 decision trees and four principal components resulted in 97.25% detection accuracy for denial-of-service attacks, 66.80% detection accuracy for probe attacks, 6.58%

detection accuracy for user-to-root attacks, and 0.01% detection accuracy for remote-to-login attacks, with 1.0% false positive rate.

## **2.2 Anomaly Detection Research in Other Domains**

Khatkhate et al. in [1] presented the symbolic time series analysis method for detecting anomalies resulting from fatigue cracks in ductile alloys. The first step in symbolic time series analysis involved partitioning the time series for constructing symbol sequences. For this purpose, Khatkhate et al. [1] used wavelet space partitioning, in which the time series data was converted into wavelet space at different scales and time shifts. Next, graphs of wavelet coefficients versus the scales were stacked starting with the smallest value of the scale and ending with the largest value. This wavelet space was then partitioned using the maximum entropy partitioning. A Hidden Markov Model (HMM) was used to probabilistically score a given set of symbols as normal or anomaly. The scores from the HMM were compared with the scores from three machine learning methods, namely the Principle Component Analysis (PCA), Multi-Layer Perceptron Neural Network (MLPNN), and Radial Basis Function Neural Network (RBFNN). Khatkhate et al. empirically demonstrated that the symbolic time series method for anomaly detection outperforms PCA, MLPNN, and RBFNN in identifying early fatigue crack anomalies, while all the four methods performed well in detecting evolved anomalies.

Chin et al. in [7] applied the concept of symbolic time series based anomaly detection to detect anomalies in a non-linear electronic system. The symbolic time series method was tested on an electronic system implementing a second-order, non-autonomous forced Duffing equation. The dissipation parameter ' $\beta$ ', implemented as

resistance in the electronic circuit was varied between 0.10 - 0.35 to generate system response. The system response at  $\beta = 0.10$  was recorded as normal and the response at  $\beta > 0.10$  was recorded as anomaly. Three machine learning methods—namely PCA, MLPNN, and RBFNN—were used to compare the anomaly detection performance of the symbolic time series method. Results of the experiments in [7] showed that the symbolic time series analysis method outperformed the PCA, MLPNN, and RBFNN methods in detecting slowly evolving anomalies, i.e., the system response when  $\beta$  is between 0.15 and 0.27.

Ye et al. [24] present multivariate statistical analysis of audit trails for host-based intrusion detection. Hotelling's  $T^2$  test and the chi-squared ( $\chi^2$ ) test, which are multivariate statistical process control techniques, were used to analyze audit trails. The Hotelling's  $T^2$  statistic and the  $\chi^2$  statistic were used to calculate the amount of deviation between a given test sample and the normal (in-control) population. Experiments were performed on two datasets: (1) a four hour Basic Security Model (BSM) audit trail data consisting of 1,406 audit trails of normal events and 1,225 events of intrusive activities and 2) a large BSM audit trail containing 3,174,584 normal events and 48,000 audit trails of intrusive events. Accuracy results and Receiver Operating Characteristic (ROC) curves on these datasets showed that the chi-squared test results were either better than or comparable to the Hotelling's test for both the datasets. The reason for the relatively better performance of the chi-squared test, as hypothesized by Nong Ye et al., was that the intrusive activities manifest themselves mainly through mean shifts, which the chi-squared test has captured very effectively.

Chang et al. in [25] performed anomaly detection and classification in hyperspectral imagery using two methods: (1) Reed and Yu (RXD) method based on Mahalanobis distance and (2) Low Probability Detection (LPD) method. In hyperspectral image analysis, anomaly detection refers to the identification of targets whose signatures are distinct from their surroundings. Chang et al. demonstrated that the RXD method coupled with linearly-constrained minimum variance classification method outperformed the LPD anomaly detection method in detecting anomalies in hyperspectral images of forest landscapes. Additionally, methods for anomaly detection appear in [42], [43], [44], [45], [46], [47], [48], and [49].

## CHAPTER 3

### THE K-MEANS+ID3 ANOMALY

#### DETECTION METHOD

In this chapter, we introduce the K-Means+ID3 method for anomaly detection and briefly discuss anomaly detection with individual k-Means and ID3 learning algorithms. K-Means+ID3 is a novel supervised anomaly detection method developed by cascading two machine learning algorithms: (1) the k-Means clustering and (2) the ID3 decision tree learning. In the first stage, k-Means clustering is performed on training instances to obtain  $k$  disjoint clusters. Each k-Means cluster represents a region of similar instances, “similar” in terms of Euclidean distances between the instances and their cluster centroids. We choose k-Means clustering because (1) it is a data-driven method with relatively few assumptions on the distributions of the underlying data and (2) the greedy search strategy of k-Means guarantees at least a local minimum of the criterion function, thereby accelerating the convergence of clusters on large datasets. In the second stage of K-Means+ID3, the k-Means method is cascaded with the ID3 decision tree learning by building an ID3 decision tree using the instances in each k-Means cluster. Cascading the k-Means clustering method with ID3 decision tree learning alleviates two problems in k-Means clustering: (1) the *Forced Assignment* problem, and (2) the *Class Dominance* problem. The *Forced Assignment* problem arises when parameter  $k$  in k-Means is set to a

value considerably less than the inherent number of natural groupings within the training data. The k-Means procedure initialized with a low  $k$  value under estimates the natural groups within the training data and therefore will not capture the overlapping groups within a cluster, forcing the instances from different groups to be a part of the same cluster. Such ‘forced assignments’ in anomaly detection may increase the false positive rate or decrease the detection accuracy. As an example of forced assignment in an anomaly detection setting, consider an anomaly in network traffic originating from a particular type of attack (say a ‘remote-to-user’ attack) whose network traffic may be very similar to that of normal traffic. In this case, a low value of  $k$  may force the k-Means to assign attack instances to a normal cluster because the value of  $k$  is insufficient to capture the inherent sub-group structure of the attack that differentiates it from the normal traffic; more specifically, the distance (similarity) between the attack instance and the cluster representing a normal class is less than the distance between the attack instance and the cluster representing an anomaly class. The second problem which K-Means+ID3 alleviates, *Class Dominance*, arises in a cluster when the training data have a large number of instances from one particular class and very few instances from the remaining classes. Such clusters, dominated by a single class, show weak association to the remaining classes. That is, when classifying an anomaly associated with a cluster dominated by normal instances or vice-versa, decisions based exclusively on the probabilistic likelihood of the instance being associated with the cluster will most likely misclassify the instance. The *Forced Assignment* and *Class Dominance* problems cause instances from different classes, like the normal and anomaly classes in our case, to overlap within the same cluster. However, a decision tree trained on each cluster learns



the sub-grouping (if any) present within each cluster and refines the decision boundaries within the clusters dominated by a single class by partitioning the instances with a set of *if-then constraints* over the feature space. Cascading the decisions from the k-Means and ID3 methods involves two phases: (1) the Candidate Selection phase, and (2) the Candidate Combination phase. In the Candidate Selection phase,  $f$  clusters nearest in Euclidean distance between the cluster centroids and the test instance are selected. In the Candidate Combination phase, two rules are used—(1) the nearest consensus rule and (2) the nearest neighbor rule—to combine the decisions of the k-Means and the ID3 algorithms to obtain a final classification decision over a test instance.

We perform experiments on three datasets: (1) the network anomaly data, which is feature extracted from the 1998, 1999, and 2000 MIT-DARPA network traffic [17][26][27] using an artificial neural network based non-linear component analysis method presented in [28]; (2) the Duffing equation data [7], containing measurements from an active electronic circuit implementing a forced Duffing equation; and (3) the mechanical systems data [1], containing measurements drawn from a mechanical apparatus that excites a mass-beam structure for generating small fatigue cracks. The three datasets contain representative anomalous and normal behavioral patterns from three distinct domains of computer networks, an active electronic circuit system, and a mechanical system. Performance evaluation of the K-Means+ID3 cascading approach is conducted using six measures: (1) detection accuracy or True Positive Rate (TPR), (2) False Positive Rate (FPR), (3) precision, (4) total accuracy (or accuracy), (5) F-measure, and (6) Receiver Operating Characteristic (ROC) curves and Areas Under ROC Curves (AUCs). The performance of K-Means+ID3 is empirically compared with the

performance of individual k-Means clustering and the ID3 decision tree classification algorithms. Next, we briefly discuss the individual k-Means clustering and the ID3 decision tree learning methods for anomaly detection.

### **3.1 Anomaly Detection with Individual K-Means Clustering Method**

The k-Means algorithm groups  $N$  data points into  $k$  disjoint clusters where  $k$  is a predefined parameter. The steps in the k-Means clustering based anomaly detection method are as follows.

1. Select  $k$  random instances from the training data subset as the centroids of the clusters  $C_1, C_2, \dots, C_k$ .
2. For each training instance  $X$ :
  - a. Compute the Euclidean distance  $D(C_i, X)$ ,  $i = 1 \dots k$ . Find cluster  $C_q$  that is closest to  $X$ .
  - b. Assign  $X$  to  $C_q$ . Update the centroid of  $C_q$ . (The centroid of a cluster is the arithmetic mean of the instances in the cluster.)
3. Repeat step (2) until the centroids of clusters  $C_1, C_2, \dots, C_k$  stabilize in terms of mean-squared-error criterion.
4. For each test instance  $Z$ :
  - a. Compute the Euclidean distance  $D(C_i, Z)$ ,  $i = 1 \dots k$ . Find cluster  $C_r$  that is closest to  $Z$ .

- b. Classify  $Z$  as an anomaly or a normal instance using either the *Threshold* rule or the *Bayes Decision* rule. The *Threshold* rule for classifying a test instance  $Z$  that belongs to cluster  $C_r$  is

$$\text{Assign } Z \rightarrow 1 \text{ if } P(\omega_{1r} | Z \in C_r) > \tau; \text{ Otherwise } Z \rightarrow 0,$$

where ‘0’ and ‘1’ represent normal and anomaly classes,  $\omega_{1r}$  represents the anomaly class in cluster  $C_r$ ,  $P(\omega_{1r} | Z \in C_r)$  represents the probability of anomaly instances in  $C_r$ , and  $\tau$  is a predefined threshold. In our experiments, the threshold is set to 0.5 so that a test instance is classified as an anomaly only if it belongs to a cluster that has anomaly instances in majority. The *Bayes Decision* rule is

$$\text{Assign } Z \rightarrow 1 \text{ if } P(\omega_{1r} | Z \in C_r) > P(\omega_{0r} | Z \in C_r); \text{ Otherwise } Z \rightarrow 0,$$

where  $\omega_{0r}$  represents the normal class in cluster  $C_r$ ,  $P(\omega_{0r} | Z \in C_r)$  is the probability of normal instances in cluster  $C_r$ .

### 3.2 Anomaly Detection with ID3 Decision Tree Learning

The ID3 decision tree learning algorithm computes the Information Gain  $G$  on each attribute  $A$ , defined as

$$G(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v),$$

where  $S$  is the total input space,  $S_v$  is the subset of  $S$  for which attribute  $A$  has a value  $v$ .

The  $Entropy(S)$  over  $c$  classes is given by  $\sum_{i=1}^c -p_i \log_2(p_i)$ , where  $p_i$  represents the probability of class ‘ $i$ ’. The attribute with the highest information gain, say  $B$ , is chosen

as the root node of the tree. Next, a new decision tree is recursively constructed over each value of  $B$  using the training subspace  $S - \{S_B\}$ . A leaf-node or a decision-node is formed when all the instances within the available training subspace are from the same class. For detecting anomalies, the ID3 decision tree outputs binary classification decision of '0' to indicate normal and '1' to indicate anomaly class assignments to test instances.

**CHAPTER 4**

**TRAINING AND TESTING THE**

**K-MEANS+ID3 ANOMALY**

**DETECTION METHOD**

We are provided with a training dataset  $(X_i, Y_i)$ ,  $i = 1, 2, \dots, N$  where  $X_i$  represents an  $n$ -dimensional continuous valued vector and  $Y_i = \{0, 1\}$  represents the corresponding class label with '0' for normal and '1' for anomaly. The K-Means+ID3 method has two steps: (1) training and (2) testing. During training, steps 1-3 of the k-Means based anomaly detection method are first applied to partition the training space into  $k$  disjoint clusters  $C_1, C_2, \dots, C_k$ . Then, an ID3 decision tree is trained with the instances in each k-Means cluster. The k-Means method ensures that each training instance is associated with only one cluster. However, if there are any sub-groups or overlaps within a cluster, the ID3 decision tree trained on that cluster refines the decision boundaries by partitioning the instances with a set of *if-then rules* over the feature space. The testing step of the K-Means+ID3 has two phases: (1) the Candidate Selection phase and (2) the Candidate Combination phase. In Candidate Selection, decisions from k-Means and ID3 based anomaly detection methods are extracted. In Candidate Combination, the decisions of the k-Means and ID3 decision tree methods are combined to give a final decision on the class membership of a test instance. For combining the k-

Means and ID3 decision tree methods, we present two combination rules: (1) the nearest neighbor rule, and (2) the nearest consensus rule. A detailed explanation of the two phases follows.

#### 4.1 The Candidate Selection Phase

Figure 4.1 presents the procedure for the Candidate Selection. Let  $DT_1, DT_2, \dots, DT_k$  be the ID3 decision trees on clusters  $C_1, C_2, \dots, C_k$  formed by applying the k-Means method on the training instances. Let  $r_1, r_2, \dots, r_k$  be the centroids of  $C_1, C_2, \dots, C_k$  respectively. Given a test instance  $Z_i$ , the Candidate Selection procedure extracts anomaly scores from  $f$  candidate clusters  $G_1, G_2, \dots, G_k$ . The ' $f$  candidate clusters' are  $f$  clusters in  $C_1, C_2, \dots, C_k$  that are nearest to  $Z_i$  in terms of the Euclidean distance between  $Z_i$  and the cluster centroids. Here,  $f$  is a user-defined parameter.

```

Input: Test instances  $Z_i, i = 1 \dots n; f$  value.
Output: Anomaly score matrix for  $Z_i, i = 1 \dots n$ .

Procedure Candidate_Selection {
  Step 1: For each test instance  $Z_i$ 
    a. Compute Euclidean distance  $D(Z_i, r_j), j = 1 \dots k$ ,
       and find  $f$  clusters closest to  $Z_i$ 
    b. Compute k-Means and ID3 decision tree scores
       for  $f$  nearest (candidate) clusters.
  Step 2: Return Anomaly Score Matrix for  $Z_i$ .
} /* End Procedure */

```

Figure 4.1 Procedure of Candidate Selection.

Figure 4.2 illustrates the extraction of anomaly scores from k-Means clustering and ID3 decision tree learning methods for  $f$  candidate clusters. Let  $m_1, m_2, \dots, m_f$  represent the centroids of candidate clusters  $G_1, G_2, \dots, G_f$ . Let  $D(Z_i, m_1) = d_1$ ,  $D(Z_i, m_2) = d_2$ , and  $D(Z_i, m_f) = d_f$ , represent the Euclidean distances between the test vector  $Z_i$  and the  $f$  candidate clusters. The k-Means anomaly scores  $P_s$ ,  $s = 1, \dots, f$ , for each of the  $f$  candidate clusters is given by

$$P_s = P(\omega_{1s}) \times \left[ 1 - \frac{d_s}{\sum_{l=1}^k D(Z_i, r_l)} \right] \quad \text{Equation 4.1}$$

where  $P(\omega_{1s})$  is the probability of anomaly instances in cluster 's'. In Equation 4.1, the

term ' $1 - \frac{d_s}{\sum_{l=1}^k D(Z_i, r_l)}$ ' is called the *Scaling Factor (SF)*.

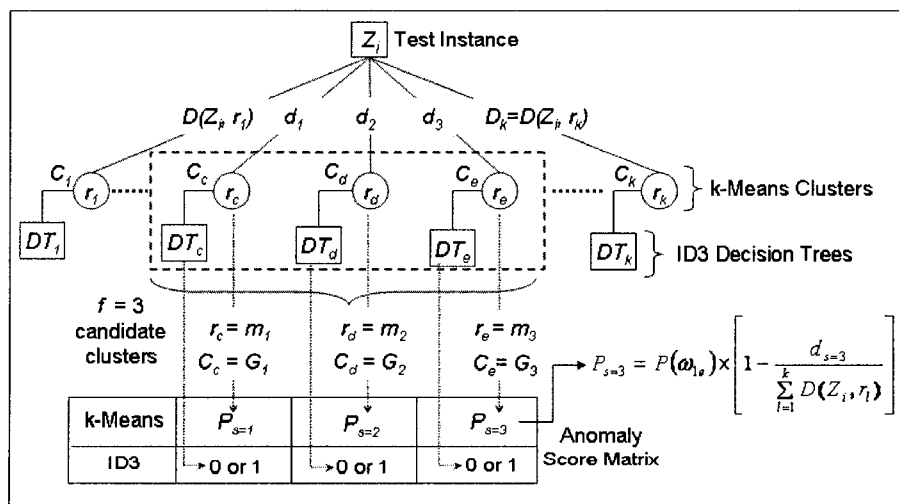


Figure 4.2 Extraction of k-Means and ID3 decision tree scores from  $f=3$  candidate clusters for the test instance  $Z_i$ .

The  $SF$  scales  $P(\omega_{1s})$  by weighing it against the ratio of the Euclidean distance between the cluster  $s$  and  $Z_i$  and the sum of Euclidean distances between  $Z_i$  and the clusters  $C_1, C_2, \dots, C_k$ . The  $SF$  penalizes the probability of anomaly  $P(\omega_{1s})$  in cluster  $s$  with its distance from the test vector  $Z_i$  i.e., a high value of  $d_s$  yields a low  $P_s$  value and vice versa. The decisions from the ID3 decision trees associated with the  $f$  candidate clusters are either '0' representing normal or '1' representing an anomaly classes. The Candidate Selection phase outputs an anomaly score matrix with the decisions extracted from the k-Means and ID3 anomaly detection methods for a given test vector. The decisions stored in the anomaly score matrix are combined in the Candidate Combination phase to yield a final decision on the test vector. A detailed description of the Candidate Combination follows.

#### **4.2 The Candidate Combination Phase**

The input to the Candidate Combination phase is the anomaly score matrix containing the anomaly scores  $P_s$ ,  $s = 1, \dots, f$ , of the k-Means and the decisions of the ID3 based anomaly detection methods over  $f$  candidate clusters. To combine the decisions of the k-Means and ID3 algorithms, we first harden the anomaly scores of the k-Means method by using the *Threshold Rule* presented in Section 2.1. Next, we use two rules: (1) the nearest consensus rule and (2) the nearest neighbor rule to combine the decisions.



#### 4.2.1 The Nearest Consensus Rule

Figure 4.3 shows an example of an anomaly score matrix for the test vector  $Z$ . The  $f$  candidate clusters  $G_1, G_2, \dots, G_f$  are ordered in the anomaly score matrix such that the distances  $d_1, d_2, \dots, d_f$  between  $Z$  and the candidate clusters  $G_1, G_2, \dots, G_f$  respectively, satisfy  $d_1 < d_2 < \dots < d_f$ . In the nearest consensus rule, the decision of the nearest candidate cluster in which consensus exists between the decisions of the k-Means and the ID3 decision tree methods is selected as the combined classification decision. For example, in the anomaly score matrix shown in Figure 4.3, the nearest consensus occurs in candidate cluster  $G_2$  and therefore the test vector is classified as '1' i.e., an anomaly.

	$G_1$	$G_2$	$G_3$	.....	$G_f$
k-Means	1	1	0	.....	1
ID3	0	1	0	.....	0

↑  
Consensus

**Figure 4.3 An example anomaly score matrix for test instance  $Z$ . The anomaly scores of the k-Means method are hardened using the Threshold rule.**

#### 4.2.2 The Nearest Neighbor Rule

The nearest neighbor rule chooses the decision of the ID3 decision tree associated with the nearest candidate cluster within the  $f$  candidate clusters. In the anomaly score matrix shown in Figure 4.3,  $G_1$  is the nearest candidate cluster to the test vector  $Z$ . Therefore, the nearest neighbor rule classifies the test vector as '0' (normal), which is the decision of the ID3 decision tree associated with candidate cluster  $G_1$ .

## CHAPTER 5

### DATASETS FOR TESTING THE K-MEANS+ID3 METHOD

In this chapter we discuss three experimental datasets: (1) Network Anomaly Data (NAD), (2) Duffing Equation Data (DED), and (3) Mechanical Systems Data (MSD). The NAD contains three data subsets: (i) NAD-98, (ii) NAD-99, and (iii) NAD-00, obtained by feature-extracting the 1998, 1999, and 2000 MIT-DARPA network traffic corpora. The DED dataset was obtained from an active non-linear electronic circuit implementing a second-order forced Duffing equation. The MSD dataset was obtained from an apparatus designed to induce small fatigue cracks in ductile alloy (mass beam) structures.

Table 5.1 summarizes the proportion of normal and anomaly instances, and the number of dimensions in the three datasets. The training and testing data subsets were randomly drawn from the original NAD, DED, and MSD datasets. The number of instances in all the training data subsets was restricted to utmost 5000 instances, with 70% of them being normal and the rest being anomaly instances. The testing datasets contain utmost 2500 unseen instances (i.e., those excluded in training data subsets), with 80% of them being normal and the remaining 20% being anomaly instances. The ratio of training datasets to the testing datasets is 65% to 35% except for the NAD-2000 and DED

datasets. The NAD-2000 and DED datasets contain comparatively less number of training and testing instances because of the limited number of normal instances available in DED and the limited number of anomaly instances available in NAD-2000. Therefore, the training-to-testing dataset ratio for DED is 60% to 40% and for the NAD-2000 is 50% to 50%. A brief description of each dataset follows.

**Table 5.1 Characteristics of the NAD, DED and MSD datasets used in the anomaly detection experiments.**

Datasets		Dimensions	Training Instances		Testing Instances	
			Normal	Anomaly	Normal	Anomaly
NAD	1998	12	3500	1500	2000	500
	1999	10	3500	1500	2000	500
	2000	10	294	126	336	84
DED		4	1288	502	860	215
MSD		4	3500	1500	2000	500

### 5.1 Network Anomaly Data

The NAD-98, NAD-99, and NAD-00 data subsets contain artificial neural network based Non-Linear Component Analysis (NLCA) feature-extracted 1998, 1999, and 2000 MIT-DARPA network traffic [28], respectively. The 1998 MIT-DARPA datasets [17] were collected on an evaluation test bed simulating network traffic similar to that seen between an Air Force base (INSIDE network) and the Internet (OUTSIDE network). Thirty-eight different attacks (documented in [17]) were launched from the OUTSIDE network. Approximately seven weeks of training data and two weeks of test data were collected by a sniffer deployed between the INSIDE and OUTSIDE network. List files provide attack labels for the seven-week training data. However, the list files

associated with the test data do not contain attack labels. For this reason, we use only the seven-week training data for both training and testing purposes. The 1999 MIT-DARPA datasets [26] were generated on a test bed similar to that used for 1998 MIT-DARPA datasets. Twenty-nine additional attacks (documented in [26]) were developed. The datasets contain approximately three weeks of training data (with two weeks of data exclusively containing normal traffic) and two weeks of test data. In our experiments we use the tcpdumps collected by the sniffer in the INSIDE network on weeks 1, 3, 4 and 5. The tcpdumps from Week-2 were excluded because the list files associated with datasets were not available. The 2000 MIT-DARPA datasets [27] are attack-scenario specific datasets. The datasets contain three attack scenarios simulated with the background traffic being similar to those in 1999 MIT-DARPA datasets. The first dataset, LLS DDOS 1.0, simulates a 3.5 hour attack scenario in which a novice attacker launches a Distributed Denial of Service (DDoS) attack against a naive adversary. The second dataset, LLS DDOS 2.0.2, is a two-hour stealthy DDoS attack scenario. The third dataset, Windows NT Attack, is a nine-hour dataset containing five phased Denial-of-Service (DoS) attack on Windows NT hosts.

## 5.2 Duffing Equation Data

This section describes the preparation of the Duffing Equation Dataset (DED). Chin et al. [7] use an active non-linear electronic circuit to generate the data. The circuit implements a second-order, non-autonomous, forced Duffing equation represented as

$$\frac{d^2 x(t)}{dt^2} + \beta(t_s) \frac{dx(t)}{dt} + x(t) + x^3(t) = A \cos \omega t$$

The dissipation parameter  $\beta(t_s)$ , implemented as resistance in the circuit, varies in the slow-time  $t_s$  and is constant in the fast time-scale  $t$  at which the dynamical system is excited. Although the system dynamics is represented by a low order differential equation, it exhibits chaotic behavior that is sufficiently complex from thermodynamic perspectives and is adequate for illustration of the anomaly detection concept. The goal is to detect, the changes in  $\beta(t_s)$ , which are associated with an anomaly. Setting the stimulus with amplitude  $A = 5.5$  and  $\omega = 5.0$  rad/sec, the stationary behavior of the system response for this input stimulus is obtained for several values of  $\beta$  in the range of 0.10 to 0.35. In all our experiments with DED, we have considered the datasets with  $\beta = 0.1$ ,  $\beta = 0.32$ ,  $\beta = 0.33$ ,  $\beta = 0.34$ , and  $\beta = 0.35$  to randomly select 1790 instances for preparing the training data subsets and 1075 unseen random instances for preparing the test data subset.

### 5.3 Mechanical Systems Data

This section discusses the preparation of Mechanical System Data (MSD). Khatkhate et al. [1] present the test apparatus to generate the MSD. The test apparatus has two subsystems: (1) the plant subsystem consisting of the mechanical structure including test specimens (i.e., the mass-beams that undergo fatigue crack damage), electromagnetic shakers, and displacement measuring sensors; and (2) the instrumentation and control subsystem consisting of the hardware and software components related to data acquisition and processing. The mechanical structure of the test apparatus is persistently excited near resonance to induce a stress level that causes fatigue cracks in the mass beam specimens and yields an average life of approximately 20,000 cycles or 36 minutes.

The mass beams attain stationary behavior in the fast-time scale of machine vibrations when persistently excited in the vicinity of its resonant frequency. Fatigue cracks occur at a slow time scale that is slow relative to the fast time scale dynamics of the vibratory motion. The goal here is to detect the slowly evolving anomaly, possibly due to fatigue cracks, by observing the time series data from displacement measuring sensors. There is a total of 36 minutes of data. The first two minutes of data are considered transient (normal) and the rest from 3 to 36 minutes of data are considered as steady state asymptotic behavior (anomaly). In all our experiments with MSD, we used the data recorded during the 1<sup>st</sup>, 33<sup>rd</sup>, 34<sup>th</sup>, 35<sup>th</sup>, and the 36<sup>th</sup> minute to randomly select 5000 instances for preparing the training data subsets and 2500 unseen random instances for preparing the test data subset.

## **CHAPTER 6**

### **RESULTS OF K-MEANS+ID3 ANOMALY**

#### **DETECTION METHOD**

In this chapter we present the results of the K-Means+ID3 method with the nearest neighbor and nearest consensus combination rules and compare it with the individual k-Means and ID3 decision tree methods over the NAD, DED, and MSD datasets. We use six measures for comparing the performance—(1) TPR or recall is the percentage of anomaly instances correctly detected, (2) FPR is the percentage of normal instances incorrectly classified as anomaly, (3) ‘precision’ is the percentage of correctly detected anomaly instances over all the detected anomaly instances, (4) ‘total accuracy’ or ‘accuracy’ is the percentage of all normal and anomaly instances that are correctly classified, (5) the ‘F-measure’ is the equally-weighted (harmonic) mean of precision and recall, and (6) the ROCs [29] and AUCs [30] give the performance of an anomaly detection system with FPR on the x-axis and TPR on the y-axis. The performance measures: precision, recall, and F-measure determine how the K-Means+ID3, the k-Means, and the ID3 methods perform in identifying anomaly instances. The performance measure ‘accuracy’ determines the number of normal and anomaly instances correctly classified by the anomaly detection methods. The measures FPR and AUC determine the

number of false positives that the anomaly detection systems generate at specific detection accuracies. The results of our experiments on the NAD, DED, and MSD follow.

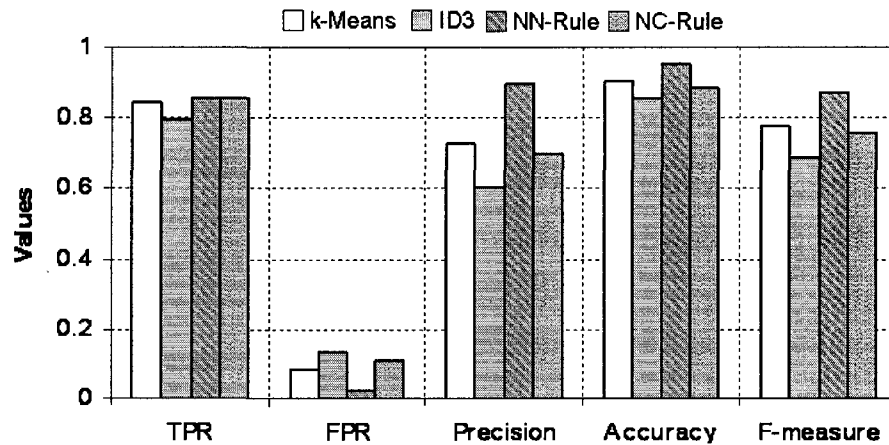
### 6.1 Results on the NAD-1998 Dataset

Here we present the results of the k-Means and ID3 decision tree based anomaly detection methods and the K-Means+ID3 method over the NAD-1998 datasets. Figure 6.1 illustrates the performance of the k-Means, the ID3, and the K-Means+ID3 methods averaged over 10 trials for k-means and K-means+ID3. For the NAD-1998 datasets, the  $k$  value of the k-Means method was set to 20. For the ID3, the training space was discretized into 45 equal-width intervals. For the K-Means+ID3 cascading method the  $k$  was set to 20 and the data discretized into 45 equal-width intervals. The choice of  $k$  value used in our experiments was based on 10 trial experiments conducted with  $k$  set to 5, 10, 15, and 20. The performance of the k-Means based anomaly detection showed no significant improvement when  $k$  value was set to a value greater than 20. Similarly, the choice of the number of equal-width intervals for discretization was based on 19 experiments conducted with different discretization values (e.g. 10, 15, ..., 100). Figure 6.1 shows that: (i) the K-Means+ID3 cascading method based on Nearest Neighbor (NN) combination rule has better performance than the k-means and ID3 in terms of TPR, FPR, Precision, and Accuracy; (ii) the TPR, FPR, Precision, Accuracy, and F-measure of the K-Means+ID3 cascading with NC combination is in-between the k-Means and the ID3; and (iii) the K-Means+ID3 with NN combination outperforms the k-Means and ID3 algorithms in terms of F-measure, obtained from combining precision and recall.

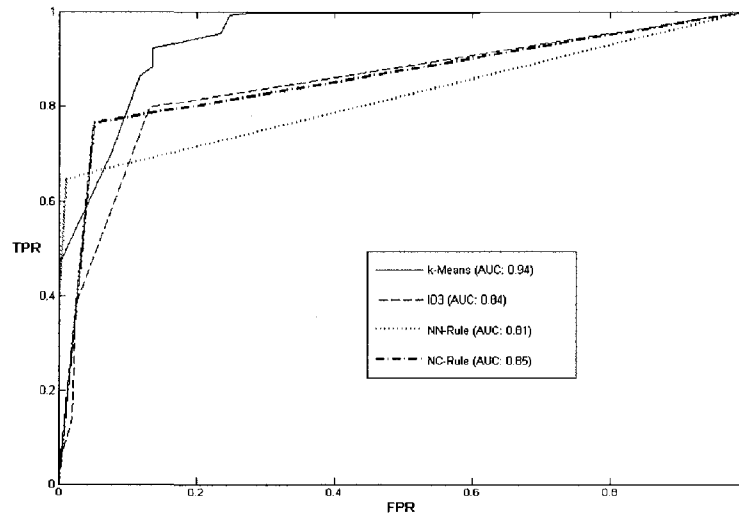
Figure 6.2 shows the ROC curves and AUC values for the k-Means, ID3 and K-Means+ID3 methods. The ROC curves for the K-Means+ID3 and the k-Means



algorithms were plotted for the trials with the AUC values closest to the mean TPR values shown in Figure 6.1. The ROC for K-Means+ID3 cascading algorithm with NN combination rule shows that the best TPR is achieved at 0.76 with an FPR as low as 0.05.



**Figure 6.1 Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the NAD-1998 test dataset.**

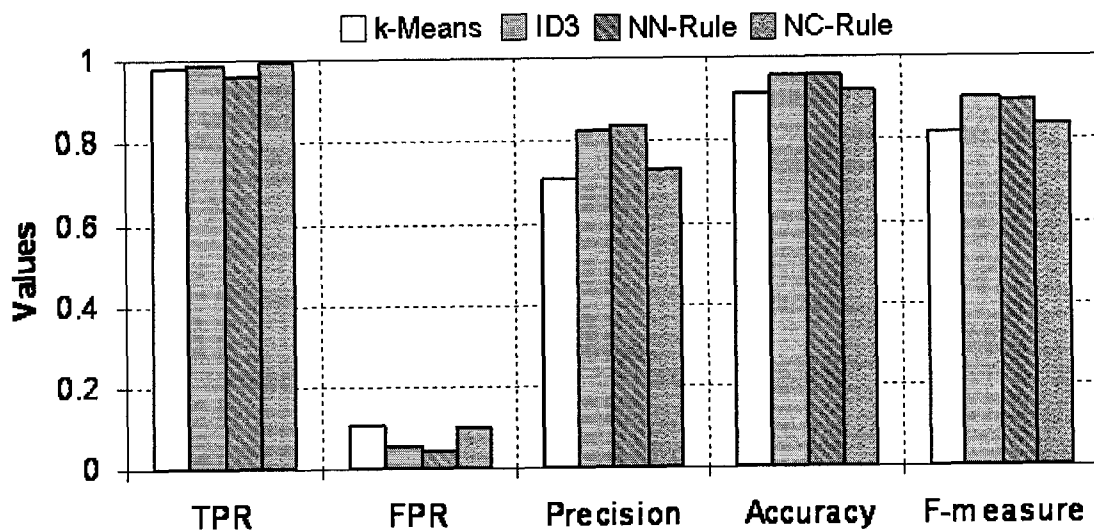


**Figure 6.2 ROC Curves and AUCs of k-Means, ID3, and K-Means+ID3 with NN-Rule and NC-Rule over the NAD-1998 test dataset.**

## 6.2 Results on the NAD-1999 Dataset

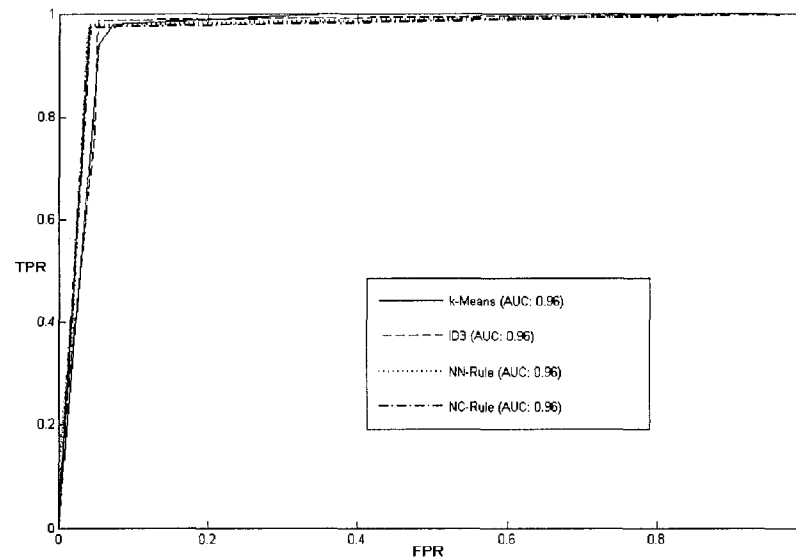
Figure 6.3 illustrates the performance of the k-Means, the ID3, and the K-Means+ID3 methods averaged over 10 trials for k-Means and K-Means+ID3. For the NAD-1999 datasets, the  $k$  value of individual k-Means was set to 5. For the ID3 algorithm, the training space was discretized into 25 equal-width intervals. For the K-Means+ID3 cascading, the value of  $k$  was set to 5 and the data was discretized into 25 equal-width intervals.

Figure 6.3 shows that (i) the K-Means+ID3 cascading with NC combination has better performance than the k-Means and ID3 in terms of TPR, and (ii) precision, accuracy, and F-measure of the K-Means+ID3 with NN combination is higher than the k-Means and ID3.



**Figure 6.3 Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the NAD-1999 test dataset.**

Figure 6.4 shows the ROC curves and AUC values of the k-Means, ID3 and K-Means+ID3 methods over NAD-1999. The ROC curves for K-Means+ID3 and k-Means method were plotted for the trial with the AUC values closest to the mean TPR values shown in Figure 6.3. The K-Means+ID3 cascading with NN and NC combination has the same AUC performance as compared to k-Means and ID3 methods.

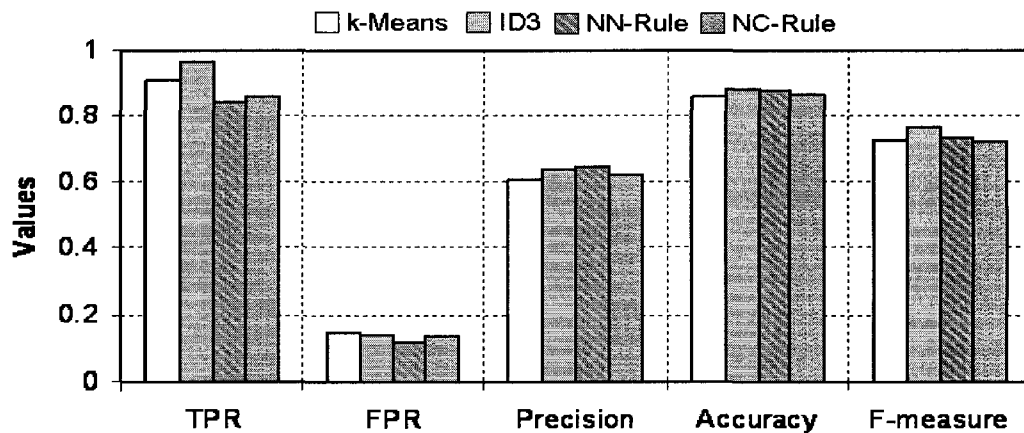


**Figure 6.4 ROC Curves and AUCs of k-Means, ID3, and K-Means+ID3 with NN-Rule and NC-Rule over the NAD-1999 test dataset.**

### **6.3 Results on the NAD-2000 Dataset**

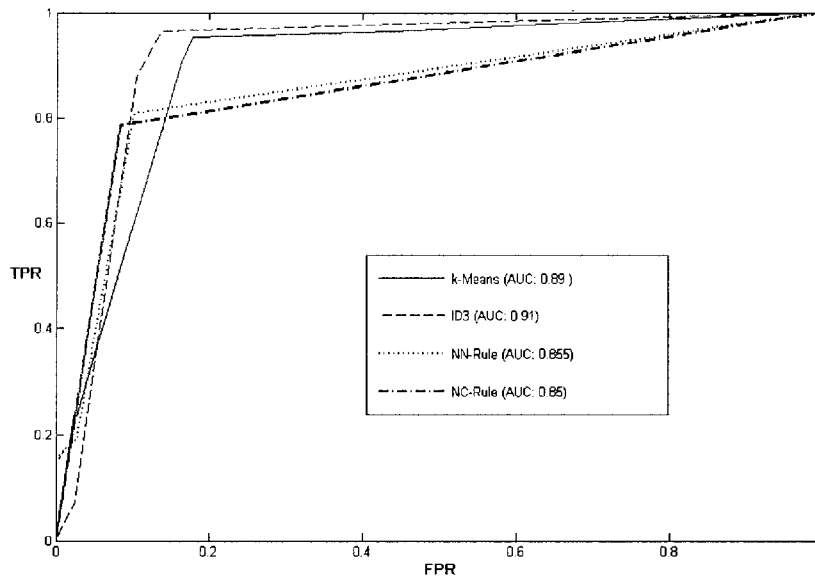
Figure 6.5 illustrates the performance of the k-Means, the ID3, and the K-Means+ID3 methods averaged over 10 trials for k-Means and K-Means+ID3. For the NAD-2000 datasets, the  $k$  value of the k-Means was set to 10. For the ID3 algorithm, the training space was discretized into 15 equal-width intervals. For the K-Means+ID3 cascading algorithm, we set the value of  $k$  to 10 and discretized the data into 15 equal-width intervals.

Figure 6.5 shows that (i) the K-Means+ID3 cascading with NN combination has better performance than the k-Means and ID3 in terms of FPR and Precision, (ii) the TPR of the K-Means+ID3 cascading is less than the k-Means and ID3 methods, and (iii) the accuracy of the K-Means+ID3 is similar to the k-Means and ID3 methods.



**Figure 6.5 Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the NAD-2000 test dataset.**

Figure 6.6 shows the ROC curves and AUC values of the k-Means, ID3 and K-Means+ID3 methods over NAD-2000 test dataset. The ROC curves for the K-Means+ID3 and k-Means methods were plotted for the trial with the AUC value closest to the mean TPR values in Figure 6.5. The ROC curves for the k-Means, and ID3 methods show better performance than the K-Means+ID3 cascading algorithm over the NAD-2000 datasets.

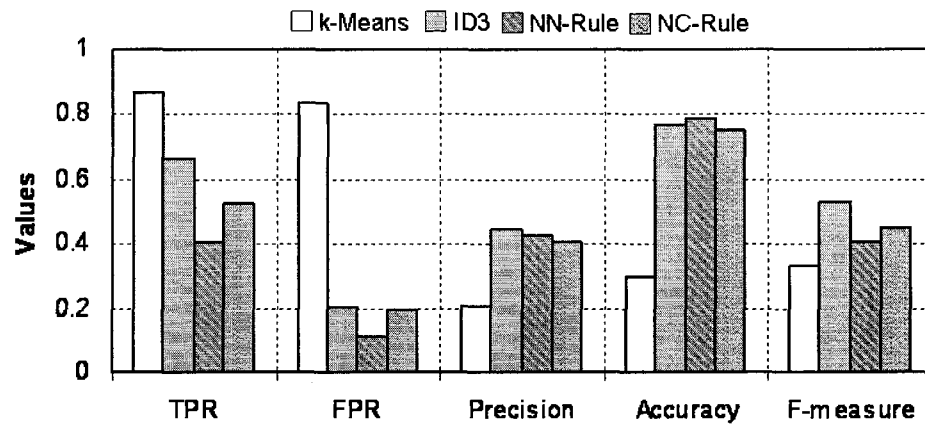


**Figure 6.6 ROC Curves and AUCs of k-Means, ID3, and K-Means+ID3 methods over the NAD-2000 test dataset.**

#### **6.4 Results on the Duffing Equation Dataset**

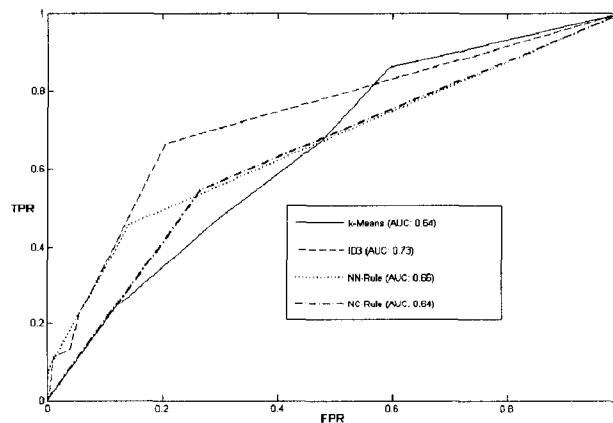
Figure 6.7 illustrates the performance of the k-Means, the ID3, and the K-Means+ID3 methods averaged over 10 trials for k-Means and K-Means+ID3. For the DED datasets, the  $k$  value for the k-Means was set to 5 clusters. For the ID3, the training space was discretized into 45 equal-width intervals. For the K-Means+ID3 method, we set the value of  $k$  to 5 and discretized the data into 45 equal-width intervals.

Figure 6.7 shows that (i) the K-Means+ID3 cascading with NC and NN combination has better performance than the k-Means in terms of FPR, precision, and accuracy, (ii) the F-measure of the K-Means+ID3 cascading is in-between the k-Means and the ID3, (iii) the TPR of the k-Means+ID3 is less than the k-Means and ID3 methods.



**Figure 6.7 Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the DED test dataset.**

Figure 6.8 shows the ROC curves and AUC values of the k-Means, ID3 and K-Means+ID3 methods over DED. The ROC curves for K-Means+ID3 and k-Means algorithm were plotted for the trial with the AUC value that is closest to the mean TPR values shown in Figure 6.7. The ROC curve for the K-Means+ID3 cascading with NC and NN combinations is in-between the k-Means and the ID3 methods over the DED test datasets.

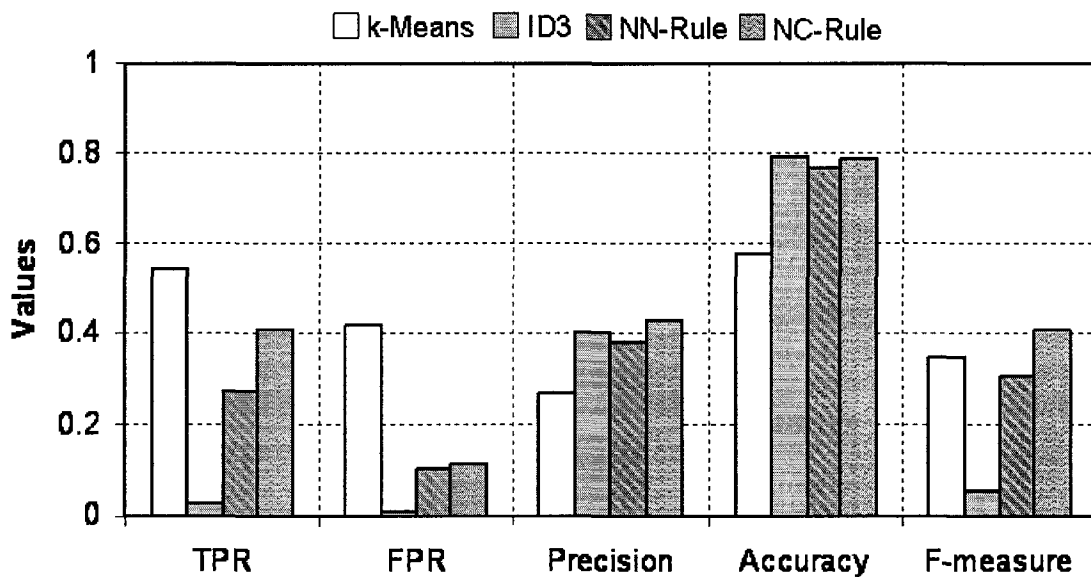


**Figure 6.8 ROC Curves and AUCs of k-Means, ID3 and K-Means+ID3 methods over the DED test dataset.**

### 6.5 Results on the Mechanical Systems Dataset

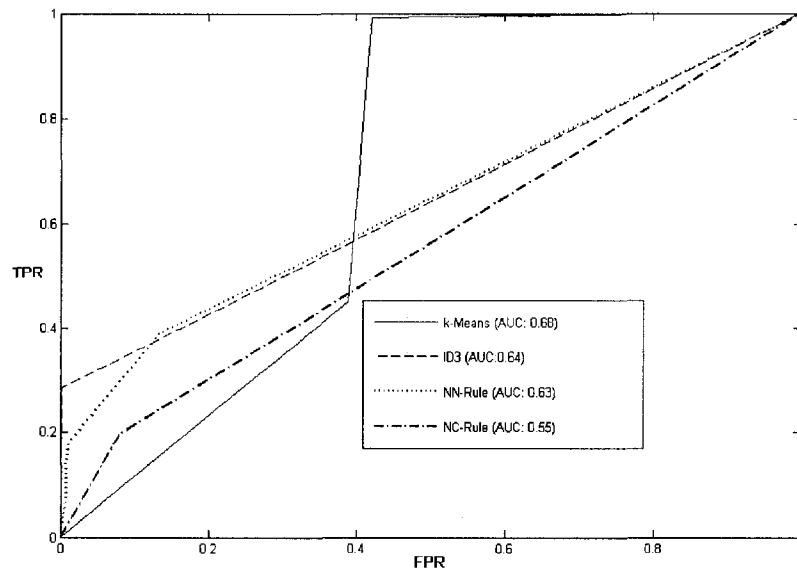
Figure 6.9 illustrates the performance of the k-Means, the ID3, and the K-Means+ID3 algorithms averaged over 10 trials for k-Means and K-Means+ID3. For the MSD datasets, the  $k$  value of the k-Means was set to 5. For the ID3 method, the training space was discretized into 65 equal-width intervals. For the K-Means+ID3 method, we set the value of  $k$  to 5 and discretize the data into 65 equal-width intervals.

Figure 6.9 shows that: (i) K-Means+ID3 with NC combination has better performance than the k-Means in terms of FPR, precision, and F-measure, and (ii) the precision, accuracy, and the F-measure of the K-Means+ID3 with NC combination is higher than the k-Means method.



**Figure 6.9 Performance of the k-Means, the ID3 decision tree, and the K-Means+ID3 method with Nearest Neighbor (NN-Rule) and Nearest Consensus (NC-Rule) combination rules over the MSD test dataset.**

Figure 6.10 shows the ROC curves and AUCs of the k-Means, ID3 and K-Means+ID3 methods over MSD. The ROC curves for K-Means+ID3 and k-Means methods were plotted for the trial with the AUC value that is closest to the mean TPR values in Figure 6.9. The ROC curves for the K-Means+ID3 with NN combination shows a TPR rate as high as 0.98 at a FPR of 0.4 over the MSD test dataset.



**Figure 6.10 ROC Curves and AUCs of k-Means, ID3 and K-Means+ID3 methods over the MSD test dataset.**



## CHAPTER 7

### ANOMALY DETECTION USING DEPENDENCE TREES

In this chapter we present the details of dependence tree based anomaly detection method for detecting attacks on a computer network system. The goal of the dependence tree based anomaly detection method is to build dependence trees that achieve high classification accuracy in detecting five types of network traffic instances originating from: (1) normal traffic, (2) denial-of-service attacks, (3) probing attacks, (4) user-to-root attacks, and (5) remote-to-login attacks. The two major motivations for using dependence trees for detecting network attacks is as follows:

- Dependence trees have an advantage of making the classification models more *explicit* with regard to features and their relationships. Such explicit representations of relationships between features facilitate the ensuing steps of network forensic analysis and vulnerability inspections over the feature space, which are inevitably performed for effective corrective actions.
- Dependence trees, being probabilistic by nature, assign probability scores indicating the “degree” to which a network traffic instance belongs to particular type of attack. Such quantitative assignments of scores to network traffic

instances will assist human experts and network administrators in culling the false positives generated by the attack detection system.

Next, we present the problem formulation for network attack detection using the dependence tree based anomaly detection method.

### 7.1 Problem Formulation

Let  $X = (x_1, x_2, \dots, x_n)$  denote an  $n$ -dimensional feature vector. The feature vector  $X$  represents a set of  $n$  measurements recorded over a computer network (e.g., type of service, protocol, number of source bytes, etc.). Let  $W = \{\omega_1, \omega_2, \dots, \omega_r\}$  denote a set of  $r$  classes. The set  $W$  represents five types of network traffic instances, i.e., normal, denial-of-service attack, probing attack, user-to-root attack, and remote-to-login attack. We assign an optimal label  $\omega^* \in W$  to  $X$  using the Bayes classification rule, given by

$$P(\omega^* | X) = \max_{k=1:r} \{P(\omega_k | X)\} \quad \text{Equation 7.1}$$

where  $P(\omega_k | X)$  is the posterior probability of the class  $\omega_k \in W$  given the feature vector  $X$ . Using Bayes formula [13], the posterior probability can be expressed as a function of class-conditional probability  $P(X | \omega_k)$  by

$$P(\omega_k | X) = \frac{P(\omega_k)P(X | \omega_k)}{\sum_{k=1}^r P(\omega_k)P(X | \omega_k)}. \quad \text{Equation 7.2}$$

where  $P(\omega_k)$  is the prior probability of class  $\omega_k$  and  $\sum_{k=1}^r P(\omega_k)P(X | \omega_k)$  is the normalization factor that scales the posterior probability between 0 and 1. Through Equation 7.2, the problem of classifying  $X$  into one of the  $r$  classes becomes one of estimating the class-conditional probability function  $P(X | \omega_k)$ . However, estimating

$P(X | \omega_k)$  involves computations of (exponential) order  $N^m$ , where  $m$  is the number of features and  $N$  is the number of unique values in each feature. The exponential complexity makes estimating  $P(X | \omega_k)$  infeasible when dealing with a large number of features, which are typically available for the anomaly detection problem. Therefore, we estimate the class-conditional probability with a product of second-order joint probability distributions using dependence tree approximation. The dependence tree approximation requires at most  $(N^2 \cdot m^2)$  computations, which mean far fewer computations than  $N^m$  even for moderate values of  $N$  and  $m$ . Here, it is important to note that we build a number of dependence trees equal to that of the classes available (i.e., five classes in our case).

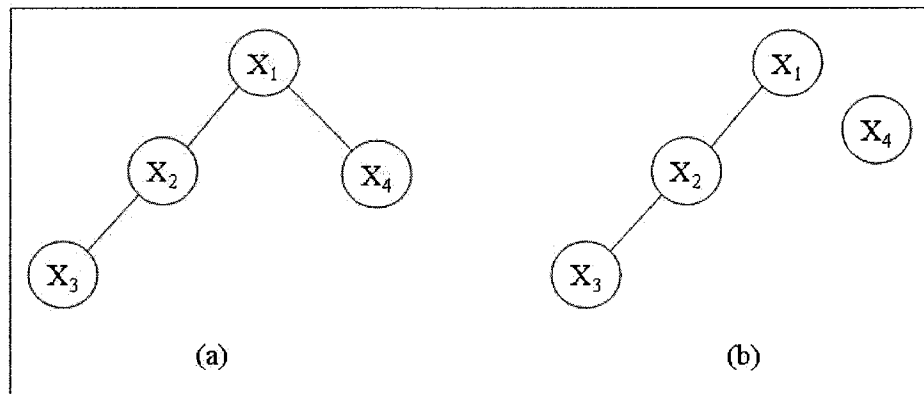
## 7.2 Dependence Trees

Chow and Liu [31] first introduced dependence trees to approximate an  $n^{\text{th}}$  order discrete joint probability distribution using a product of second order discrete joint probability distributions. Let  $X = (x_1, x_2, \dots, x_n)$  denote an  $n$ -dimensional discrete random feature vector. Let  $P(X) = P(x_1, x_2, \dots, x_n)$  be the joint distribution of the feature vector. In dependence tree approximation, the joint probability distribution  $P(x_1, x_2, \dots, x_n)$  is approximated by  $\hat{P}(X)$  as

$$P(X) \approx \hat{P}(X) = \prod_{i=1}^n P(x_{m_i} | x_{m_{j(i)}}), \quad 0 \leq j(i) < i, \quad \text{Equation 7.3}$$

where  $(m_1, m_2, \dots, m_n)$  is an unknown permutation of integers  $1, 2, \dots, n$ ,  $P(x_{m_i} | x_{m_{j(i)}})$  is a component probability in which each feature  $x_{m_i}$  is conditioned on at most one variable  $x_{m_{j(i)}}$ , and the component probability of the form  $P(x_{m_i} | x_o)$  is by definition equal to  $P(x_{m_i})$ .

Figure 7.1 gives examples of two dependence trees containing four features  $(x_1, x_2, x_3, x_4)$ . In Figure 7.1(a), the joint probability distribution  $P(x_1, x_2, x_3, x_4)$  is approximated by  $P(x_1)P(x_2 | x_1)P(x_3 | x_2)P(x_4 | x_1)$ . In Figure 7.1(b),  $P(x_1, x_2, x_3, x_4)$  is approximated by  $P(x_1)P(x_2 | x_1)P(x_3 | x_2)P(x_4)$ . Note that the dependence tree in (b) illustrates a case where there can be more than one independent component, i.e.,  $P(x_1)$  and  $P(x_4)$ .



**Figure 7.1** Dependence trees approximating the joint probability distribution  $P(x_1, x_2, x_3, x_4)$ . The dependence tree in (a) approximates  $P(x_1, x_2, x_3, x_4)$  as  $P(x_1)P(x_2|x_1)P(x_3|x_2)P(x_4|x_1)$  and the dependence tree in (b) approximates  $P(x_1, x_2, x_3, x_4)$  as  $P(x_1)P(x_2|x_1)P(x_3|x_2)P(x_4)$ .

### 7.3 Steps for Building Dependence Trees

Let  $X = (x_1, x_2, \dots, x_n)$  be an  $n$ -dimensional discrete feature vector. Let  $W = \{\omega_1, \omega_2, \dots, \omega_r\}$  denote a set of  $r$  classes. Let  $P(x_1, x_2, \dots, x_n)$  be the joint distribution to be approximated. There are five steps to build a dependence tree to approximate  $P(x_1, x_2, \dots, x_n)$ .

STEP 1) Find the mutual information  $I(x_i, x_j)$  between all pairs of features, where  $(x_i, x_j)$  are any two features in  $X = (x_1, x_2, \dots, x_n)$  and  $i \neq j$ . The mutual information between the feature pair  $(x_i, x_j)$  is defined as

$$I(x_i, x_j) = \sum_{x_i} \sum_{x_j} P(x_i, x_j) \log \left( \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \right)$$

STEP 2) Build a complete undirected graph  $G$  with the features  $(x_1, x_2, \dots, x_n)$  as vertices and the mutual information between the vertices as edge weights.

STEP 3) Use Kruskal's algorithm [32] to find a maximum spanning tree in  $G$ .

STEP 4) In the maximum spanning tree, choose any node as a root node and set the direction of all edges outwards, pointing towards the root node.

STEP 5) Repeat STEP 1 through STEP 4 for each class in  $W$ .

#### 7.4 Optimality of Dependence Tree Approximation

Chow and Liu, in [31], have shown that the dependence tree with the maximum edge weights (i.e., mutual information) gives the optimal second order approximation of  $P(X) = P(x_1, \dots, x_n)$ , the true joint probability distribution of an  $n$ -dimensional feature vector  $X$ . A brief overview of their proof follows.

Let  $\hat{P}(X)$  be any second order dependence tree approximation of  $P(X) = P(x_1, \dots, x_n)$ . Then, the closeness of approximation between the probabilities  $P(X)$  and  $\hat{P}(X)$  is quantified by the Kullback-Liebler divergence measure [15], given as

$$KL(P, \hat{P}) = \sum_X P(X) \log \left( \frac{P(X)}{\hat{P}(X)} \right). \quad \text{Equation 7.4}$$

With Equation 7.4, the problem of finding the optimal second order dependence tree approximation to  $P(X)$  is transformed into the problem of finding  $\hat{P}(X)$  which minimizes  $KL(P, \hat{P})$ . On replacing  $\hat{P}(X)$  with the dependence tree approximation given in Equation 7.3, the Kullback-Leibler distance between  $P(X)$  and  $\hat{P}(X)$  becomes

$$\begin{aligned} KL(P, \hat{P}) &= -\sum_{i=1}^n \sum_{x_{m_i}, x_{m_{j(i)}}} P(x_{m_i}, x_{m_{j(i)}}) \log \left( \frac{P(x_{m_i}, x_{m_{j(i)}})}{P(x_{m_i})P(x_{m_{j(i)}})} \right) + \sum_{i=1}^n H(x_{m_i}) - H(X) \\ &= -\sum_{i=1}^n I(x_{m_i}, x_{m_{j(i)}}) + \sum_{i=1}^n H(x_{m_i}) - H(X), \end{aligned} \quad \text{Equation 7.5}$$

where  $H(x_{m_i}) = -\sum_{x_{m_i}} P(x_{m_i}) \log P(x_{m_i})$  and  $H(X) = -\sum_X P(X) \log P(X)$ . Because  $H(x_{m_i})$

and  $H(X)$  are independent of the dependence tree approximation, minimizing  $KL(P, \hat{P})$

is equivalent to maximizing the total mutual information  $\sum_{i=1}^n I(x_{m_i}, x_{m_{j(i)}})$ . Thus, the

problem of finding the optimal dependence tree approximation  $\hat{P}(X)$  is transformed to that of finding a dependence tree with maximum total branch weight.

## 7.5 Estimating Mutual Information

Estimating mutual information between pairs of features involves estimating the marginal and joint probabilities of features. To estimate the marginal and joint probabilities, we use relative frequencies derived from the training samples. Let  $P(x_i, x_j)$  be the joint probability distribution of two features  $x_i$  and  $x_j$ . The value of  $P(x_i, x_j)$  is calculated as

$$P(x_i = u, x_j = v) = \frac{F(x_i = u, x_j = v)}{T},$$

where  $F(x_i = u, x_j = v)$  denotes the number of training samples whose feature  $x_i$  is equal to  $u$  and whose feature  $x_j$  is equal to  $v$ .  $T$  denotes the total number of training samples. Similarly, the marginal distributions of the features  $x_i$  and  $x_j$ , that is,  $P(x_i)$  and  $P(x_j)$  are calculated as

$$P(x_i = u) = \frac{F(x_i = u)}{T} \text{ and } P(x_j = v) = \frac{F(x_j = v)}{T}$$

where  $F(x_i = u)$  is the number of training samples whose feature  $x_i$  is equal to  $u$  and  $F(x_j = v)$  is the number of training samples whose feature  $x_j$  is equal to  $v$ .

## CHAPTER 8

### RELATIONSHIP BETWEEN DEPENDENCE

### TREE CLASSIFICATION ERROR AND

### BAYES ERROR RATE

Let  $X = (X_1, \dots, X_n)$  denote an  $n$ -dimensional discrete random feature vector. Let  $W = \{\omega_1, \dots, \omega_r\}$ , be a discrete random variable whose values are the class labels. Let  $P(x|\omega)$  be the conditional distribution of  $X$  given  $W$ , where  $x = (x_1, \dots, x_n)$  is a value of the feature vector  $X$  and  $\omega$  is a value of  $W$ . In Chow and Liu's dependence tree approximation, the probability distribution  $P(x|\omega)$  is approximated by  $\hat{P}(x|\omega)$  as

$$P(x|\omega) \approx \hat{P}(x|\omega) = \prod_{i=1}^n P(x_{m_i} | x_{m_{j(i)}}, \omega), \quad 0 \leq j(i) < i, \quad \text{Equation 8.1}$$

where  $(m_1, \dots, m_n)$  is an unknown permutation of integers  $1, 2, \dots, n$ ,  $P(x_{m_i} | x_{m_{j(i)}}, \omega)$  is a component probability in which each variable  $x_{m_i}$  is conditioned on at most one variable  $x_{m_{j(i)}}$ , and the component probability of the form  $P(x_i | x_o, \omega)$  is by definition equal to  $P(x_i | \omega)$ . The unknown permutation is obtained using Kruskal's algorithm [32], which finds the spanning tree with maximum pairwise mutual information between the features. For notational simplicity, we will hereafter omit the subscript  $m$  of each variable and represent, for example,  $x_{m_i}$  as  $x_i$ .



Hellman and Raviv [33] proved that an upper bound on Bayes error rate “ $\sigma_e$ ” is  $\frac{1}{2}H(\omega|X)$ , where  $H(\omega|X)$  is the conditional entropy of class  $\omega$  given the  $n$ -dimensional feature vector  $X$ . Wong and Poon, in [34], extended Hellman and Raviv’s result (see [35] for tighter bounds on the Bayes error rate) and showed that, under certain assumptions, Chow and Liu’s dependence tree approximation procedure can be derived by minimizing the upper bound of the Bayes error rate. Wong and Poon’s result comes from Equation (5) in their paper [34], which expands the entropy function  $H(\omega|X)$  and replaces  $P(x|\omega)$  with probability distribution  $\hat{P}(x|\omega)$  using the dependence tree approximation. The equation appeared as

$$\hat{H}(\omega|X) = H(\omega) - H(X) - \sum_{\omega} P(\omega) \sum_{i=1}^n I_{\omega}(X_i, X_{j(i)}) - \sum_{\omega} P(\omega) \sum_{i=1}^n H_{\omega}(X_i) \quad \text{Equation 8.2}$$

where

$$H(\omega) = -\sum_{\omega} P(\omega) \log P(\omega),$$

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

$$I_{\omega}(X_i, X_{j(i)}) = -\sum_{x_i, x_{j(i)}} P(x_i, x_{j(i)} | \omega) \log \frac{P(x_i, x_{j(i)} | \omega)}{P(x_i | \omega) P(x_{j(i)} | \omega)}, \text{ and}$$

$$H_{\omega}(X_i) = -\sum_{x_i} P(x_i | \omega) \log P(x_i | \omega).$$

The correct expansion of the conditional entropy function  $\hat{H}(\omega|X)$  (derived in the next section) and is given as:

$$\hat{H}(\omega|X) = H(\omega) - H(X) - \sum_{\omega} P(\omega) \sum_{i=1, j(i) \neq 0}^n I_{\omega}(X_i, X_{j(i)}) + \sum_{\omega} P(\omega) \sum_{i=1}^n H_{\omega}(X_i). \quad \text{Equation 8.3}$$

Equation 8.3 corrects Equation 8.2 by reversing the sign of the term  $\sum_{\omega} P(\omega) \sum_{i=1}^n H_{\omega}(X_i)$ . Though this correction appears as a minor issue, it invalidates the misleading idea purported by Equation 8.2 that every component probability in the dependence tree approximation decreases the value of  $\hat{H}(\omega|X)$  thereby reducing the upper bound on the Bayes error rate. The corrected equation (Equation 8.3) shows that each component probability in the dependence tree approximation, whether in the form of  $P(x_i | x_{j(i)}, \omega)$ ,  $j(i) \neq 0$ , or  $P(x_i | x_0, \omega)$ , adds  $\sum_{\omega} P(\omega) H_{\omega}(X_i)$  to  $\hat{H}(\omega|X)$  and does not necessarily contribute toward decreasing the upper bound on Bayes error rate. Therefore, caution is advised when selecting component probabilities for dependence tree approximation.

Below, we give two conditions to guarantee that every component probability in the dependence tree approximation decreases the value of  $\hat{H}(\omega|X)$ , thereby decreasing the upper bound on Bayes error rate.

**Condition 1:** *In a dependence tree approximation, for each component probability of the form  $P(x_i | x_{j(i)}, \omega)$ ,  $j(i) \neq 0$ ,  $\sum_{\omega} P(\omega) I_{\omega}(X_i, X_{j(i)})$  should be greater than*

$$\sum_{\omega} P(\omega) H_{\omega}(X_i).$$

Condition 1 follows from expansion in Equation 8.9 in the derivation (in the next section) and concerns with component probabilities of the form  $P(x_i | x_{j(i)}, \omega)$ ,  $j(i) \neq 0$ , in the dependence tree approximation. We explain Condition 1 with an example. Let  $X = (X_1, X_2, X_3)$  be a three-dimensional discrete random feature vector. Let

$P(X_1|\omega)P(X_2|X_1,\omega)P(X_3|\omega)$  be the dependence tree approximation of  $P(X|\omega)$ . In this dependence tree approximation, there is one component probability of the form  $P(x_i|x_{j(i)},\omega)$ , i.e.,  $P(X_2|X_1,\omega)$ . Expansion in Equation 8.9 shows that each component probability of the form  $P(x_i|x_{j(i)},\omega)$  adds  $\sum_{\omega} P(\omega)H_{\omega}(X_i)$  to  $\hat{H}(\omega|X)$ .

Therefore,  $P(X_2|X_1,\omega)$  adds  $\sum_{\omega} P(\omega)H_{\omega}(X_2)$  to  $\hat{H}(\omega|X)$ . However, if

$\sum_{\omega} P(\omega)I_{\omega}(X_2, X_1)$  is greater than  $\sum_{\omega} P(\omega)H_{\omega}(X_2)$ , then from expansion in Equation 8.9

we see that the presence of component probability  $P(X_2|X_1,\omega)$  in the dependence tree approximation decreases the value of  $\hat{H}(\omega|X)$ , thereby decreasing the bound on Bayes error rate.

**Condition 2:** *In a dependence tree approximation, for each component probability of the form  $P(x_i|x_0,\omega)$ ,  $0 < i \leq n$ , there must be a nonempty set  $l_i, |l_i| \leq n$ , of component probabilities of the form  $P(x_s|x_i,\omega)$ ,  $0 < s \leq n$ , so that  $\sum_{\omega} P(\omega) \sum_n I_{\omega}(X_s, X_i)$  is greater than  $\sum_{\omega} P(\omega)H_{\omega}(X_i)$ .*

Condition 2 follows from Equation 8.10 in the next section and concerns with component probabilities of the form  $P(x_i|x_0,\omega)$ . We explain Condition 2 with an example. Let  $X = (X_1, X_2, X_3, X_4)$  be a four-dimensional discrete random feature vector. Let  $P(X_1|\omega)P(X_2|X_1,\omega)P(X_3|X_1,\omega)P(X_4|\omega)$  be the dependence tree approximation of  $P(X|\omega)$ . In this dependence tree (or more precisely, dependence forest) approximation, there are two component probabilities of the form  $P(x_i|x_0,\omega)$ ,

i.e.,  $P(X_1 | \omega)$  and  $P(X_4 | \omega)$ . Equation 8.10 shows that  $P(X_1 | \omega)$  and  $P(X_4 | \omega)$  add  $\sum_{\omega} P(\omega)H_{\omega}(X_1)$  and  $\sum_{\omega} P(\omega)H_{\omega}(X_4)$  to  $\hat{H}(\omega | X)$ . Now, consider the component probability  $P(X_1 | \omega)$ . From Condition 2,  $I_1$  contains all the component probabilities conditioned on  $X_1$ , i.e.,  $I_1 = \{P(X_2 | X_1, \omega), P(X_3 | X_1, \omega)\}$ . If  $\sum_{\omega} P(\omega)[I_{\omega}(X_2, X_1) + I_{\omega}(X_3, X_1)] > \sum_{\omega} P(\omega)H_{\omega}(X_1)$ , then from Equation 8.10 we see that the presence of the variable  $X_1$  decreases  $\hat{H}(\omega | X)$ , thereby decreasing the upper bound on the Bayes error rate. However, the component probability  $P(X_4 | \omega)$  does not satisfy Condition 2 because  $I_4$  is an empty set. Therefore, the presence of  $P(X_4 | \omega)$  certainly increases  $\hat{H}(\omega | X)$ , thereby increasing the upper bound on the Bayes error rate. Consequently, the variable  $X_4$  may be omitted when approximating  $P(X | \omega)$ .

### **8.1 Derivation Relating Bayes Error Rate To Dependence Tree Classification Error**

It is known [15] that

$$H(\omega | X) = H(\omega) - I(X, \omega). \quad \text{Equation 8.4}$$

Using the definition of mutual information [5],  $I(X, \omega)$  in Equation 8.4 can be expanded as

$$H(\omega | X) = H(\omega) - \sum_{x, \omega} P(x, \omega) \log P(x, \omega) + \sum_{x, \omega} P(x, \omega) \log P(x) + \sum_{x, \omega} P(x, \omega) \log P(\omega) \quad \text{Equation 8.5}$$

By the definition of entropy

$$\sum_{x, \omega} P(x, \omega) \log P(x) = \sum_x P(x) \log P(x) = -H(X) \quad \text{and}$$

$$\sum_{x,\omega} P(x,\omega)\log P(\omega) = \sum_{\omega} P(\omega)\log P(\omega) = -H(\omega).$$

Therefore, Equation 8.5 can be written as

$$H(\omega|X) = -H(X) - \sum_{x,\omega} P(x,\omega)\log P(x,\omega) = -H(X) - \sum_{\omega} P(\omega)\sum_x P(x|\omega)\log(P(x|\omega)P(\omega))$$

Equation 8.6

Using dependence tree approximation in Equation 8.1,  $\log(P(x|\omega)P(\omega))$  in Equation 8.6

is replaced by  $\log(\hat{P}(x|\omega)P(\omega))$ , so that

$$\begin{aligned} \hat{H}(\omega|X) &= -H(X) + H(\omega) - \sum_{\omega} P(\omega)\sum_x P(x|\omega)\sum_{i=1}^n \log P(x_i|x_{j(i)},\omega), \quad 0 \leq j(i) < i \\ &= -H(X) + H(\omega) - \underbrace{\sum_{\omega} P(\omega)\sum_x P(x|\omega)\sum_{i=1, j(i) \neq 0}^n \log P(x_i|x_{j(i)},\omega)}_{\text{TERM I}} \\ &\quad - \underbrace{\sum_{\omega} P(\omega)\sum_x P(x|\omega)\sum_{i=1, j(i)=0}^n \log P(x_i|x_{j(i)},\omega)}_{\text{TERM 2}}. \end{aligned}$$

Equation 8.7

Term I (sign included) in Equation 8.7 contains the component probabilities of the form  $P(x_i|x_{j(i)},\omega)$ ,  $j(i) < i$ , and  $j(i) \neq 0$ . Term II (sign included) contains the remaining component probabilities of the form  $P(x_i|x_0,\omega) = P(x_i|\omega)$ . Term I can be expanded as

$$- \sum_{\omega} P(\omega)\sum_x P(x|\omega)\sum_{i=1, j(i) \neq 0}^n \log \frac{P(x_i, x_{j(i)}|\omega)}{P(x_i|\omega)P(x_{j(i)}|\omega)} - \sum_{\omega} P(\omega)\sum_x P(x|\omega)\sum_{i=1, j(i) \neq 0}^n \log P(x_i|\omega).$$

Equation 8.8

Since  $P(x_i, x_{j(i)}|\omega)$  and  $P(x_i|\omega)$  are components (marginal distributions) of  $P(x|\omega)$ ,

we know that

$$\sum_x P(x|\omega)\log \frac{P(x_i, x_{j(i)}|\omega)}{P(x_i|\omega)P(x_{j(i)}|\omega)} = \sum_{x_i, x_{j(i)}} P(x_i, x_{j(i)}|\omega)\log \frac{P(x_i, x_{j(i)}|\omega)}{P(x_i|\omega)P(x_{j(i)}|\omega)} \text{ and}$$

$$\sum_x P(x|\omega) \log P(x_i|\omega) = \sum_{x_i} P(x_i|\omega) \log P(x_i|\omega).$$

Therefore, the expansion in Equation 8.8 can be rewritten as

$$-\sum_{\omega} P(\omega) \sum_{i=1, j(i) \neq 0}^n I_{\omega}(X_i, X_{j(i)}) + \sum_{\omega} P(\omega) \sum_{i=1, j(i) \neq 0}^n H_{\omega}(X_i). \quad \text{Equation 8.9}$$

Expansion in Equation 8.9 shows that each component probability of the form  $P(x_i | x_{j(i)}, \omega)$ ,  $j(i) \neq 0$ , adds  $\sum_{\omega} P(\omega) H_{\omega}(X_i)$  to  $\hat{H}(\omega | X)$ . Now, consider Term II in

Equation 8.7. Let there be  $K$  component probabilities of the form

$P(x_i | x_0, \omega) = P(x_i | \omega)$ . Then, Term II can be written as

$$-\sum_{\omega} P(\omega) \sum_x P(x|\omega) \sum_{i=1}^K \log P(x_i|\omega) = -\sum_{\omega} P(\omega) \sum_{i=1}^K \sum_{x_i} P(x_i|\omega) \log P(x_i|\omega) = \sum_{\omega} P(\omega) \sum_{i=1}^K H_{\omega}(X_i)$$

$$\text{Equation 8.10}$$

where  $K \geq 1$  and  $K \leq n$  from the definition of dependence tree approximation in

Equation 8.1. Equation 8.10 shows that each component probability of the form

$P(x_i | x_0, \omega)$  adds  $\sum_{\omega} P(\omega) H_{\omega}(X_i)$  to  $\hat{H}(\omega | X)$ . By substituting Equation 8.9 and

Equation 8.10 for Term I and Term II respectively, Equation 8.7 becomes

$$\hat{H}(\omega | X) = H(\omega) - H(X) - \sum_{\omega} P(\omega) \sum_{i=1, j(i) \neq 0}^n I_{\omega}(X_i, X_{j(i)}) + \sum_{\omega} P(\omega) \sum_{i=1}^n H_{\omega}(X_i).$$

## CHAPTER 9

### MAXIMUM RELEVANCE MINIMUM

### REDUNDANCY FEATURE

### SELECTION

Feature selection (in the context of classification) is the process of identifying the most characterizing features that minimize classification error [36]. Let dataset  $D$  contain  $M$  samples with  $n$ -dimensional features, i.e.,  $D = \{X_1, X_2, \dots, X_M\}$  and  $X_i = (x_1, x_2, \dots, x_n)$ . Each sample in  $D$  belongs to one of the classes in  $\mathcal{W} = \{\omega_1, \omega_2, \dots, \omega_r\}$ . The feature selection problem is to find a subset of  $k$  features that minimize classification error. One of the most popular methods of feature selection, known as the Maximum Relevance feature selection [37], selects features with the highest relevance to the target class  $\mathcal{W}$ . In Maximum Relevance feature selection, “relevance” is usually characterized using correlation or mutual information between features and the classes. Mutual information is preferred over correlation because correlation captures only linear dependencies between features and classes where as mutual information captures linear as well as nonlinear dependencies (see [16]). A detailed investigation of the advantages of mutual information over correlation is given in [38], [39], and [40]. Next, we briefly discuss Maximum Relevance feature selection.

### 9.1 Maximum Relevance Feature Selection

In Maximum Relevance feature selection, the objective is to identify a subset of  $k$  features among a set of  $n$  features such that the mutual information between the  $k$  features and the class is maximized. In other words, Maximum Relevance feature selection maximizes

$$I(X', W) = \sum_X \sum_W P(X', W) \log \left( \frac{P(X', W)}{P(X')P(W)} \right)$$

where  $X'$  is a feature vector of size  $k$ . Identifying maximally relevant features by calculating  $I(X', W)$  is computationally prohibitive, especially when there are a large number of features, because (1) computing  $I(X', W)$  requires estimating higher order probability terms, which require a minimum of  $M \cdot 2^n$  computations assuming that there are  $M$  training samples and that each of the  $n$  features has exactly two values and (2) the consideration of all possible subsets of features requires computing  $I(X', W)$  for  $\binom{n}{k}$  number of times. One is therefore forced to choose approximations to Maximum Relevance feature selection. A possible way to approximate Maximum Relevance feature selection is to calculate the mutual information between the individual features and class and then, incrementally select  $k$  features with the highest relevance, so that the selected  $k$  features maximize

$$\sum_{i=1}^k \sum_{x_i} \sum_W P(x_i, W) \log \left( \frac{P(x_i, W)}{P(x_i)P(W)} \right).$$

However, it is likely that the  $k$  features selected through Maximum Relevance feature selection may contain large dependencies. That means that features may highly



depend on each other, and therefore removing any one of them may not significantly change the overall classification error. To eliminate such redundancy but at the same time retain relevance of the features to the classes, Maximum Relevance Minimum Redundancy feature selection is used. A brief description of Maximum Relevance Minimum Redundancy feature selection follows.

## 9.2 Maximum Relevance Minimum

### Redundancy (MRMR)

Let  $F$  be a set containing  $n$  features  $\{x_1, x_2, \dots, x_n\}$ . Let  $S (S \subseteq F)$  be a set of  $k-1$  features which jointly have the largest dependency on the class variable  $W$ . The objective of Maximum Relevance Minimum Redundancy (MRMR) is to add the  $k^{\text{th}}$  feature  $x_i$  from  $F$  into  $S$  so that  $x_i$  maximizes

$$I(x_i, W) - \frac{1}{|S|} \sum_{x_j \in S} I(x_i, x_j), \quad i \in F \quad \text{Equation 9.1}$$

From Equation 9.1, it is clear that a feature is selected from  $F$  not only if it maximizes the mutual information with the class variable  $W$ , but also if it is unpredictable by the current set of already selected features in  $S$ . The criterion in Equation 9.1 can be applied incrementally to select a set of  $k$  features from a set of  $n$  features that optimally characterize the class variable  $W$ . The steps for performing MRMR feature selection are:

- 1) (Initialization Step) Set  $F$  to contain the initial set of  $n$  features;  $S$  containing the final set of  $k$  features is initialized to empty.
- 2) Compute the mutual information between  $W$  and each of the individual features in  $F$ .

3) (Selection of the first feature) Select the first feature  $x_i$  that maximizes  $I(x_i, W)$ ;

Remove  $x_i$  from  $F$  and add  $x_i$  to  $S$ .

4) Repeat until  $|S| = k$

a. Compute the mutual information between pairs of variables  $I(x_i, x_j)$  such that  $x_i \in F$  and  $x_j \in S$ , if it is not already available.

b. (Selection of the next feature) Select feature  $x_i \in F$ , which maximizes the

criterion  $I(x_i, W) - \frac{1}{|S|} \sum_{x_j \in S} I(x_i, x_j)$ ; add  $x_i$  to  $S$ ; remove  $x_i$  from  $F$ .

In several studies (see [16] and [37]), the above procedure for MRMR feature selection has been empirically shown to decrease classification error and improve classification results. However, until now, the criterion used in MRMR feature selection (Equation 9.1) has been used as heuristic, without formal proof showing how the criterion is instrumental in reducing classification error. In the next section, we present a set of derivations to show that the MRMR feature selection procedure, under some assumptions, gives an upper bound on the Bayes error rate.

### 9.3 Relationship Between MRMR Feature

#### Selection and Bayes Error Rate

In this section we derive the relationship between MRMR feature selection and Bayes error rate and show that the criterion of MRMR feature selection (Equation 9.1) is an approximation to an upper bound of Bayes error rate. Let  $X = (X_1, X_2, \dots, X_n)$  denote an  $n$ -dimensional discrete random feature vector. Let  $W = (\omega_1, \omega_2, \dots, \omega_n)$  be a discrete

random variables whose values are the class labels. Let  $x = (x_1, x_2, \dots, x_n)$  be a value of the feature vector  $X$  and  $\omega$  is a value of  $W$ .

**Assumption 9.1** *The features  $x_1, x_2, \dots, x_{n-1}$  are independent features and  $x_1, x_2, \dots, x_{n-1}$  are conditionally independent given the feature  $x_n$ .*

**Lemma 9.1** *If  $x_1, x_2, \dots, x_{n-1}$  are independent features and if  $x_1, x_2, \dots, x_{n-1}$  are conditionally independent given the feature  $x_n$  (Assumption 8.4.1), then*

$$P(x_n | x_1, x_2, \dots, x_{n-1}) = \frac{\prod_{i=1}^{n-1} P(x_n | x_i)}{[P(x_n)]^{n-2}}.$$

**Proof:**

From Assumption 9.1, it follows that

$$P(x_1, x_2, \dots, x_{n-1}) = P(x_1)P(x_2) \dots P(x_{n-1}) \text{ and} \quad \text{Equation 9.2}$$

$$P(x_1, x_2, \dots, x_{n-1} | x_n) = P(x_1 | x_n)P(x_2 | x_n) \dots P(x_{n-1} | x_n). \quad \text{Equation 9.3}$$

Using Bayes formula, Equation 9.3 can be expanded as

$$P(x_1, x_2, \dots, x_{n-1} | x_n) = \frac{[P(x_n | x_1)P(x_n | x_2) \dots P(x_n | x_{n-1})] [P(x_1)P(x_2) \dots P(x_{n-1})]}{[P(x_n)]^{n-1}}.$$

Equation 9.4

From Bayes formula, we have

$$P(x_n | x_1, x_2, \dots, x_{n-1}) = \frac{P(x_1, x_2, \dots, x_{n-1} | x_n)P(x_n)}{[P(x_1, x_2, \dots, x_{n-1})]} \quad \text{Equation 9.5}$$

Substituting Equation 9.4 in Equation 9.5, we get

$$P(x_n | x_1, x_2, \dots, x_{n-1}) = \frac{[P(x_n | x_1)P(x_n | x_2) \dots P(x_n | x_{n-1})] [P(x_1)P(x_2) \dots P(x_{n-1})]}{[P(x_n)]^{n-2} [P(x_1, x_2, \dots, x_{n-1})]}$$

Equation 9.6

Using Equation 9.1 in Equation 9.6, we get

$$P(x_n | x_1, x_2, \dots, x_{n-1}) = \frac{[P(x_n | x_1)P(x_n | x_2) \dots P(x_n | x_{n-1})]}{[P(x_n)]^{n-2}} = \frac{\prod_{i=1}^{n-1} P(x_n | x_i)}{[P(x_n)]^{n-2}}$$

Thus, Lemma 9.1 is proved.

**Lemma 9.2** *If  $x_1, x_2, \dots, x_{n-1}$  are independent features and if  $x_1, x_2, \dots, x_{n-1}$  are conditionally independent given the feature  $x_n$  (Assumption 9.1), then the conditional entropy function  $H(x_n | x_1, x_2, \dots, x_{n-1})$  is equal to*

$$H(x_n) - \sum_{i=1}^{n-1} I(x_i, x_n),$$

where  $I(x_i, x_n)$  is the mutual information between features  $x_i$  and  $x_n$ .

**Proof:**

It is well known (see [15]) that the conditional entropy function  $H(x_n | x_1, x_2, \dots, x_{n-1})$  can be expanded as

$$H(x_n | x_1, x_2, \dots, x_{n-1}) = - \sum_{x_1, x_2, \dots, x_n} P(x_1, x_2, \dots, x_n) \log P(x_n | x_1, x_2, \dots, x_{n-1}). \quad \text{Equation 9.7}$$

From Lemma 9.1,  $P(x_n | x_1, x_2, \dots, x_{n-1})$  in Equation 9.7 can be replaced by

$$\frac{\prod_{i=1}^{n-1} P(x_n | x_i)}{[P(x_n)]^{n-2}}, \text{ so that}$$

$$H(x_n | x_1, x_2, \dots, x_{n-1}) = - \sum_{x_1, x_2, \dots, x_n} P(x_1, x_2, \dots, x_n) \log \left( \frac{\prod_{i=1}^{n-1} P(x_n | x_i)}{[P(x_n)]^{n-2}} \right)$$

$$\begin{aligned}
&= - \sum_{x_1, x_2, \dots, x_n} P(x_1, x_2, \dots, x_n) \sum_{i=1}^{n-1} \log P(x_n | x_i) + (n-2) \sum_{x_1, x_2, \dots, x_n} P(x_1, x_2, \dots, x_n) \log P(x_n) \\
&= - \sum_{i=1}^{n-1} \sum_{x_i, x_n} P(x_i, x_n) \log P(x_n | x_i) + (n-2) \sum_{x_n} P(x_n) \log P(x_n) \\
&= \sum_{i=1}^{n-1} H(x_n | x_i) - (n-2) H(x_n) \tag{Equation 9.8}
\end{aligned}$$

where  $H(x_n)$  is the entropy and  $H(x_n | x_i)$  is the conditional entropy defined as

$$H(x_n) = - \sum_{x_n} P(x_n) \log P(x_n) \text{ and}$$

$$H(x_n | x_i) = - \sum_{x_n, x_i} P(x_n, x_i) \log P(x_n | x_i)$$

respectively. It is known (see [15]) that the conditional entropy  $H(x_n | x_i)$  can be expanded as

$$H(x_n | x_i) = H(x_n) - I(x_i, x_n) \tag{Equation 9.9}$$

where  $I(x_i, x_n) = \sum_{x_i, x_n} P(x_i, x_n) \log \frac{P(x_i, x_n)}{P(x_i)P(x_n)}$  is the mutual information between the

features  $x_i$  and  $x_n$ . By substituting Equation 9.9 in Equation 9.8, we get

$$\begin{aligned}
H(x_n | x_1, x_2, \dots, x_{n-1}) &= \sum_{i=1}^{n-1} H(x_n) - \sum_{i=1}^{n-1} I(x_i, x_n) - (n-2) H(x_n) \\
&= (n-1) H(x_n) - (n-2) H(x_n) - \sum_{i=1}^{n-1} I(x_i, x_n) \\
&= H(x_n) - \sum_{i=1}^{n-1} I(x_i, x_n)
\end{aligned}$$

Thus, Lemma 9.2 is proved.

**Assumption 9.2** *The features  $x_1, x_2, \dots, x_n$  are independent.*

**Lemma 9.3** *If  $x_1, x_2, \dots, x_n$  are independent features (Assumption 9.2) and  $\omega$  is the class label associated with the feature vector  $(x_1, x_2, \dots, x_n)$ , then the joint entropy function  $H(\omega, x_1, x_2, \dots, x_n)$  is equal to*

$$H(\omega) + \sum_{i=1}^n H(x_i) - \sum_{i=1}^n I(\omega, x_i)$$

**Proof:**

By the chain rule of entropy (see [15]), the joint entropy function  $H(\omega, x_1, x_2, \dots, x_n)$  can be written as

$$H(\omega, x_1, x_2, \dots, x_n) = H(\omega) + H(x_1 | \omega) + H(x_2 | x_1, \omega) + \dots + H(x_n | x_1, x_2, \dots, x_{n-1}, \omega)$$

Equation 9.10

If the features  $x_1, x_2, \dots, x_n$  are independent (by Assumption 9.2), then

$$\begin{aligned} H(x_n | x_1, x_2, \dots, x_{n-1}, \omega) &= - \sum_{x_1, x_2, \dots, x_n, \omega} P(x_1, x_2, \dots, x_n, \omega) \log P(x_n | x_1, x_2, \dots, x_{n-1}, \omega) \\ &= - \sum_{x_1, x_2, \dots, x_n, \omega} P(x_1, x_2, \dots, x_n, \omega) \log P(x_n | \omega) \\ &= - \sum_{x_1, x_2, \dots, x_n, \omega} P(x_n, \omega) \log P(x_n | \omega) \\ &= H(x_n | \omega) \end{aligned}$$

Equation 9.11

Similarly,

$$\begin{aligned} H(x_2 | x_1, \omega) &= H(x_2 | \omega) \\ &\vdots \\ H(x_{n-1} | x_1, x_2, x_3, \dots, x_{n-2}, \omega) &= H(x_{n-1} | \omega) \end{aligned}$$

Equation 9.12

From Equation 9.11 and Equation 9.12, Equation 9.10 becomes

$$H(\omega, x_1, x_2, \dots, x_n) = H(\omega) + H(x_1 | \omega) + H(x_2 | \omega) + \dots + H(x_n | \omega) \quad \text{Equation 9.13}$$

It is known (see [15]) that  $H(x_i | \omega)$  can be expanded as  $H(x_i) - I(\omega, x_i)$ . Therefore, Equation 9.13 can be written as

$$H(\omega, x_1, x_2, \dots, x_n) = H(\omega) + \sum_{i=1}^n H(x_i) - \sum_{i=1}^n I(\omega, x_i)$$

Thus, Lemma 9.3 is proved.

**Theorem 9.1** *The Maximum Relevance Minimal Redundancy (MRMR) feature selection procedure, which maximizes the criterion*

$$\sum_{i=1}^k I(x_i, \omega) - \sum_{i=2}^k \sum_{j=1}^{i-1} I(x_i, x_j)$$

*minimizes an upper bound on the Bayes Error rate.*

**Proof:**

Hellman and Raviv in [33] showed that

$$\sigma_e \leq \frac{1}{2} H(\omega | X), \quad \text{Equation 9.14}$$

where  $\sigma_e$  is the Bayes error rate and  $H(\omega | X)$  is the class conditional entropy. Equation 9.14 shows that greater the value of  $H(\omega | X)$ , the greater is the upper bound on the Bayes error rate. Therefore, to minimize classification error, one needs to minimize the class conditional entropy  $H(\omega | X)$ . It is known (see [15]) that

$$H(\omega | X) = H(\omega) - I(X; \omega).$$

Because  $I(X; \omega)$  is a negative term, minimizing  $H(\omega | X)$  is equivalent to maximizing  $I(X; \omega)$ . As  $x = (x_1, x_2, \dots, x_n)$ , we have

$$I(X; \omega) = I(x_1, x_2, \dots, x_n; \omega).$$

It is known (see [15]) that  $I(x_1, x_2, \dots, x_n; \omega)$  can be expanded as

$$I(x_1, x_2, \dots, x_n; \omega) = \underbrace{\sum_{i=1}^n H(x_i | x_{i-1}, \dots, x_1)}_{\text{TERM 1}} - \underbrace{\sum_{i=1}^n H(x_i | x_{i-1}, \dots, x_1, \omega)}_{\text{TERM 2}} \quad \text{Equation 9.15}$$

TERM 1 in Equation 9.15 can be written as

$$\sum_{i=1}^n H(x_i | x_{i-1}, \dots, x_1) = H(x_1) + H(x_2 | x_1) + H(x_3 | x_1, x_2) + \dots + H(x_n | x_1, x_2, \dots, x_{n-1}) \quad \text{Equation 9.16}$$

From Lemma 9.2, we have

$$\begin{aligned} H(x_2 | x_1) &= H(x_2) - I(x_2, x_1) \\ H(x_3 | x_2, x_1) &= H(x_3) - I(x_3, x_1) - I(x_3, x_2) \\ &\vdots \\ H(x_n | x_1, x_2, \dots, x_{n-1}) &= H(x_n) - \sum_{i=1}^{n-1} I(x_i, x_n) \end{aligned} \quad \text{Equation 9.17}$$

Now using Equation 9.17, Equation 9.16 can be rewritten as

$$\sum_{i=1}^n H(x_i | x_{i-1}, \dots, x_1) = \sum_{i=1}^n H(x_i) - \sum_{i=2}^n \sum_{j=1}^{i-1} I(x_i, x_j) \quad \text{Equation 9.18}$$

Replacing TERM 1 in Equation 9.15 using Equation 9.18 gives

$$I(x_1, x_2, \dots, x_n; \omega) = \underbrace{\sum_{i=1}^n H(x_i) - \sum_{i=2}^n \sum_{j=1}^{i-1} I(x_i, x_j)}_{\text{TERM 1}} - \underbrace{\sum_{i=1}^n H(x_i | x_{i-1}, \dots, x_1, \omega)}_{\text{TERM 2}} \quad \text{Equation 9.19}$$

From Equation 9.19, it is straight forward to see that maximizing TERM 1 results in maximizing  $I(x_1, x_2, \dots, x_n; \omega)$ , thereby minimizing the upper bound on Bayes error rate. Here, we point out that TERM 1 does not involve class information and therefore, the MRMR feature selection based on maximizing TERM 1 is an *unsupervised* version of MRMR feature selection. Now, TERM 2 in Equation 9.19 can be written as



$$\sum_{i=1}^n H(x_i | x_{i-1}, \dots, x_1, \omega) = H(x_1 | \omega) + H(x_2 | x_1, \omega) + \dots + H(x_n | x_1, x_2, \dots, x_{n-1}, \omega)$$

Equation 9.20

It is known (see [15]) that

$$H(\omega, x_1, x_2, \dots, x_n) = H(\omega) + H(x_1 | \omega) + H(x_2 | x_1, \omega) + \dots + H(x_n | x_1, x_2, \dots, x_{n-1}, \omega).$$

Equation 9.21

Using Equation 9.21, Equation 9.20 can be written as

$$\sum_{i=1}^n H(x_i | x_{i-1}, \dots, x_1, \omega) = H(\omega, x_1, x_2, \dots, x_n) - H(\omega).$$

Equation 9.22

From Lemma 9.3, we have

$$H(\omega, x_1, x_2, \dots, x_n) = H(\omega) + \sum_{i=1}^n H(x_i) - \sum_{i=1}^n I(\omega, x_i).$$

Equation 9.23

From Equation 9.23, Equation 9.22 can be written as

$$\sum_{i=1}^n H(x_i | x_{i-1}, \dots, x_1, \omega) = \sum_{i=1}^n H(x_i) - \sum_{i=1}^n I(\omega, x_i)$$

Equation 9.24

Now, substituting Equation 9.24 for TERM 2 in Equation 9.19, we get

$$I(x_1, x_2, \dots, x_n; \omega) = \underbrace{\sum_{i=1}^n H(x_i) - \sum_{i=2}^n \sum_{j=1}^{i-1} I(x_i, x_j)}_{\text{TERM 1}} - \underbrace{\sum_{i=1}^n H(x_i) - \sum_{i=1}^n I(\omega, x_i)}_{\text{TERM 2}}$$

Equation 9.25

On simplifying Equation 9.25, we get

$$I(x_1, x_2, \dots, x_n; \omega) = \sum_{i=1}^n I(\omega, x_i) - \sum_{i=2}^n \sum_{j=1}^{i-1} I(x_i, x_j).$$

Equation 9.26

Substituting Equation 9.26, in Equation 9.14

$$\sigma_e \leq \frac{1}{2} \left[ H(\omega) - \left( \sum_{i=1}^n I(\omega, x_i) - \sum_{i=2}^n \sum_{j=1}^{i-1} I(x_i, x_j) \right) \right]$$

Equation 9.27

From Equation 9.27, it is straight forward to see that any set of  $k$  features ( $k \leq n$ ) that maximize the criterion for MRMR feature selection, i.e.,  $\left( \sum_{i=1}^n I(\omega, x_i) - \sum_{i=2}^n \sum_{j=1}^{i-1} I(x_i, x_j) \right)$ , also maximize  $I(x_1, x_2, \dots, x_n; \omega)$ , thereby minimizing an upper bound on Bayes error rate. Thus, we prove Theorem 9.1.

## **CHAPTER 10**

### **THE KDD CUP 1999 INTRUSION**

#### **DETECTION DATASET**

The dependence tree based anomaly detection method is tested using the benchmark KDD Cup 1999 dataset [41]. The entire KDD Cup 1999 dataset contains about 5,000,000 connection records. However, a concise dataset known as the “10% training” dataset has been provided to allow for faster training of anomaly and intrusion detection systems. We use the 10% training dataset. The 10% training dataset consists of 494, 021 connection records, each record labeled as normal or as a specific attack type. There are 22 different attack types in the training dataset. The KDD Cup 1999 test dataset contains 311, 030 connection records. The test dataset contains 17 additional attack types that are not present in the training data.

Each connection record in the KDD Cup 1999 dataset contains 41 features and a label indicating whether the connection is normal or an attack. The 41 features fall into three categories: (1) basic features, (2) content features, and (3) temporal features. A brief description of the 41 KDD Cup 1999 dataset features is provided in the following section.

### 10.1 Features in KDD Cup 1999 Dataset

Here we give a brief description of features in the KDD Cup 1999 datasets. The first nine features are known as “basic” features and contain intrinsic information of a single network connection. The basic features are described in Table 10.1.

**Table 10.1 Basic features of KDD Cup 1999 dataset.**

No.	Feature Name	Description
1	Duration	Length (number of seconds) of the connection
2	Protocol	Type of the protocol, e.g. tcp, udp, etc.
3	Service	Network service on the destination, e.g., http, telnet, etc.
4	Source Bytes	Number of data bytes from source to destination
5	Destination Bytes	Number of data bytes from destination to source
6	Flag	Normal or error status of the connection
7	Land	1 if connection is from/to the same host/port; 0 otherwise
8	Wrong Fragment	Number of “wrong” fragments
9	Urgent	Number of urgent packets

The next thirteen features in a KDD Cup 1999 connection record are known as “content” features. These features use domain knowledge to assess the payload of TCP packets. A brief description of content features is given in Table 10.2.

**Table 10.2 Content features of KDD Cup 1999 dataset.**

No.	Feature Name	Description
10	Hot	Number of “hot” indicators
11	Num_failed_logins	Number of failed login attempts
12	Logged_in	1 if successfully logged in; 0 otherwise
13	Num_compromised	Number of “compromised” conditions
14	Root_shell	1 if root shell is obtained; 0 otherwise
15	Su_attempted	1 if “su root” command attempted; 0 otherwise
16	Num_root	Number of “root” accesses
17	Num_file_creations	Number of file creation operations
18	Num_shells	Number of shell prompts
19	Num_access_files	Number of operations on access control files
20	Num_outbound_cmds	Number of outbound commands in an ftp session

21	Is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise
22	Is_guest_login	1 if the login is a “guest” login; 0 otherwise

The next nineteen features in a connection record are known as “temporal” features. These features are collected over a 2 second time-window. A brief description of content features is given in Table 10.3.

**Table 10.3 Temporal features of KDD Cup 1999 dataset.**

No.	Feature Name	Description
23	Count	Number of connections to the same host as the current connection
24	Serror_rate	Percentage of connections that have “SYN” errors
25	Rerror_rate	Percentage of connections that have “REJ” errors
26	Same_srv_rate	Percentage of connections to the same service
27	Diff_srv_rate	Percentage of connections to different services
28	Srv_count	Number of connections to the same service as the current connection
29	Srv_serror_rate	Percentage of connections that have “SYN” errors
30	Srv_rerror_rate	Percentage of connections that have “REJ” errors
31	Srv_diff_host_rate	Percentage of connections to different hosts
32	dst_host_count	Number of connections to the same destination host as the current connection
33	dst_host_srv_count	Percentage of connections to the same service at the destination host
34	dst_host_same_srv_rate	Percentage of connections to different services at the destination host
35	Dst_host_diff_srv_rate	Percentage of connections to different services at the destination host
36	dst_host_same_src_port_rate	Number of connections to the same port at destination host
37	dst_host_srv_diff_host_rate	Percentage of connections to different hosts at the destination host
38	dst_host_serror_rate	Percentage of connections that have “SYN” errors at the destination host
39	dst_host_srv_serror_rate	Percentage of connections that have “SYN” errors at the destination host
40	dst_host_rerror_rate	Percentage of connections that have “REJ” errors at the destination host
41	dst_host_srv_rerror_rate	Percentage of connections that have “REJ” errors at the destination host

It is to be noted here that all continuous features in the KDD Cup 1999 dataset (i.e., Duration, Source Bytes, Destination Bytes, and all “content” features) are discretized using Fayyad and Irani’s discretization method [50] before being input to the dependence tree based anomaly detection method.

Table 10.4 gives the number of normal and attack connection records in the KDD Cup 1999 training and test datasets. There are four classes of attacks in KDD Cup 1999: (1) denial-of-service (DoS) attacks (e.g., “syn flood”), (2) surveillance and other probing (Probe) attacks (e.g., “port scanning”), (3) unauthorized access attacks to local superuser privileges (U2R) (e.g., “buffer overflow”), and (4) unauthorized access from a remote machine (R2L) (e.g., “guess password”). The training data contains 24 different attack types that fall in to one of the four classes. The KDD Cup 1999 test data includes an addition 14 attack types that are not present in the training data.

**Table 10.4 Distribution of normal and attack connections in the KDD Cup 1999 dataset.**

<b>Dataset</b>	<b>Normal</b>	<b>DOS</b>	<b>PROBE</b>	<b>U2R</b>	<b>R2L</b>	<b>Total</b>
Training	97278	391458	4107	52	1126	494021
Testing	60593	229853	4166	228	16189	311029

The major motivations for using the benchmark KDD Cup 1999 datasets are:

- The KDD Cup 1999 dataset has been used popularly as a standard for comparing the performance of intrusion detection methods. This allows us to compare the performance of our dependence tree based anomaly detection method with the performance of the other intrusion detection methods reported in recent literature.

- The data instances in KDD Cup 1999 test dataset are labeled, making it possible for us to verify the detection accuracy and false positive rate of our dependence tree based anomaly detection method.
- The KDD Cup 1999 test dataset contains 17 additional attacks that are not included in the training dataset. This feature of the dataset allows us to gauge the performance of the dependence tree based anomaly detection method on unseen or new attacks.

## CHAPTER 11

### EXPERIMENTS AND RESULTS

In this chapter we present the results of MRMR feature selection and the results of the dependence tree based anomaly detection method on the KDD Cup 1999 dataset.

#### 11.1 Results of MRMR Feature Selection on

##### KDD Cup 1999 Dataset

In Table 11.1, we give the results of MRMR *supervised* feature selection (see Equation 9.26) on KDD Cup 1999 datasets. Each feature in Table 11.1 is ranked based on its relevance to the class variable.

**Table 11.1** The first eight features in KDD Cup 1999 datasets selected through MRMR supervised feature selection.

Rank	Feature No.	Feature Name
1	5	Source Bytes
2	14	Root shell
3	6	Flag
4	32	Dst host count
5	23	Count
6	12	Logged in
7	37	Dst host srv diff host rate
8	31	Srv diff host rate



In Table 11.2, we give the results of MRMR *unsupervised* feature selection (see Equation 9.19) on KDD Cup 1999 datasets. Each feature in Table 11.2 is ranked based on its entropy value.

**Table 11.2 The first eight features in KDD Cup 1999 datasets selected through MRMR unsupervised feature selection.**

Rank	Feature No.	Feature Name
1	23	Count
2	24	Srv_Count
3	5	Source Bytes
4	33	Dst_host_srv_count
5	3	Service
6	35	Dst_host_diff_srv_rate
7	34	Dst_host_same_srv_rate
8	36	Dst_host_same_src_port_rate

Only eight of the 41 KDD Cup 1999 features have been incrementally selected by both the supervised and unsupervised MRMR feature selection methods. For the remaining features, the supervised and unsupervised MRMR feature selection criteria (Equation 9.26 and Equation. 9.19) incurred negative values, meaning that the remaining features had more redundancy than relevance to classification. Next, we present the results of the dependence tree based anomaly detection method on the features selected through the supervised and unsupervised versions of MRMR feature selection method.

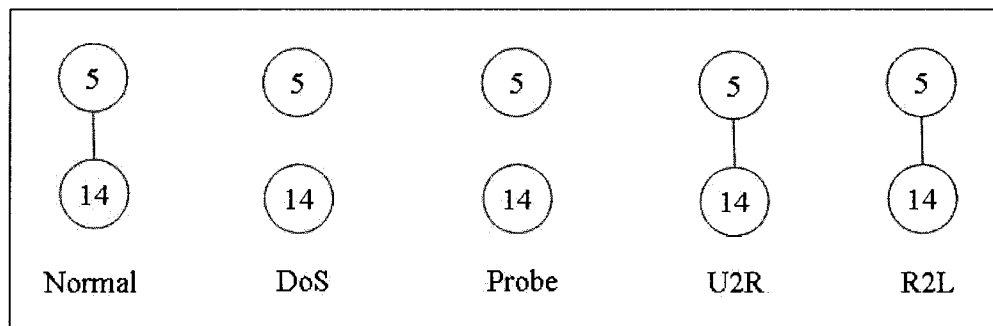
## 11.2 Dependence Tree Results

The performance of the dependence tree based anomaly detection method is gauged using three measures: (1) detection accuracy, which is the percentage of instances correctly detected in each of the five classes (i.e., normal, denial-of-service attack, probe attack, user-to-root attack, and remote-to-login attack), (2) false positive rate, which is the

percentage of normal instances detected as attacks, and (3) the total attack detection accuracy, which is the percentage of all attacks correctly detected. Next, we present the results of the dependence tree based anomaly detection method using the features selected by supervised MRMR feature selection method (see Table 11.1).

### 11.2.1 Dependence Tree Results with Supervised MRMR Selection

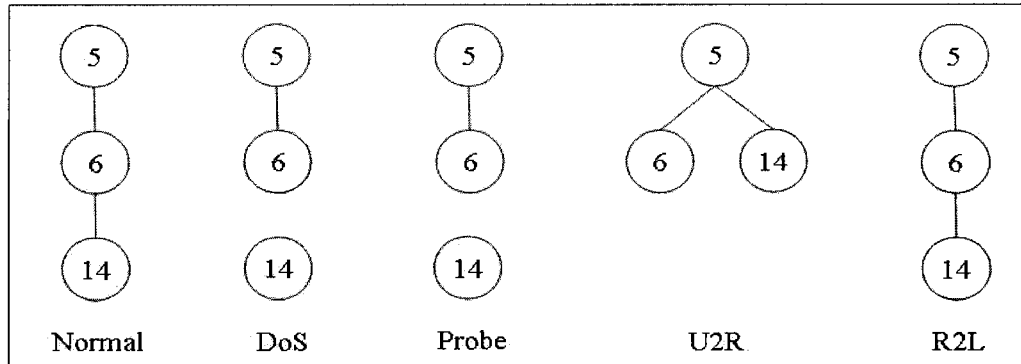
In this section we present the class-wise detection accuracy of dependence trees built using the features selected by the supervised MRMR selection algorithm. Figure 11.1 shows dependence trees with two features: Source (5) and Root\_shell (14). The class-wise detection accuracy is 93.68% for Normal, 68.9% for DoS, 94.89% for Probe, 17.54% for U2R, and 3.5% for R2L. The total attack detection accuracy is 91.11% at a false positive rate of 6.31%.



**Figure 11.1 Dependence trees with two features (5, 14) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.**

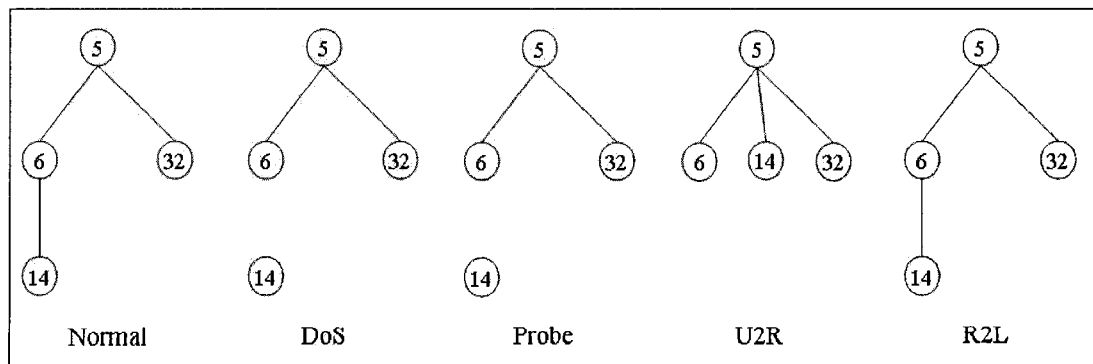
Figure 11.2 shows dependence trees with three features: Source (5), Root\_shell (14), and Flag (6). The class-wise detection accuracy for the dependence trees in Figure 11.2 is 97.45% for Normal, 68.91% for DoS, 92.51% for g()Probe, 17.98% for U2R, and

0.01% for R2L. The total attack detection accuracy is 89.40% at a false positive rate of 2.53%.



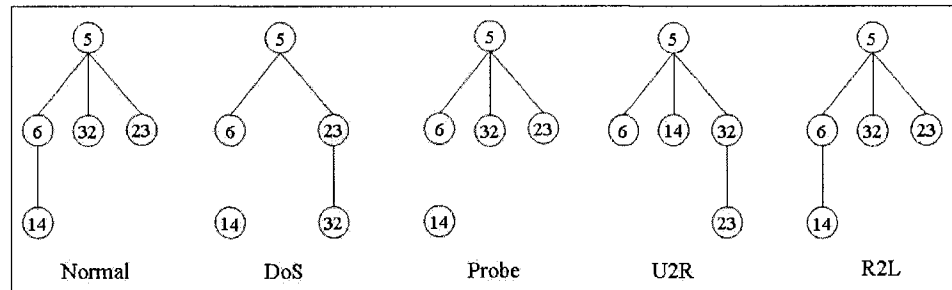
**Figure 11.2** Dependence trees with three features (5, 14, 6) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.

Figure 11.3 shows dependence trees with four features: Source (5), Root\_shell (14), Flag (6), and Dst\_host\_count (32). The class-wise detection accuracy for the dependence trees in Figure 11.3 is 97.87% for Normal, 68.91% for DoS, 91.14% for Probe, 12.28% for U2R, and 2.13% for R2L. The total attack detection accuracy is 89.56% at a false positive rate of 2.13%.



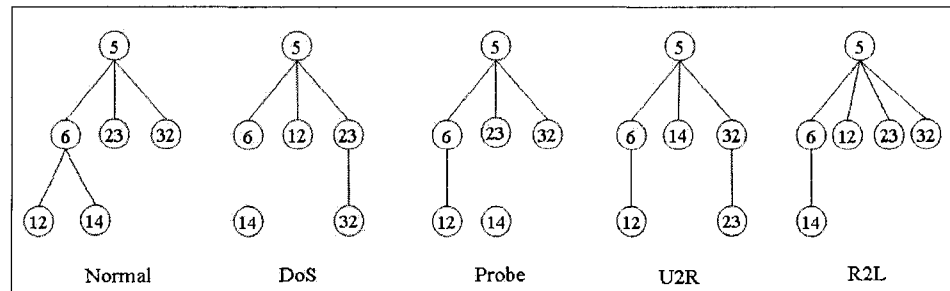
**Figure 11.3** Dependence trees with four features (5, 14, 6, 32) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.

Figure 11.4 shows dependence trees with five features: Source (5), Root\_shell (14), Flag (6), Dst\_host\_count (32), and Count (23). The class-wise detection accuracy for the dependence trees in Figure 11.4 is 97.89% for Normal, 93.83% for DoS, 81.18% for Probe, 12.28% for U2R, and 0.01% for R2L. The total attack detection accuracy is 88.30% at a false positive rate of 2.13%.



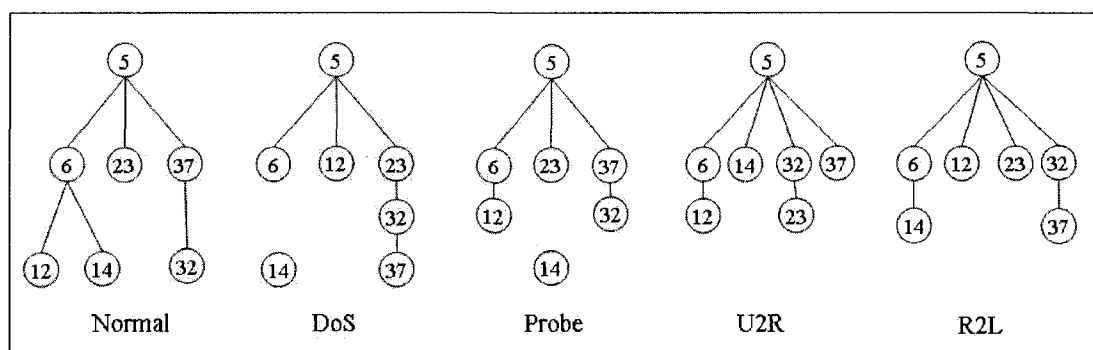
**Figure 11.4** Dependence trees with five features (5, 14, 6, 32, 23) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.

Figure 11.5 shows dependence trees with six features: Source (5), Root\_shell (14), Flag (6), Dst\_host\_count (32), Count (23), and Logged\_in (12). The class-wise detection accuracy for the dependence trees in Figure 11.5 is 97.95% for Normal, 93.83% for DoS, 81.93% for Probe, 9.658% for U2R, and 0.01% for R2L. The total attack detection accuracy is 88.31% at a false positive rate of 2.0%.



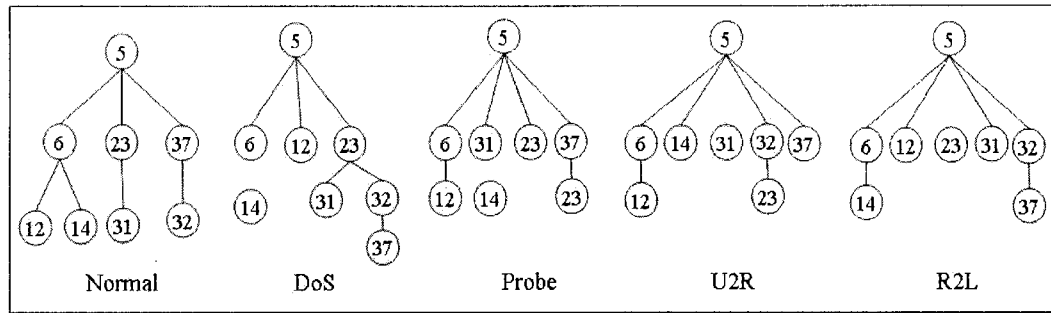
**Figure 11.5** Dependence trees with six features (5, 14, 6, 32, 23, 12) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.

Figure 11.6 shows dependence trees with seven features: Source (5), Root\_shell (14), Flag (6), Dst\_host\_count (32), Count (23), Logged\_in (12), and Dst\_host\_srv\_diff\_host\_rate (37). The class-wise detection accuracy for the dependence trees in Figure 11.6 is 98.90% for Normal, 93.82% for DoS, 77.2% for Probe, 9.21% for U2R, and 0.03% for R2L. The total attack detection accuracy is 88.21% at a false positive rate of 1.1%.



**Figure 11.6 Dependence trees with seven features (5, 14, 6, 32, 23, 12, 37) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.**

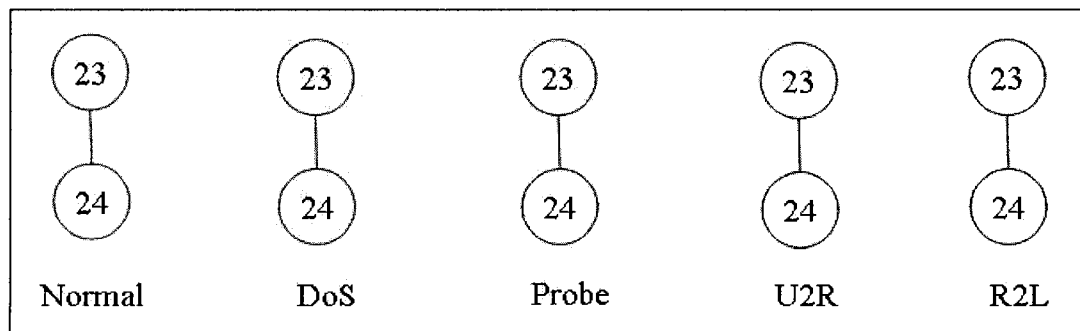
Figure 11.7 shows dependence trees with eight features: Source (5), Root\_shell (14), Flag (6), Dst\_host\_count (32), Count (23), Logged\_in (12), Dst\_host\_srv\_diff\_host\_rate (37), and Srv\_diff\_host\_rate (31). The class-wise detection accuracy for the dependence trees in Figure 11.7 is 98.76% for Normal, 93.81% for DoS, 68.24% for Probe, 9.21% for U2R, and 0.01% for R2L. The total attack detection accuracy is 88.07% at a false positive rate of 1.2%.



**Figure 11.7 Dependence trees with eight features (5, 14, 6, 32, 23, 12, 37, 31) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.**

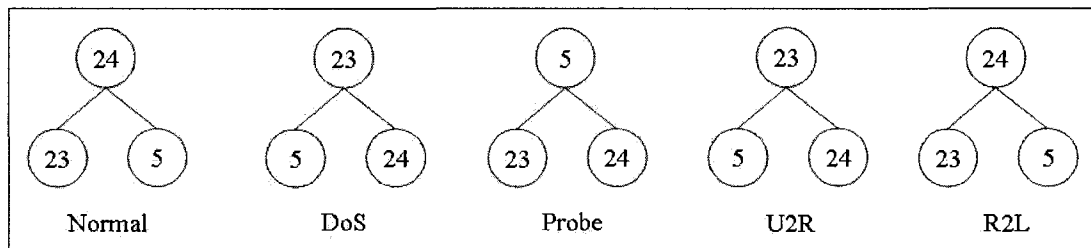
### 11.2.2 Dependence Tree Results with Unsupervised MRMR Selection

In this section we present the class-wise detection accuracy of dependence trees built using the features selected by the unsupervised MRMR feature selection algorithm. Figure 11.8 shows dependence trees with two features: Count (23) and Srv\_count (24). The class-wise detection accuracies for the dependence trees in Figure 11.8 are 60.00% for Normal, 81.16% for DoS, 58.74% for Probe, 86.40% for U2R, and 23.92% for R2L. The total attack detection accuracy is 97.13% at a false positive rate of 40.00%.



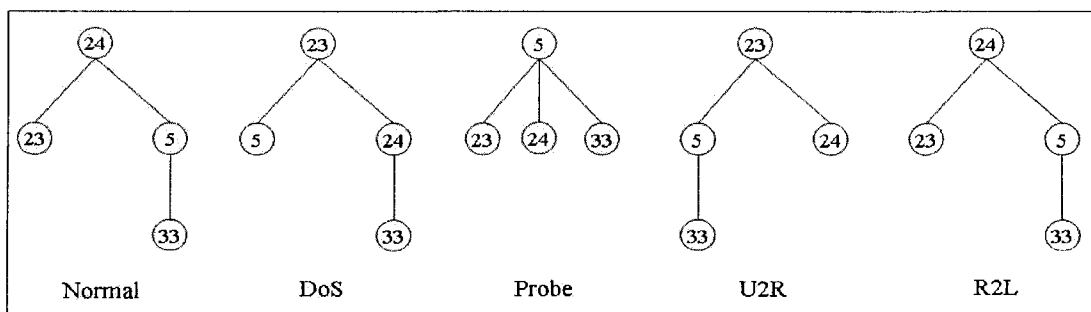
**Figure 11.8 Dependence trees with two features (23, 24) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.**

Figure 11.9 shows dependence trees with three features: Count (23), Srv\_count (24), and Source (5). The class-wise detection accuracies for the dependence trees in Figure 11.9 are 92.06% for Normal, 92.90% for DoS, 73.93% for Probe, 27.19% for U2R, and 16.17% for R2L. The total attack detection accuracy is 90.85% at a false positive rate of 7.94%.



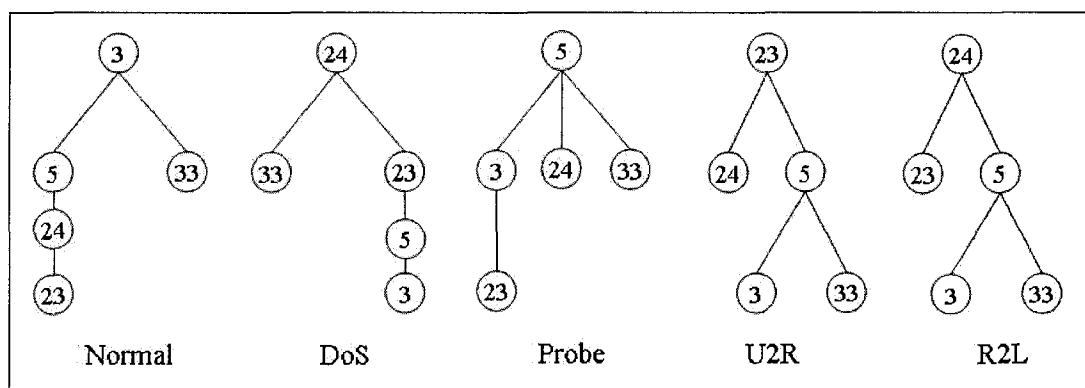
**Figure 11.9** Dependence trees with three features (23, 24, 5) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.

Figure 11.10 shows dependence trees with four features: Count (23), Srv\_count (24), Source (5), and Dst\_host\_srv\_count (33). The class-wise detection accuracies for the dependence trees in Figure 11.10 are 98.79% for Normal, 92.64% for DoS, 68.72% for Probe, 42.11% for U2R, and 0.43% for R2L. The total attack detection accuracy is 87.92% at a false positive rate of 1.2%.



**Figure 11.10** Dependence trees with four features (23, 24, 5, 33) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.

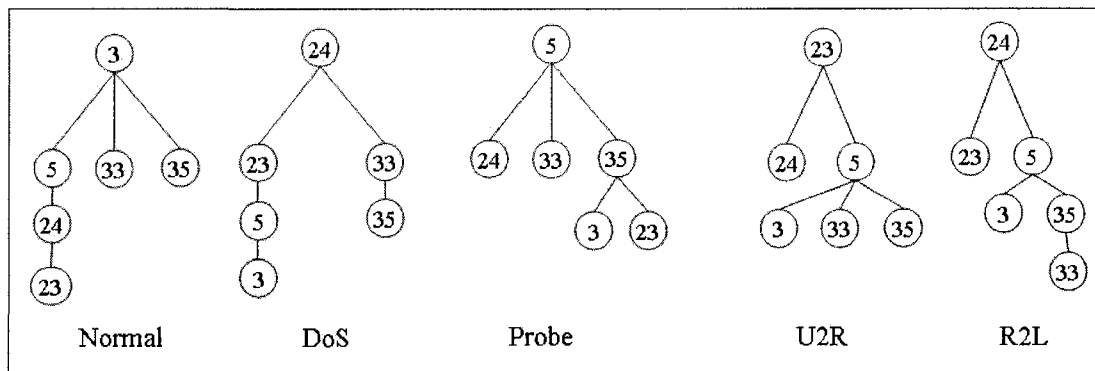
Figure 11.11 shows dependence trees with five features: Count (23), Srv\_count (24), Source (5), Dst\_host\_srv\_count (33), and Service (3). The class-wise detection accuracies for the dependence trees in Figure 11.11 are 99.50% for Normal, 92.82% for DoS, 72.30% for 8Probe, 8.33% for U2R, and 0.31% for R2L. The total attack detection accuracy is 87.85% at a false positive rate of 0.49%.



**Figure 11.11** Dependence trees with five features (23, 24, 5, 33, 3) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.

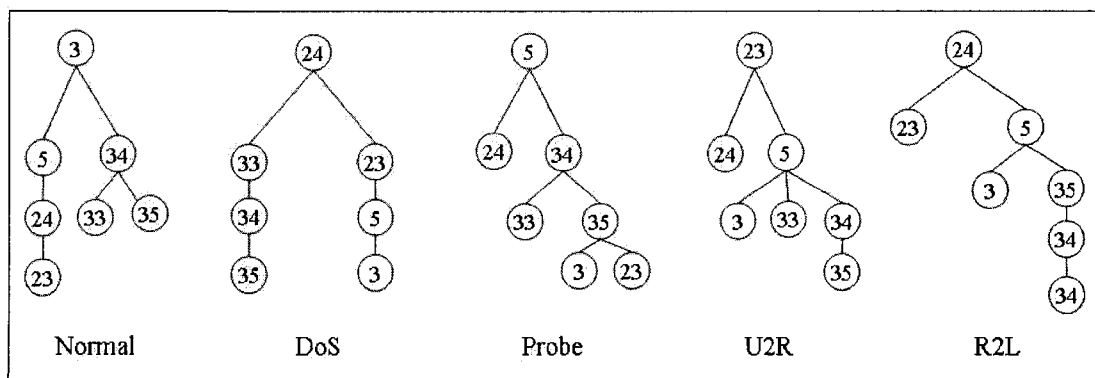
Figure 11.12 shows dependence trees with six features: Count (23), Srv\_count (24), Source (5), Dst\_host\_srv\_count (33), Service (3), and Dst\_host\_diff\_srv\_rate (35). The class-wise detection accuracies for the dependence trees in Figure 11.12 are 99.64% for Normal, 93.88% for DoS, 59.82% for Probe, 3.07% for U2R, and 0.98% for R2L. The total attack detection accuracy is 87.57% at a false positive rate of 0.36%.





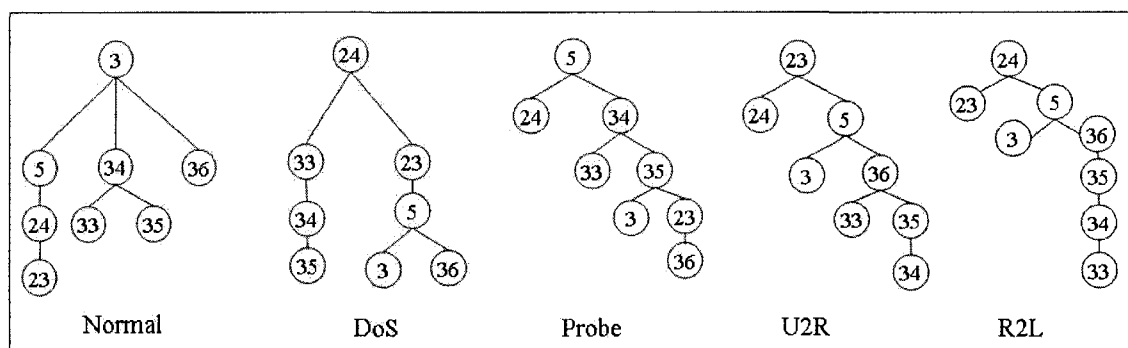
**Figure 11.12 Dependence trees with six features (23, 24, 5, 33, 3, 35) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.**

Figure 11.13 shows dependence trees with seven features: Count (23), Srv\_count (24), Source (5), Dst\_host\_srv\_count (33), Service (3), Dst\_host\_diff\_srv\_rate (35), and Dst\_host\_same\_srv\_rate (34). The class-wise detection accuracies for the dependence trees in Figure 11.3 are 99.70% for Normal, 93.62% for DoS, 61.69% for Probe, 3.07% for U2R, and 0.97% for R2L. The total attack detection accuracy is 87.45% at a false positive rate of 0.30%.



**Figure 11.13 Dependence trees with seven features (23, 24, 5, 33, 3, 35, 34) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.**

Figure 11.14 shows dependence trees with seven features: Count (23), Srv\_count (24), Source (5), Dst\_host\_srv\_count (33), Service (3), Dst\_host\_diff\_srv\_rate (35), Dst\_host\_same\_srv\_rate (34), and Dst\_host\_same\_src\_port\_rate (36). The class-wise detection accuracies for the dependence trees in Figure 11.14 are 99.76% for Normal, 93.65% for DoS, 57.73% for Probe, 3.51% for U2R, and 1.02% for R2L. The total attack detection accuracy is 87.38% at a false positive rate of 0.24%.



**Figure 11.14 Dependence trees with eight features (23, 24, 5, 33, 3, 35, 34, 36) for classifying Normal, DOS, Probe, U2R, and R2L connections in KDD Cup 1999 dataset.**

### 11.3 Comparison with Naïve Bayes and ID3 Anomaly

#### Detection Methods

In this section, we compare the performance of dependence tree based anomaly detection method with the performance of naïve Bayes [13] and ID3 [14] anomaly detection using the KDD Cup 1999 features selected from supervised and unsupervised MRMR feature selection.

### 11.3.1 Results with Supervised MRMR Feature Selection

Table 11.3 gives the detection accuracies and false positive rates obtained by three methods: (1) dependence tree based anomaly detection (DTree), (2) naïve Bayes (NB), and (3) ID3 decision tree (ID3) on KDD Cup 1999 dataset. The features in Table 11.3 are obtained using the supervised MRMR feature selection method.

**Table 11.3 Results of Dependence Tree based anomaly detection (DTree), Naïve Bayes (NB) anomaly detection, and ID3 anomaly detection on KDD Cup 1999 features selected by the supervised MRMR feature selection method.**

Features	Method	Normal	DoS	Probe	U2R	R2L	TAD	FPR
5, 14	DTree	93.68	68.90	94.89	17.54	3.50	91.11	6.30
	NB	98.50	94.60	17.70	9.20	0.40	87.14	1.50
	ID3	98.46	94.58	17.69	8.77	0.36	87.14	1.54
5, 14, 6	DTree	97.47	68.91	92.51	17.98	0.02	89.40	2.53
	NB	99.40	94.30	17.90	11.00	0	86.83	0.60
	ID3	98.81	94.28	24.89	6.14	0.02	86.95	1.19
5, 14, 6, 32	DTree	97.87	68.91	91.14	12.28	0.02	89.56	2.13
	NB	99.50	94.20	19.80	9.20	0	86.82	0.50
	ID3	98.81	94.24	18.36	5.26	0.02	86.80	1.19
5, 14, 6, 32, 23	DTree	97.89	93.83	81.18	12.28	0.02	88.30	2.11
	NB	98.70	97.10	26.90	9.20	0	89.54	1.30
	ID3	97.80	94.26	69.56	5.26	0.03	87.68	2.20
5, 14, 6, 32, 23, 12	DTree	97.95	93.83	81.93	9.65	0.02	88.31	2.06
	NB	98.60	97.10	35.40	9.20	0	89.72	1.40
	ID3	97.80	94.26	69.49	3.51	0.01	87.68	2.20
5, 14, 6, 32, 23, 12, 37	DTree	98.90	93.82	77.20	9.21	0.04	88.21	1.10
	NB	98.60	97.10	30.30	10.10	0.80	89.68	1.40
	ID3	97.97	94.27	69.49	4.83	0.03	87.68	2.03
5, 14, 6, 32, 23, 12, 37, 31	DTree	98.76	93.81	68.24	9.21	0.02	88.07	1.24
	NB	98.63	97.01	31.04	10.53	1.15	89.64	1.37
	ID3	99.65	94.27	69.16	4.82	0.02	87.67	0.35

From Table 11.3, we observe that there are minor differences between the performances of the three methods when detecting normal class except in the case of the DTree method with two features (i.e., 5 and 13), which has the least detection accuracy of

93.68% for normal instances (and therefore, the highest false positive rate of 6.30%). However, in this case we note that the DTree method yields relatively high detection accuracies in detecting Probe, U2R, and R2L attacks. Further, the DTree method with three features (5, 14, and 6) and four features (5, 14, 6, and 32) outperforms the NB and ID3 methods in detecting Probe and U2R attacks. In these cases, however, the DTree method has relatively lower accuracies (68.91% with three features and 68.91% with four features) in detecting DoS attacks. The reason for such low detection accuracies is that a considerable number of DoS instances have been misclassified as Probe attacks. Such misclassification, though undesirable, is better than the unacceptably low detection accuracies for Probe attacks, achieved by NB and ID3 methods. In the rest of the cases with five, six, seven, and eight features, the DTree method outperforms the NB and the ID3 methods in detecting Probe and U2R attacks, yet maintaining very comparable accuracies in detecting Normal, DoS, and R2L attacks.

### **11.3.2 Results with Unsupervised MRMR Feature Selection**

Table 11.4 gives the percentage attack detection accuracies and false positive rates obtained by three methods: (1) DTree, (2) NB, and (3) ID3 on KDD Cup 1999 dataset. The features in Table 11.4 are obtained using the unsupervised MRMR feature selection method.

From Table 11.4, we observe that the DTree method outperforms the NB and ID3 methods in detecting normal instances except in two cases: (1) the DTree method with two features (23 and 24), and (2) the DTree method with three features (23, 24, and 5). However, in both these cases, we note the detection accuracies of the DTree method are considerably higher than the detection accuracies of NB and ID3 methods in detecting

Probe, U2R, and R2L attacks. The DTree method with four features (23, 24, 5, and 33) achieves detection accuracy as high as 42.11% for U2R attacks, which is significantly higher than the detection accuracies of NB and ID3 methods for detecting U2R attacks with four features. The DTree method with five features (23, 24, 5, 33, and 3) achieves comparable detection accuracies for DoS, Probe, and U2R methods at a false positive rate as low as 0.5%, which is considerably lower than the false positive rate of NB and ID3 methods implemented with five features. The DTree method with six, seven, and eight features achieves fine improvements in detecting normal instances but falls behind the NB and ID3 methods in detecting Probe, U2R and R2L methods

**Table 11.4 Results of Dependence Tree based anomaly detection (DTree), Naïve Bayes (NB) anomaly detection, and ID3 anomaly detection on KDD Cup 1999 features selected by the unsupervised MRMR feature selection method.**

Features	Method	Normal	DoS	Probe	U2R	R2L	TAD	FPR
23, 24	DTree	60.00	81.16	58.74	86.40	23.92	97.13	40.00
	NB	97.90	93.40	10.80	0	0	85.92	2.10
	ID3	98.28	82.62	45.10	0	0	76.58	1.72
23, 24, 5	DTree	92.06	92.90	73.93	27.19	16.17	90.85	7.95
	NB	97.70	96.60	37.80	0	3.50	89.52	2.30
	ID3	98.25	94.12	58.43	0	0.72	87.40	1.75
23, 24, 5, 33	DTree	98.79	92.64	68.72	42.11	0.43	87.92	1.21
	NB	98.00	95.20	73.40	0	1.50	88.72	2.00
	ID3	96.26	94.10	65.22	1.04	0.38	87.48	3.74
23, 24, 5, 33, 3	DTree	99.50	92.82	72.30	8.33	0.32	87.85	0.50
	NB	97.80	95.10	76.20	10.50	8.00	89.07	2.20
	ID3	97.95	94.17	74.44	7.02	0.49	87.70	2.05
23, 24, 5, 33, 3, 35	DTree	99.64	93.88	59.82	3.07	0.98	87.57	0.36
	NB	97.90	95.80	73.20	10.10	8.10	89.69	2.10
	ID3	97.81	93.80	74.56	3.51	1.01	87.40	2.19
23, 24, 5, 33, 3, 35, 34	DTree	99.70	93.62	61.69	3.07	0.98	87.45	0.30
	NB	97.60	95.00	72.00	10.50	8.50	88.91	2.40
	ID3	97.75	93.71	74.39	3.070	1.90	87.37	2.25
23, 24, 5, 33, 3, 35, 34, 36	DTree	99.76	93.65	57.73	3.51	1.02	87.38	0.24
	NB	97.80	95.00	72.40	8.30	8.80	88.93	2.20
	ID3	97.74	93.71	74.51	3.07	1.90	87.38	2.26

## 11.4 Comparison with Other Studies on

### KDD Cup 1999 Datasets

In Table 11.5, we compare the class-wise detection accuracies of dependence tree based anomaly detection method with results reported in Song et al. [19] and Bouzida et al. [18] over the KDD Cup 1999 dataset.

**Table 11.5 Comparison of class-wise percentage detection accuracies and false positive rates of the dependence tree based anomaly detection method with the network attack detection methods reported in Bouzida et al. and Song et al.**

Work	Classification Models	Normal	DoS	Probe	U2R	R2L	FPR
Ours	DTree with <i>four supervised</i> MRMR selected features.	97.87	68.91	91.14	12.28	0.02	2.13
	DTree with <i>five supervised</i> MRMR selected features.	97.89	93.83	81.18	12.28	0.02	2.11
	DTree with <i>six supervised</i> MRMR selected features.	97.95	93.83	81.93	9.65	0.02	2.06
	DTree with <i>seven supervised</i> MRMR selected features.	98.90	93.82	77.20	9.21	0.04	1.10
	DTree with <i>eight supervised</i> MRMR selected features.	98.76	93.81	68.24	9.21	0.02	1.24
	DTree with <i>four unsupervised</i> MRMR selected features.	98.79	92.64	68.72	42.11	0.43	1.21
	DTree with <i>five unsupervised</i> MRMR selected features.	99.50	92.82	72.30	8.33	0.32	0.50
	DTree with <i>seven unsupervised</i> MRMR selected features.	99.70	93.62	61.69	3.07	0.98	0.30
	DTree with <i>eight unsupervised</i> MRMR selected features.	99.76	93.65	57.73	3.51	1.02	0.24
[18]	Nearest neighbor classification with 41 features.	99.5	97.01	72.01	6.60	1.21	0.5
	Nearest neighbor classification with 4 principal components.	99.50	97.14	74.40	7.91	0.80	0.5
	C4.5 classification with 41 features.	99.49	97.31	74.70	4.39	5.84	0.51
	C4.5 classification with 4 principal components.	99.00	97.25	66.80	6.58	0.01	1.00

[19]	Genetic programming with lap distance (8-4), age 10%.	99.7	95.36	48.5	10.10	0.2	0.30
	Genetic programming with lap distance (8-8), age 10%.	98.0	95.6	55.4	18.0	3.4	2.0
	Genetic programming with lap distance (16-4), age 10%.	98.7	95.7	55.1	10.2	1.8	1.3
	Genetic programming with lap distance (8-4), age 30%.	99.1	95.36	62.6	9.2	1.6	0.92
	Genetic programming with lap distance (16-4) age 30%.	98.6	95.5	56.5	11.4	0.8	1.4

In Table 11.5, we compare the results of dependence tree based anomaly detection method with the results reported in two recent papers: (1) Bouzida et al. used nearest neighbor classification and C4.5 decision tree with principal component analysis for to detect attacks in the KDD Cup 1999 datasets and (2) Song et al. used dynamic programming approach to detect attacks in the KDD Cup 1999 datasets. Although other studies on intrusion detection using KDD Cup 1999 dataset exist (for example, Sarasamma et al. [9][20]), the reason for choosing the works by Bouzida et al. and Song et al. for comparison with the dependence tree based anomaly detection method is that the false positive rates in these two works are considerably low and are comparable with the false positive rates achieved by the dependence tree based anomaly detection method. On the other hand, intrusion detection methods reported in Sarasamma et al. yielded an unacceptably high false positive rate and therefore are excluded from the comparison.

The results in Table 11.5, show that at 0.5% false positive rate Bouzida et al.'s nearest neighbor classification rule and C4.5 decision tree implemented on 41 features achieve detection accuracies that are very similar to the detection accuracies of our dependence tree based anomaly detection method with only five features selected through unsupervised MRMR feature selection method. Further, Bouzida et al.'s C4.5 decision

tree built on 4 principal components extracted from 41 features of the KDD Cup 1999 data, has a false positive rate 1.0% which is comparable to the false positive rate of the dependence tree method with seven features selected through supervised MRMR feature selection. In this case, however, our dependence tree based anomaly detection method outperforms the C4.5 method in detecting probe and U2R attacks.

The results in Table 11.5 show that Song et al.'s genetic programming approach with tap distance parameters (16-4) at age 10% and (16-4) at age 30% have 1.3% and 1.4% false positive rates, which are comparable to the 1.24% and 1.21% false positive rate of our dependence tree method with eight features selected through supervised MRMR feature selection and the dependence tree method with four features selected through unsupervised MRMR feature selection, respectively. We note that both these dependence trees outperform Song et al.'s genetic programming models with tap distance (16-4) in detection probe and U2R attacks. Similarly, we note that the dependence tree models outperform the remaining genetic programming models from Song et al. with tap distances (8-4) and (8-8) at ages 10% and 30% in detecting probe and U2R attacks, while no significant differences in the detection accuracies for DoS attacks have been observed between Song et al.'s models and our dependence trees.



## CHAPTER 12

### CONCLUSIONS AND FUTURE

#### RESEARCH DIRECTIONS

In this dissertation, we developed two novel pattern recognition methods: (1) the K-Means+ID3, and (2) the dependence tree method for supervised anomaly detection. The first method, K-Means+ID3, was developed to classify data instances into normal or anomaly classes. To detect anomaly data instances, the K-Means+ID3 method first partitions the training data instances into  $k$  disjoint clusters. Then, an ID3 decision tree built on each cluster learns the sub-groups within the cluster and partitions the decision space into finer classification regions, thereby improving the overall classification performance. We compared the performance of K-Means+ID3 method with the individual k-means and ID3 methods in terms of six performance measures. Results on network anomaly data, Duffing equation data, and mechanical system data showed that

1. the K-Means+ID3 method outperforms individual k-Means and the ID3 methods in terms of six performance measures over the 1998 network anomaly data,
2. the K-Means+ID3 has a very high detection accuracy (99.12 percent) and AUC performance (0.96) over the 1999 network anomaly data,

3. the K-Means+ID3 method shows better false positive rate and precision as compared to the individual k-Means and the ID3 methods over the 2000 network anomaly data,
4. the false positive rate, precision, and F-measure of the K-Means+ID3 method is higher than the k-Means method and lower than the ID3 method over the Duffing equation data, and
5. the K-Means+ID3 method has the highest precision and F-Measure over the mechanical system data.

Future research directions pertaining to the K-Mean+ID3 method include: (1) developing theoretical error bounds for K-Measn+ID3 method, and (2) comparing the performance of K-Means+ID3 with cascading classifiers developed using different clustering methods like hierarchical clustering, adaptive resonance theory (ART) neural networks, Kohonen's self-organizing maps and decision trees like C4.5 and Classification And Regression Trees (CART).

The second method, dependence tree based anomaly detection, was developed to classify network traffic data instances into one of normal, denial-of-service attack, probing attack, user-to-root attack, or remote-to-login attack. The dependence tree based anomaly detection method used Bayes classification rule to classify data instances into normal or one of the four attack types. Dependence trees were implemented to approximate class conditional densities in the Bayes classification rule. For improving the classification performance of dependence tree based anomaly detection, supervised and unsupervised Maximum Relevance Minimum Redundancy (MRMR) feature selection was used to select features that optimally characterize the class information. We derived

the theoretical relationship between Bayes error rate, dependence tree based classification error, and MRMR feature selection criterion and showed that both dependence tree approximation and MRMR feature selection criterion minimize an upper bound on the Bayes error rate. The performance of the dependence tree based anomaly detection method was demonstrated on the benchmark KDD Cup 1999 dataset. Further, the performance of the dependence tree based anomaly detection method was compared with the performance of the naïve Bayes and the ID3 decision tree methods as well as with the nearest neighbor rule and C4.5 decision trees presented in [18] and the genetic programming approach presented in [19]. Our results showed that

1. the dependence tree based anomaly detection method with five and six features selected through supervised MRMR feature selection method outperforms the naïve Bayes classifier and the ID3 decision tree method in detecting probe and U2R attacks,
2. the dependence tree based anomaly detection method with four features selected through unsupervised MRMR feature selection method achieves U2R attack detection accuracy as high as 42.11% at 1.21% false positive rate, outperforming both the naïve Bayes and the ID3 methods in detecting U2R attacks, and
3. the dependence tree based anomaly detection method outperforms Song et al. 's [19] genetic programming based anomaly detection models in detecting both probe and U2R attacks, while maintaining high detection accuracies in detecting normal and denial-of-service attacks.

Future research directions pertaining to our dependence tree based anomaly detection work include (1) using robust kernel density estimators for estimating the class

entropy of and mutual information between features and (2) modifying the MRMR feature selection criterion by adding a weight matrix to represent different misclassification costs so that the features selected through the modified feature selection criterion take into account the misclassification costs that may be incurred during classification.

## REFERENCES

- [1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2<sup>nd</sup> Edition, Morgan Kaufmann Series in Data Management Systems, 2001.
- [2] J. Cannady, "Artificial Neural Networks for Misuse Detection," in *Proceedings of the 1998 National Information Systems Security Conference (NISSC' 98)*, pp. 443-456, October 1996.
- [3] Z. Zhang, J. Li, C. N. Manikopoulos, J. Jorgenson, and J. Ucles, "HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification," in *Proceedings of the 2001 IEEE Workshop on Information Assurance*, IEEE Press, pp. 85-90, New York, June 2001.
- [4] C. Kruegel and T. Toth, "Using Decision Trees to Improve Signature-based Intrusion Detection," in *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, pp.389-399, 2003.
- [5] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion Detection Using Neural Networks and Support Vector Machines," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1702-1707, 2002.
- [6] J. Gomez and D. D. Gupta, "Evolving Fuzzy Classifiers for Intrusion Detection," in *Proceedings of the 2002 IEEE Workshop on Information Assurance*, IEEE Press, New York, June 2001.
- [7] S. C. Chin, A. Ray, and V. Rajagopalan, "Symbolic Time Series Analysis for Anomaly Detection: A Comparative Evaluation," *Signal Processing*, vol. 85, no. 9, pages 1859-1868, September 2005.
- [8] A. Ray, "Symbolic Dynamic Analysis of Complex Systems for Anomaly Detection," *Signal Processing*, vol. 84, no. 7, pp. 1115-1130, 2004.
- [9] S. T. Sarasamma, Q. A. Zhu, and J. Huff, "Hierarchical Kohonen Net for Anomaly Detection in Network Security," *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 35, no.2, pages 302-312, April 2005.
- [10] N. Ye, Y. Zhang, and C. M. Borrer, "Robustness of the Markov-Chain Model for Cyber-Attack Detection," *IEEE Transactions on Reliability*, vol. 53, no. 1, pages 116-123, 2004.

- [11] A. M. Khatkhate, A. Ray, E. Keller, and S. Chin, "Symbolic Time Series Analysis of Mechanical Systems for Anomaly Detection," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 4, pages 439-447, August 2006.
- [12] R. Agarwal and M. V. Joshi, "PNrule: A New Framework for Learning Classifier Models in Data Mining (A Case-Study in Network Intrusion Detection)," Tech. Report DSTO-GD-0286, Department of Computer Science, University of Minnesota, USA, 2000.
- [13] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2<sup>nd</sup> edition, Wiley Publishers, October 2000.
- [14] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
- [15] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2<sup>nd</sup> Edition, Wiley Publishers, 2006.
- [16] R. Battiti, "Using Mutual Information for Selecting Features in Supervised Neural Net Learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pages 537-550, July 1994.
- [17] R. P. Lippman, D. J. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," in *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX '00)*, IEEE Press, pp. 12-26, South Carolina, USA, January 2000.
- [18] Y. Bouzida and S. Gombault, "Intrusion Detection Using Principal Component Analysis." in *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, Florida, July 2003.
- [19] D. Song, M. I. Heywood, and A. N. Zincir-Heywood, "Training Genetic Programming on Half a Million Patterns: An Example from Anomaly Detection," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pages 225-239, June 2005.
- [20] S. T. Sarasamma and Q. A. Zhu, "Min-Max Hyperellipsoidal Clustering for Anomaly Detection in Network Security," *IEEE Transactions Systems, Man, Cybernetics – Part B*, vol. 36, no. 4, pages 887-901, August 2006.
- [21] G. Qu, S. Hariri, and M. Yousif, "A New Dependency and Correlation Analysis of Features," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 9, pages 1199-1207, September 2005.

- [22] T. Cover and P. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pages 21-27, January 1967.
- [23] S. R. Safavian and D. Landgrebe, "A Survey of Decision Tree Classification Methodology," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pages 660-674, June 1991.
- [24] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate Statistical Analysis of Audit Trails for Host-based Intrusion Detection," *IEEE Transactions on Computers*, vol. 51, no. 7, pages 810-820, 2002.
- [25] C-I. Chang and S. S. Chiang, "Anomaly Detection and Classification for Hyperspectral Imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 6, pp. 1314-1325, June 2002.
- [26] J. Haines, L. Rossey, R. P. Lippman, and R. K. Cunningham, "Extending the DARPA Offline Intrusion Detection Evaluation," in *Proceedings of the DARPA Information Survivability Conference and Exposition*, IEEE Press, California, USA, June 2001.
- [27] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA Off-Line Intrusion Detection Evaluation," in *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID 2000)*, Springer Verlag, pp. 162-182, Toulouse, France, October 2000.
- [28] G. K. Kuchimanachi, V. V. Phoha, K. S. Balagani, and S. R. Gaddam, "Dimension Reduction Using Feature Extraction Methods for Real-time Misuse Detection Systems," in *Proceedings of the IEEE 2004 Information Assurance Workshop*, pp. 195-202, West Point Military Academy, New York, June 2004.
- [29] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Data Mining Researchers," Technical Report HPL-2003-4, HP Labs, 2003.
- [30] J. Huang and C. Ling, "Using AUC and Accuracy in Evaluating Learning Algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pages 299-310, 2005.
- [31] C. K. Chow and C. N. Liu, "Approximating Discrete Probability Distributions with Dependence Trees," *IEEE Transactions on Information Theory*, IT-14, no.3, pages 462-467, 1968.
- [32] J. B. Kruskal Jr., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," in *Proceedings of the American Mathematical Society*, vol. 7, pages 48-50, 1956.

- [33] M. E. Hellman and J. Raviv, "Probability of Error, Equivocation, and the Chernoff Bound," *IEEE Transactions on Information Theory*, vol.16, pages 368-372, May 1970.
- [34] S.K.M. Wong and F.C.S. Poon, "Comments on Approximating Discrete Probability Distributions with Dependence Trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 3, pages 333-335, March 1989.
- [35] H. Avi-Itzhak and T. Diep, "Arbitrarily Tight Upper and Lower Bounds on the Bayesian Probability of Error," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, pages 89-91, January 1996.
- [36] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Learning*, vol. 22, no.1, pages 4-37, January 2000.
- [37] H. Peng, F. Long, and C. Ding, "Feature Selection Based on Mutual Information: Criteria of Max Dependency, Max-Relevance, and Min-Redundancy," *IEEE Transactions on Pattern Analysis and Machine Learning*, vol. 27, no. 8, August 2005.
- [38] A. M. Fraser, "Restructuring Attractors from Scalar Time Series: A Comparison of Singular System and Redundancy Criteria," *Physica D*, vol.34, pages 391-404, 1989.
- [39] W. Li, "Mutual Information Functions Versus Correlation Functions," *Journal of Statistical Physics*, vol. 60, no. 5, pages 823-837, 1990.
- [40] F. Kanaya and K. Nakagawa, "On the Practical Application of Mutual Information for Statistical Decision Making," *IEEE Transactions on Information Theory*, vol. 37, pages 1151-1156, 1991.
- [41] S. Stolfo et al., The Third International Knowledge Discovery and Data Mining Tools Competition (Online). Available at: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
- [42] A. Lazarevic, A. Ozgur, L. Ertöz, J. Srivastava, and V. Kumar, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," in *Proceedings of the SIAM International Conference on Data Mining*, San Francisco, CA, May 2003.
- [43] D. Mutz, F. Valeur, G. Vigna, and C. Kruegel, "Anomalous System Call Detection," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pages 61-93, February 2006.



- [44] S. Kumar and E. H. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection," in *Proceedings of the 17<sup>th</sup> National Computer Security Conference*, pp. 11–21, October 1994.
- [45] M. Thottan and C. Ji, "Anomaly Detection in IP Networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pages 2191–2204, 2003.
- [46] R. A. Maxion and K. M. C. Tan, "Anomaly Detection in Embedded Systems," *IEEE Transactions on Computers*, vol. 5, no. 2, pages 108-120, February 2002.
- [47] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data," in *Proceedings of Data Mining for Security Applications*, Boston, MA.
- [48] G. A. Barreto, J. C. M. Mota, L. G. M. Souza, R. A. Frota, and I. Aguayo, "Condition Monitoring of 3G Cellular Networks Through Competitive Neural Models," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pages 1064-1075, September 2005.
- [49] A. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "Cost-based Modeling and Evaluation for Fraud and Intrusion Detection: Results from the JAM Project," in *Proceedings of the DARPA Information Survivability Conference and Exposition*, vol. 2, pp. 130-144, Hilton Head, SC, 2000.
- [50] U. M. Fayyad and K. B. Irani, "Multi-interval Discretization of Continuous Valued Attributes for Classification Learning," in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022-1027, New York, 1993.