2006

Abstraction, aggregation and recursion for generating accurate and simple classifiers

Dae-Ki Kang
Iowa State University

Abstraction, aggregation and recursion

for generating accurate and simple classifiers

by

Dae-Ki Kang

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Vasant Honavar, Major Professor
Hui-Hsien Chou
Drena L. Dobbs
Dimitris Margaritis
Johnny S. Wong

Iowa State University

Ames, Iowa

2006

DEDICATION

First of all, I express my gratitude to to my adviser Dr. Honavar for guiding the research presented in this dissertation. He has been my role model of true scholar and a constant source of motivation, encouragement, and invaluable feedback throughout my Ph.D. research.

I give my warm thanks to Prof. Johnny S. Wong for introducing me to the world of intrusion detection system and for being a close collaborator for me. Also, thanks to Prof. Hui-Hsien Chou, Prof. Drena Dobbs, and Prof. Dimitris Margaritis for being the member of the committee and their help and feedback on my preliminary research.

I cannot thank Adrian Silvescu more for his mentoring, cooperation and thoughtful discussions about various research subjects. I am fortunate that he constantly has motivated me with a lot of interesting research ideas with great enthusiasm. I believe such enthusiasm is a typical property of a man who continually pursues questions. Thanks to Dr. Jun Zhang for kindly helping me with abstraction method and other machine learning concepts. I am grateful to both of them for having encouraged me when I was frustrated in my graduate plan.

I thank Prof. Doina Caragea for her careful and professional proof-reading of my papers and research proposal with enormous important comments. Thanks to Carson Andorf for letting me use the protein sequence data he has worked on. It has been my honor to be a part of Artificial Intelligence lab. Though I don't mention their names here, I truly thank all the students who were present in the lab. and interacted with me.

I would like to dedicate this thesis to my family and my parents. Without the support from my wife Kumjoo Lee and my sons Andrew (Jun-Young) Kang and Anthony (Jun-Seo) Kang, I would not have been able to complete this work. I really miss my late father Hyun-Sik Kang, who was a very intelligent pharmacist, liked the movie "La Strada", and had guided his

young son pursuing natural science and learning English. I thank my mother Mal-Yeo Jeon for having kindly helped me to be an ethical person. Thank to my father-in-law Byung-Min Lee and my mother-in-law Jeong-Ja Jeong for their support and encouragement during my Ph.D. years.

The research described in this thesis was supported in part by grants from the National Science Foundation (NSF 0219699) and the National Institute of Health (NIH GM066387) to Prof. Vasant Honavar.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. GENERAL INTRODUCTION

1.1 Abstract

An important goal of inductive learning is to generate accurate and compact classifiers from data. In a typical inductive learning scenario, instances in a data set are simply represented as ordered tuples of attribute values. In our research, we explore three methodologies to improve the accuracy and compactness of the classifiers: abstraction, aggregation, and recursion.

Firstly, *abstraction* is aimed at the design and analysis of algorithms that generate and deal with taxonomies for the construction of compact and robust classifiers. In many applications of the data-driven knowledge discovery process, taxonomies have been shown to be useful in constructing compact, robust, and comprehensible classifiers. However, in many application domains, human-designed taxonomies are unavailable. We introduce algorithms for automated construction of taxonomies inductively from both structured (such as UCI Repository) and unstructured (such as text and biological sequences) data. We introduce AVT-Learner, an algorithm for automated construction of attribute value taxonomies (AVT) from data, and Word Taxonomy Learner (WTL), an algorithm for automated construction of word taxonomy from text and sequence data. We describe experiments on the UCI data sets and compare the performance of AVT-NBL (an AVT-guided Naive Bayes Learner) with that of the standard Naive Bayes Learner (NBL). Our results show that the AVTs generated by AVT-Learner are competitive with human-generated AVTs (in cases where such AVTs are available). AVT-NBL using AVTs generated by AVT-Learner achieves classification accuracies that are comparable to or higher than those obtained by NBL; and the resulting classifiers are significantly more compact than those generated by NBL. Similarly, our experimental results of WTL and WTNBL on protein localization sequences and Reuters newswire text categorization data sets show that

the proposed algorithms can generate Naive Bayes classifiers that are more compact and often more accurate than those produced by standard Naive Bayes learner for the Multinomial Model.

Secondly, we apply *aggregation* to construct features as a multiset of values for the intrusion detection task. For this task, we propose a bag of system calls representation for system call traces and describe misuse and anomaly detection results on the University of New Mexico (UNM) and MIT Lincoln Lab (MIT LL) system call sequences with the proposed representation. With the feature representation as input, we compare the performance of several machine learning techniques for misuse detection and show experimental results on anomaly detection. The results show that standard machine learning and clustering techniques using the simple bag of system calls representation based on the system call traces generated by the operating system's kernel is effective and often performs better than approaches that use foreign contiguous sequences in detecting intrusive behaviors of compromised processes.

Finally, we construct a set of classifiers by *recursive application* of the Naive Bayes learning algorithms. Naive Bayes (NB) classifier relies on the assumption that the instances in each class can be described by a *single* generative model. This assumption can be restrictive in many real world classification tasks. We describe recursive Naive Bayes learner (RNBL), which relaxes this assumption by constructing a tree of Naive Bayes classifiers for sequence classification, where each individual NB classifier in the tree is based on an event model (one model for each class at each node in the tree). In our experiments on protein sequences, Reuters newswire documents and UC-Irvine benchmark data sets, we observe that RNBL substantially outperforms NB classifier. Furthermore, our experiments on the protein sequences and the text documents show that RNBL outperforms C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion) and yields accuracies that are comparable to those of support vector machines (SVM) using similar information.

1.2 Motivation

1.2.1 Motivation for Abstraction

An important goal of inductive learning is to generate accurate and compact classifiers from data. In a typical inductive learning scenario, instances to be classified are represented as ordered tuples of attribute values. However, attribute values can be grouped together to reflect assumed or actual similarities among the values in a domain of interest or in the context of a specific application. Such a hierarchical grouping of attribute values yields an attribute value taxonomy (AVT).

Hierarchical groupings of attribute values (AVT) are quite common in biological sciences. For example, the Gene Ontology Consortium is developing hierarchical taxonomies for describing many aspects of macromolecular sequence, structure, and function (Ashburner et al., 2000). Undercoffer et al. (Undercoffer et al., 2004) have developed a hierarchical taxonomy which captures the features that are observable or measurable by the target of an attack or by a system of sensors acting on behalf of the target. Several ontologies being developed as part of the Semantic Web related efforts (Shadbolt et al., 2006; Berners-Lee et al., 2001) also capture hierarchical groupings of attribute values. Kohavi and Provost (Kohavi and Provost, 2001) have noted the need to be able to incorporate background knowledge in the form of hierarchies over data attributes in electronic commerce applications of data mining.

There are several reasons for exploiting AVT in learning classifiers from data, perhaps the most important being a preference for comprehensible and simple, yet accurate and robust classifiers (Pazzani et al., 1997) in many practical applications of data mining. The availability of AVT presents the opportunity to learn classification rules that are expressed in terms of *abstract* attribute values leading to simpler, easier-to-comprehend rules that are expressed in terms of hierarchically related values.

Another reason for exploiting AVTs in learning classifiers from data arises from the necessity, in many application domains, for learning from small data sets where there is a greater chance of generating classifiers that over-fit the training data. A common approach used by

statisticians when estimating from small samples involves *shrinkage* (Duda et al., 2000) or grouping attribute values (or more commonly class labels) into bins, when there are too few instances that match any specific attribute value or class label, to estimate the relevant statistics with adequate confidence. Learning algorithms that exploit AVT can potentially perform *shrinkage* automatically thereby yielding robust classifiers. In other words, exploiting information provided by an AVT can be an effective approach to performing regularization to minimize over-fitting (Zhang and Honavar, 2003).

Consequently, several algorithms for learning classifiers from AVTs and data have been proposed in the literature. This work has shown that AVTs can be exploited to improve the accuracy of classification and in many instances, to reduce the complexity and increase the comprehensibility of the resulting classifiers (Dhar and Tuzhilin, 1993; Han and Fu, 1996; Hendler et al., 1996; Taylor et al., 1997; Zhang and Honavar, 2003; Zhang et al., 2002). Most of these algorithms exploit AVTs to represent the information needed for classification at different levels of abstraction.

However, in many domains, AVTs specified by human experts are unavailable. Even when a human-supplied AVT is available, it is interesting to explore whether alternative groupings of attribute values into an AVT might yield more accurate or more compact classifiers. Thus, we explore the problem of automated construction of AVTs from data. In particular, we are interested in AVTs that are useful for generating accurate and compact classifiers.

Furthermore, we extend our exploration to word taxonomy of unstructured data such as text and sequences with the similar arguments.

Word taxonomies present the possibility of learning classification rules that are simpler and easier-to-understand when the terms in the rules are expressed in terms of abstract values. With previous work (Kang et al., 2004; Zhang and Honavar, 2004), abstraction of similar concepts by the means of attribute value taxonomy (AVT) has been shown to be useful in generating concise and accurate classifiers.

Against these backgrounds, we introduce word taxonomy guided Naive Bayes learner for the multinomial event model (WTNBL-MN). WTNBL-MN is a word taxonomy based gener-

alization of the standard Naive Bayes learning algorithm for the multinomial model.

Because word taxonomy is not available in many domains, there is a need for automated construction of word taxonomy. Hence, we describe a word taxonomy learner (WTL) that automatically generates word taxonomy from sequence data by clustering of words based on their class conditional distribution.

To evaluate our algorithms, we conducted experiments using two classification tasks: (a) assigning Reuters newswire articles to categories, (b) and classifying protein sequences in terms of their localization. We used Word Taxonomy Learner (WTL) to generate word taxonomy from the training data. The generated word taxonomy was provided to WTNBL-MN to learn concise Naive Bayes classifiers that used abstract words of word taxonomy.

1.2.2 Motivation for Aggregation

Detection of attempts to compromise the integrity, confidentiality, or availability of computing and communication networks is an extremely challenging problem (Denning, 1987). Most current approaches to the design of intrusion detection systems (IDS) are based on the premise that the actions used in an attempted intrusion can be differentiated from the actions executed by users or processes during the normal operation of the computing and communication networks (Axelsson, 2000; Murali and Rao, 2005). An effective IDS logs actions executed by users or processes for investigation, alerts the system administrator when the monitored activities are indicative of attempted intrusion, and, if appropriate, takes corrective measures e.g., expelling the intruder.

Intrusion detection and prevention generally refers to a broad range of strategies for defending against malicious attacks. Intrusion detection can be categorized into *misuse* detection and *anomaly* detection. Misuse typically is a known attack, e.g., a hacker attempting to break into an email server in a way that IDS has already trained. A misuse detection system tries to model normal and abnormal behavior from known attacks. It works by comparing network traffic, system call sequences, or other features of known attack patterns. An anomaly is something out of the ordinary, e.g., abnormal network traffic which is actually caused by unknown

attacks. An anomaly detection system models *normal* behavior and identifies a behavior as abnormal (or anomalous) if it is sufficiently different from known normal behaviors.

IDS can be classified into those that focus on modeling the behavior of users and those that focus on modeling the behavior of processes (Ghosh and Schwartzbard, 1999). System call data are one of the most common types of data used to model the behavior of processes. Such data can be collected by logging the system calls using operating system utilities e.g. Linux strace or Solaris Basic Security Module (BSM).

There has been a great deal of research on how to design and implement intrusion detection systems. For example, Mukherjee et al (Mukherjee et al., 1994) used a combination of host monitors and network monitors with a centralized director for suspicious system activities in the distributed intrusion detection system (DIDS) project. Because it is difficult to manually specify activities that signal intrusive behavior, there has been much work on adaptive or machine learning or data mining approaches for intrusion detection. Forrest et al (Forrest et al., 1996) worked on the Computer Immunology project and explored approaches inspired by the activities of the immune systems of animals for detecting and defending against intrusions. Subsequently, several groups have explored data mining approaches for intrusion detection (Lee et al., 1999; Helmer et al., 2001; Eskin et al., 2002; Campos and Milenova, 2005).

In most IDS that model the behavior of processes, intrusions are detected by observing fixed-length, contiguous subsequences of system calls. For example, in anomaly detection, subsequences of input traces are matched against normal sequences in database so that foreign sequences (Forrest et al., 1996; Hofmeyr et al., 1998) are detected. One potential drawback of this approach is that the size of the database that contains fixed-length contiguous subsequences increases exponentially with the length of the subsequences. For example, if the number of system calls is 200 and the length of the subsequences is 6, the size of the database is theoretically $200^6 = 64 \times 10^{12}$. In practice, only normal subsequences are stored, so actual database size is smaller, but still considerably bigger than the hypothesis size generated by our approach.

Against this background, we explore an alternative representation of system call traces

for intrusion detection. Specifically, we use a *bag of system calls* representation of system call sequences. In other words, we consider intrusion detection of system call sequence as a classification problem on a bag of system calls obtained from the system call sequences. With those problem setting, we constructed and evaluated decision tree (Quinlan, 1993), Naive Bayes (McCallum and Nigam, 1998), decision list (Rivest, 1987; Cohen, 1995), Support Vector Machines (SVM) (Cortes and Vapnik, 1995; Platt, 1999), and Logistic Regression (with a ridge estimator) (Cessie and Houwelingen, 1992) classifiers using bag of system calls representation of system calls for misuse detection. We also explored an approach to anomaly detection using a one class Naive Bayes classifier as well as K-means clustering (Bishop, 1996) using the same representation of system call sequences. Bag of words model is already popular in text classification and categorization area (Mitchell, 1997), and our motivation is to investigate the usefulness of the model in intrusion detection tasks.

1.2.3 Motivation for Recursion

Naive Bayes (NB) classifiers, due to their simplicity and modest computational and training data requirements, are among the most widely used classifiers on many classification tasks, including text classification tasks (McCallum and Nigam, 1998) and macromolecular sequence classification tasks that arise in bio-informatics applications (Andorf et al., 2004). NB classifiers belong to the family of generative models (a model for generating data given a class) for classification. Instances of a class are assumed to be generated by a random process which is modeled by a generative model. The parameters of the generative model are estimated (in the case of NB) assuming independence among the attributes given the class. New instances to be classified are assigned to the class that is the most probable for the instance.

NB classifier relies on the assumption that the instances in each class can be described by a *single* generative model (i.e., probability distribution). According to Langley (Langley, 1993), this assumption can be restrictive in many real world classification tasks. One way to overcome this limitation while maintaining some of the computational advantages of NB classifiers is to construct a tree of NB classifiers. Each node in the tree (a NB classifier) corresponds to one

set of generative models (one generative model per class), with different nodes in the tree corresponding to different generative models for a given class. Langley described a recursive NB classifier (RBC) for classifying instances that are represented by ordered tuples of nominal attribute values. RBC works analogous to a decision tree learner (Quinlan, 1993), recursively partitioning the training set at each node in the tree until the NB classifier of the node simply cannot partition the corresponding data set. Unlike in the case of the standard decision tree, the branches out of each node correspond to the most likely class lebels assigned by the NB classifier at that node. In cases where each class cannot be accurately modeled by a single Naive Bayes generative model, the subset of instances routed to one or more branches belong to more than one class. RBC models the distribution of instances in a class at each node using a Naive Bayes generative model. However, according to Langley's reports of experiments on some of the UC-Irvine benchmark data sets, the recursive NB classifier did not yield significant improvements over standard NB classifier (Langley, 1993).

We revisit the idea of recursive NB classifier in the context of text/sequence classification tasks and most of the UC-Irvine benchmark data sets. We describe RNBL, an algorithm for constructing a tree of Naive Bayes classifiers for sequence classification with two different event models and two stopping criteria. For text and sequence classification, each NB classifier in the tree is based on a multinomial event model (McCallum and Nigam, 1998) (one for each class at each node in the tree). Our choice of the multinomial event model is influenced by its reported advantages over the multivariate event model of sequences (McCallum and Nigam, 1998) in text classification tasks. For UC-Irvine benchmark data sets, RNBL uses multivariate event model. RNBL works in a manner similar to Langley's RBC, recursively partitioning the training set of labeled sequences at each node in the tree until a stopping criterion is satisfied. The branches out of each node correspond to the most likely class assigned by the NB classifier at that node. As for the stopping criterion, RNBL uses either a conditional minimum description length (CMDL) score for the classifier (Friedman et al., 1997) or area under the ROC curve (AUC). The CMDL score is specifically adapted to the case of RNBL based on the CMDL score for the NB classifier using the multinomial event model for sequences (Kang et al., 2005d).

1.3 Our Approach

Conceptually, our research can be recapitulated as follows:

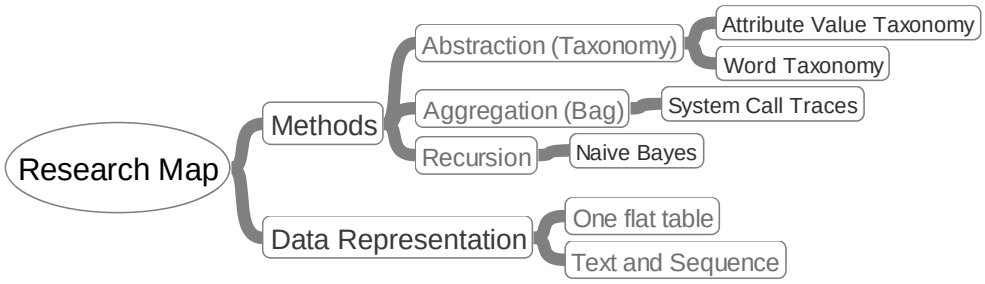


Figure 1.1 Conceptual map of this research

We use abstraction to generate taxonomy from data, aggregation to construct a bag of system calls from sequences for intrusion detection, and recursion to construct a tree of Naive Bayes (NB) classifiers where each individual NB classifier in the tree is associated with a node.

1.3.1 Learning Taxonomy from Data

We invented AVT-Learner, an algorithm for automated construction of attribute value taxonomies (AVT) from data, and Word Taxonomy Learner (WTL) for automated construction of word taxonomy from text and sequence data.

We have introduced AVT-Learner and WTL, algorithms for automated construction of taxonomies from data. The algorithm uses hierarchical agglomerative clustering (HAC) to cluster values based on the distribution of classes that co-occur with them. We have performed experiments on various benchmark data sets such as UCI data sets, text data sets (Reuters newswire articles), and protein sequences (with localization labels). The experimental results indicate that the proposed algorithm can generate classifiers that are more compact and often more accurate than those produced by standard machine learning algorithms. The results also show that the taxonomies generated by AVT-Learner are competitive with taxonomies made by human experts (in cases where such taxonomies are available).

To extend our exploration of taxonomy to unstructured data such as text and sequences

with the similar arguments, we devised word taxonomy guided Naive Bayes learner for the multinomial event model (WTNBL-MN). WTNBL-MN is a word taxonomy based generalization of the standard Naive Bayes learning algorithm for the multinomial model.

Thorough explanation on the algorithms and their experimental results are discussed in chapter 2.

1.3.2 Intrusion Detection Using a Bag of System Calls

This is to generate effective features for accurate classifiers by aggregation based on a bag of values

We have proposed a "bag of system calls" representation for intrusion detection in system call sequences and describe misuse and anomaly detection results with standard machine learning techniques on two benchmark data sets with the proposed representation. With the feature representation as input, we have compared the performance of several machine learning techniques for misuse detection and show experimental results on anomaly detection. The results show that standard machine learning and clustering techniques on simple "bag of system calls" representation of system call sequences is effective and often performs better than those approaches that use foreign contiguous subsequences in detecting intrusive behaviors of compromised processes.

Chapter 3 provides detailed explanation on the algorithms and their experimental results.

1.3.3 Recursive Naive Bayes Learner

We have designed RNBL, a recursive Naive Bayes learner which relaxes the assumption that the instances in each class can be described by a *single* generative model by constructing a tree of Naive Bayes (NB) classifiers for sequence classification where each individual NB classifier in the tree is based on a multinomial generative model (one for each class at each node in the tree). Contrary to previous reports by Langley (Langley, 1993) in the case of a recursive NB classifier (RBC) for the data sets of which the instances are represented as tuples of nominal attribute values, we observe on protein sequence and text classification tasks,

RNBL substantially outperforms NB classifier. Furthermore, our experiments show that RNBL outperforms C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion) and yields accuracies that are comparable to that of support vector machine (SVM) on text/sequence classification.

A detailed discussion can be found in chapter 4.

1.4 A Survey of Related Studies

1.4.1 Related Work on Learning Taxonomy

Gibson and Kleinberg (Gibson, 1988) introduced STIRR, an iterative algorithm based on non-linear dynamic systems for clustering categorical attributes. Ganti et. al. (Ganti et al., 1999) designed CACTUS, an algorithm that uses intra-attribute summaries to cluster attribute values. Zaki et al. (Zaki et al., 2005) presented CLICKS algorithm that finds clusters in categorical datasets based on a search for k-partite maximal cliques. However, all of them did not make taxonomies and use the generated for improving classification tasks.

Pereira et. al. (Pereira et al., 1993) described distributional clustering for grouping words based on class distributions associated with the words in text classification. Yamazaki et al., (Yamazaki et al., 1995) described an algorithm for extracting hierarchical groupings from rules learned by FOCL (an inductive learning algorithm) (Pazzani and Kibler, 1992) and reported improved performance on learning translation rules from examples in a natural language processing task. Slonim and Tishby (Slonim et al., 2006) described a technique (called the agglomerative information bottleneck method) which extended the distributional clustering approach described by Pereira et al. (Pereira et al., 1993), using Jensen-Shannon divergence for measuring distance between document class distributions associated with words and applied it to a text classification task. Baker and McCallum (Baker and McCallum, 1998) reported improved performance on text classification using a technique similar to distributional clustering and a distance measure, which upon closer examination, can be shown to be equivalent to Jensen-Shannon divergence (Slonim et al., 2005, 2006). Dimitropoulos et al. (Dimitropoulos et al., 2006) augmented Internet Autonomous System (AS) Taxonomy to the data set for ASes

classification and successfully classified 95.3% of ASes with expected accuracy of 78.1%.

To the best of our knowledge, there has been little work on the evaluation of techniques for generating hierarchical groupings of attribute values (AVTs) on classification tasks using a broad range of benchmark data sets using algorithms such as AVT-DTL or AVT-NBL that are capable of exploiting AVTs in learning classifiers from data.

1.4.2 Related Work on Intrusion Detection Using a Bag of System Calls

Liao and Vermuri (Liao and Vemuri, 2002) used k-Nearest Neighbor (kNN) algorithm to classify normal and intrusive system call traces. They tested the kNN classifier on 1998 MIT Lincoln Lab BSM data and obtained effective detection rate and low false positive rate. However, they did not perform the detailed analysis with various machine learning techniques and multiple data sets. Their algorithm did not generate comprehensible rule sets for intrusive programs, which is very important for intrusion detection system and analysis.

Warrender, Forrest, and Pearlmutter (Warrender et al., 1999) have presented several intrusion detection methods based upon system call trace data. They tested a method that utilizes sliding windows to determine a database of *normal* sequences to form a database for testing against test instances. They then used a similar method to compare windows in the test instances against the database and classify instances according to a function of the similarity of these sequences to those in the *normal* sequence database. The function requires sequential analysis of a window of system calls for each call made by a process. This requires the maintenance of a large database of *normal* system call trace sequences.

The same authors have described a rule-based classification method that requires alterations to the training data to learn. This model involves prediction of the next system call to be made by a process given some number of calls made immediately before. This method requires enumeration of all unique system call traces within a given program. This is quite demanding on a learner, especially in a situation where the datasets are quite large indeed. Even the space requirements are quite large relative to the input dataset. Finally, classification time is high for such methods because (in the worst case) each rule needs to be checked for each input

instance.

Warrender et al. have presented Hidden Markov Model (HMM) methods for intrusion detection. Although this method does not require modification of the input dataset, it does require individual examination of each dataset to determine the optimal HMM to attempt to learn in each case. While this requirement does not seem overly demanding, we would prefer a method which allows classification of multiple input datasets in the same format if possible. Additionally construction of accurate HMM models can be quite demanding in terms the amount of training data as well as computational effort. Warrender, et al. observe that, for a process that makes S system calls, S states (and thus $2S^2$ values) must be computed. Datasets of interest in practice contain large amounts of processes (eight hours per day worth in the case of the MIT Lincoln Labs datasets), and each process makes a large number of system calls throughout its lifetime. Computing even polynomially many values for each instance becomes a problem at this scale.

Normalized frequency of audit data was used in SRI NIDES (Anderson et al., 1995). In NIDES, probability distribution of long term behavior of a program is generated and maintained as its profile. For detecting the anomalous behavior of the program, the profile is compared with short term behavior of the program, which is also maintained as probability distribution, using a statistical test similar to χ^2 test. The behavior of a program is characterized by its audit data such as file access, CPU usage, etc. We maintain the raw count of system calls that are sequentially observed from the program as its profile, but this approach can be applied to other types of audit data. In some machine learning algorithms, raw counts are normalized and statistically compared with new behavior of the program. The Naive Bayes learning algorithm, which is one of the learning algorithms reported in this study, generates class-conditional probability distributions and prior distributions of the raw counts and statistically compares them with new distribution from the new behavior of the program. Moreover, as we showed, our profile representation can be used effectively with various machine learning algorithms.

One of the most popular rule induction techniques used in IDS is Repeated Incremental

Pruning to Produce Error Reduction (RIPPER) rule learning algorithm (Cohen, 1995). Lee et al. (Lee and Stolfo, 1998) used RIPPER on a set of substrings of length 7 generated by the sliding window from *sendmail* system call traces. The generated rules are based on the insight that intrusion can be captured from the fixed-length substrings. For example, the rule *'normal:* $p_2 = 104, p_7 = 112$*'* means *'if* p_2 *is 104 and* p_7 *is 112 then the substring is normal'.* This approach, as in the case of *STIDE*, employs a user-supplied threshold to determine if the input trace is normal or intrusive. We applied RIPPER on a bag of system calls representation, and we obtained rules based on counts such as *'(count(fcntl)* $\geq$ *1) and (count(rename)* $\leq$ *0) and (count(read)* $\geq$ *5)* $\rightarrow$ *class=intrusion'* where *count(X)* returns the number of occurrence of system call X in the input trace. The rules generated by our method apply to the entire system call trace (as opposed to fixed length substring of traces). In our case, the relevant thresholds are learned directly from the training data, thereby avoiding the necessity of user-supplied thresholds.

Supervised learning techniques like Multi Layer Perceptron (MLP) with Error Back Propagation (Ghosh and Schwartzbard, 1999) have been investigated, as have unsupervised learning techniques like Self-Organized Feature Map (SOM) (Gunes Kayacik, 2003). Kang et al. (Kang et al., 2005a) used principal component analysis (PCA) and time-delay neural network (TDNN) for mutated attacks. In their model, a network packet is considered as a gray level image where each byte of a packet is represented a pixel. Chebrolua et al. (Chebrolua et al., 2005) used Bayesian Network and Classification and Regression Trees (CART) to detect important features for intrusion detection. Spencer (Spencer, 2005) used artificial neural network to detect anomalies in wireless devices. Jiang et al. (Jiang et al., 2005) combined neural network with hidden Markov model (HMM) for efficient intrusion detection. Cha et al. (Cha et al., 2005) designed neural network system using Soundex algorithm to find anomalous behavior patterns. Xu and Xie (Xu and Xie, 2005) introduced Markov reward process model for the behavior of the system call sequences and converted the intrusion detection to predicting the value function of the Markov reward process. Yang et al. (Yang et al., 2005) designed intrusion detection system based on radial basis function (RBF) and compared the performance with back prop-

agation network. Yang (Yang, 2005) viewed intrusion detection task as a case of data mining applied to time series. He used autoregressive moving average (ARMA) and Hopfield models to analyze the time series. Gao et al. (Gao et al., 2005) proposed a method of applying principal component neural networks for intrusion feature extraction. The extracted features are employed by SVM for classification. Using neural networks generally requires the specification of hidden nodes, and the generated model from learning neural network is hard to comprehend. Sy (Sy, 2005) defined the access signature as the collection of the statistically significant association patterns of 4th order using mutual information from the sequence of UNIX command data and used the signature for masquerader detection. Lu et al. (Lu et al., 2005) used several data mining techniques such as clustering, classification, and association rules to maximize the effectiveness in identifying attacks, thereby helping the users to construct more secure information systems. Jiang et al. (Jiang et al., 2006) proposed a novel method to compute the cluster radius threshold. They perform the data classification by an improved nearest neighbor (INN) method and presented a powerful clustering-based method for the unsupervised intrusion detection (CBUID).

All of these approaches use n-gram representation for modeling intrusion, but our approaches uses a bag of system calls representation.

Peddabachigaria et al. (Peddabachigaria et al., 2005) modeled intrusion detection system using decision tree and support vector machines (SVM). Their hybrid system combined classifiers to maximize the accuracy and had more accurate results. One class support vector machines (OCSVM) which can be useful in learning unlabeled data sets, are used for supervised anomaly detection by a few researchers. Heller et al. (Heller et al., 2003) compared OCSVM with probabilistic anomaly detection (PAD) algorithm for Windows Registry data, and concluded that well-defined kernels are important to enhance the performance of OCSVM. Lee et al. (Lee et al., 2005) proposed Multi-step Multi-class Intrusion Detection System (MMIDS), which alleviates some drawbacks associated with misuse detection and anomaly detection. The MMIDS consists of a hierarchical structure of one-class SVM, novel multi-class SVM, and incremental clustering algorithm: Fuzzy-ART. Yilmazel et al. (Yilmazel et al., 2005) compared bag

of words representation (BOW) and NLP based representation for both typical and one-class classification problem using SVM algorithm. Ma et al. (Ma et al., 2005) implemented multi-class SVMs (one-versus-rest, one-versus-rest method and a new Decision Tree (DT) SVM) for intrusion detection. They also applied a support vector (SV) reduction algorithm and found that it decreases the training time dramatically while improves the detection rate. Steinwart et al. (Steinwart et al., 2005) interpreted anomaly detection as a binary classification problem of finding level sets for the data generating density. They compared the corresponding classification risk with the standard performance measure for the density level problem, and found that the empirical classification risk can serve as an empirical performance measure for the anomaly detection. According to the interpretation, they proposed a support vector machine (SVM) for anomaly detection and compared their SVM to other commonly used methods including the standard one-class SVM.

For protein classification, Leslie et al. introduced spectrum kernel (Leslie et al., 2002a) and mismatch kernel (Leslie et al., 2002b). Spectrum kernel is for k-length continuous subsequences, and mismatch kernel is similar to spectrum kernel but mismatches are allowed. Tian et al. (Tian et al., 2004) developed string kernel for intrusion detection. Their kernel penalizes non-continuous occurrences and feature map is indexed by all possible subsequences. Our future work includes one class SVM experiment on a bag of words feature representation.

Liu et al. (Liu et al., 2005a) investigated three system-call-based feature representations for "insider threat" and "external threat": n-grams of system call names, histograms of system call names, and individual system calls with associated parameters, and found that none of these representations consistently performs as well when dealing with the internal threat as previous results show for external threat detection.

Recently, string alignment techniques has been used for intrusion detection and worm detection. Using string alignment techniques for intrusion detection is one of our important future work.

Coull et al. (Coull et al., 2003) first proposed bio-informatics techniques for intrusion detection. They used a semi-global alignment and unique scoring function for detecting intrusive

sequences. Takeda (Takeda, 2005) also applied bio-informatics techniques for network intrusion detection. Tripp (Tripp, 2005) describes a finite state machine approach to string matching for an intrusion detection system. To obtain high performance, he designed a hardware for a parallel string matching. Newsome et al. (Newsome et al., 2005) used an adaptation of the Smith-Waterman (Smith and Waterman, 1981) algorithm to find an alignment for generating signatures, which are applied to match polymorphic worm payloads. Tang and Chen (Tang and Chen, 2005) introduced position-aware distribution signature (PADS), which fits in the gap between the traditional signatures and the anomaly-based systems, and proposed two algorithms based on Expectation-Maximization (EM) and Gibbs Sampling for efficient computation of PADS from polymorphic worm samples. Jiang and Xu (Jiang and Xu, 2005) proposed behavioral footprinting. They modeled each infection step as a behavior phenotype and the entire infection session as a sequential behavioral footprint, and presented advanced sequence analysis techniques to extract a worm's behavioral footprint from its infection traces

1.4.3 Related Work on Recursive Naive Bayes Learner

As noted earlier, Langley (Langley, 1993) investigated recursive Bayesian classifiers for the instances described by tuples of nominal attribute values. RNBL reported in this study applies to not only the data of such kind, but also text/sequence data with multivariate/multinomial event models.

There have been research work on relaxing the independence assumption of a Naive Bayes learning algorithm. Kohavi (Kohavi, 1996) introduced NBTree algorithm, a hybrid of a decision tree and Naive Bayes classifiers for instances represented using tuples of nominal attributes. NBTree evaluates the attributes available at each node to decide whether to continue building a decision tree or to terminate with a Naive Bayes classifier. In contrast, RNBL algorithm, like Langley's RBC, builds a decision tree, whose nodes are all Naive Bayes Classifiers.

Webb et al. (Webb et al., 2005) proposed an approach to improve the accuracy of Naive Bayes by weakening its attribute independence assumption by averaging all of a constrained class of classifiers. Langseth and Nielsen (Langseth and Nielsen, 2006) focus on a relatively

new set of models, termed Hierarchical Naive Bayes models. Hierarchical Naive Bayes models extend the modeling flexibility of Naive Bayes models by introducing latent variables to relax some of the independence statements in these models. Liu et al. (Liu et al., 2005b) propose an algorithm named Graph-NB, which upgrades Naive Bayesian classifier to deal with multiple tables directly. In order to take advantage of linkage relationships among tables, and treat different tables linked to the target table differently, a semantic relationship graph is developed to describe the relationship and to avoid unnecessary joins.

Gama and Brazdil (Gama and Brazdil, 2000) proposed an algorithm that generates a cascade of classifiers. Their algorithm combines Naive Bayes, C4.5 decision tree and linear discriminants, and introduces a new attribute at each stage of the cascade. Gama (Gama, 2001) proposed an algorithm for multivariate tree learning that combines a univariate decision tree with a discriminant function by means of constructive induction. The algorithm uses Linear Bayes classifier for constructing new attributes. For growing trees, the algorithm builds multivariate decision nodes, and for pruning, the algorithm builds functional decision nodes. In both approaches, they performed experiments on several UC-Irvine benchmark data sets (Blake and Merz, 1998) for classifying instances represented as tuples of nominal attribute values. RNBL also recursively applies only the Naive Bayes classifier based on the multivariate/multinomial event models for text and sequences.

Area under the curve (AUC) has been used for the evaluation of prediction ability of the learning algorithms. Huang and Ling (Huang and Ling, 2005) established formal criteria for comparing AUC and accuracy for learning algorithms and showed theoretically and empirically that AUC is a better measure (defined precisely) than accuracy. Brefeld and Scheffer (Brefeld and Scheffer, 2005) presented a rigorous derivation of an AUC maximizing Support Vector Machine (SVM). In our research, we maximize AUC for recursive Naive Bayes learner.

1.5 Dissertation Organization

In this study, we aim to explore the approaches to generate accurate and compact classifiers from data. Following is the outline of this study.

The rest of this dissertation is organized as follows:

Chapter 1: This chapter presents the problems we address, a survey of current studies, and the outline of the dissertation.

Chapter 2: We developed algorithms for automated construction of taxonomies inductively from both structured (such as UCI Repository) and unstructured (such as text and biological sequences) data. AVT-Learner is an algorithm for automated construction of attribute value taxonomies (AVT) from data, and Word Taxonomy Learner (WTL) is for automated construction of word taxonomy from text and sequence data. AVT-Learner and WTL use Hierarchical Agglomerative Clustering (HAC) to cluster attribute values based on the distribution of classes that co-occur with the values. For WTL, we also devised Word Taxonomy guided Naive Bayes Learner for the Multinomial Event Model (WTNBL-MN) that exploits word taxonomy to generate compact classifiers. WTNBL-MN is a generalization of the Naive Bayes learner for the Multinomial Event Model for learning classifiers from data using word taxonomy. We describe experiments on UCI data sets that compare the performance of AVT-NBL (an AVT-guided Naive Bayes Learner) with that of the standard Naive Bayes Learner (NBL) applied to the original data set. The result have been published in in IEEE International Conference on Data Mining in 2004 (Kang et al., 2004) and Symposium on Abstraction, Reformulation, and Approximation in 2005 (Kang et al., 2005d).

Chapter 3: We propose a *bag of system calls* representation for intrusion detection in system call sequences and describe misuse and anomaly detection results with standard machine learning techniques on University of New Mexico (UNM) and MIT Lincoln Lab (MIT LL) system call sequences with the proposed representation. With the feature representation as input, we compare the performance of several machine learning techniques for misuse detection and show experimental results on anomaly detection. The results show that standard machine learning and clustering techniques on simple bag of system calls representation of system call sequences in the operating system's kernel is effective and often performs better than those approaches that use foreign contiguous sequences in detecting intrusive behaviors of compromised processes. We published the results in IEEE International Conference on Intelligence

and Security Informatics in 2005 (Kang et al., 2005c) and IEEE Systems Man and Cybernetics Information Assurance Workshop (Kang et al., 2005b).

Chapter 4: We describe recursive Naive Bayes learner (RNBL), which relaxes this assumption by constructing a tree of Naive Bayes classifiers for sequence classification, where each individual NB classifier in the tree is based on an event model (one model for each class at each node in the tree). In our experiments on protein sequences, Reuters newswire documents and UC-Irvine benchmark data sets, we observe that RNBL substantially outperforms NB classifier. Furthermore, our experiments on the protein sequences and the text documents show that RNBL outperforms C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion) and yields accuracies that are comparable to those of support vector machines (SVM) using similar information. We had published preliminary results in the Tenth Pacific-Asia Conference on Knowledge Discovery and Data Mining (Kang et al., 2006).

Chapter 5: presents the summary of this study, contributions and the future work of my dissertation.

CHAPTER 2. LEARNING TAXONOMIES FROM DATA

This chapter is an extended version of the papers published in IEEE International Conference on Data Mining in 2004 (Kang et al., 2004) and Symposium on Abstraction, Reformulation, and Approximation in 2005 (Kang et al., 2005d).

2.1 Abstract

Taxonomies have been shown to be useful in constructing compact, robust, and comprehensible classifiers. However, in many application domains, human-designed taxonomies are unavailable. We introduce algorithms for automated construction of taxonomies inductively from both structured (such as UCI Repository) and unstructured (such as text and biological sequences) data. We invented AVT-Learner, an algorithm for automated construction of attribute value taxonomies (AVT) from data, and Word Taxonomy Learner (WTL) for automated construction of word taxonomy from text and sequence data. AVT-Learner and WTL use Hierarchical Agglomerative Clustering (HAC) to cluster attribute values based on the distribution of classes that co-occur with the values. For WTL, we also devised Word Taxonomy guided Naive Bayes Learner for the Multinomial Event Model (WTNBL-MN) that exploits word taxonomy to generate compact classifiers. WTNBL-MN is a generalization of the Naive Bayes learner for the Multinomial Event Model for learning classifiers from data using word taxonomy. We describe experiments on UCI data sets that compare the performance of AVT-NBL (an AVT-guided Naive Bayes Learner) with that of the standard Naive Bayes Learner (NBL) applied to the original data set. Our results show that the AVTs generated by AVT-Learner are competitive with human-generated AVTs (in cases where such AVTs are available). AVT-NBL using AVTs generated by AVT-Learner achieves classification accuracies

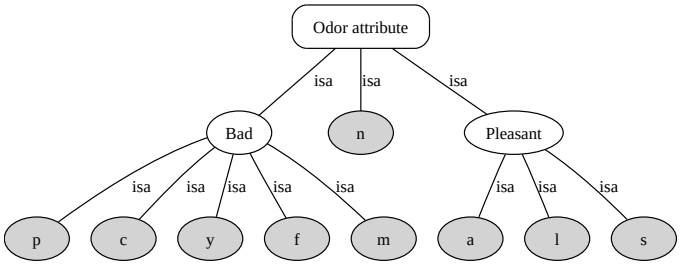


Figure 2.1 Human-made AVT from 'odor' attribute of UCI AGARICUS-LEPIOTA mushroom data set.

that are comparable to or higher than those obtained by NBL; and the resulting classifiers are significantly more compact than those generated by NBL. Similarly, our experimental results of WTL and WTNBL on protein localization sequences and Reuters text show that the proposed algorithms can generate Naive Bayes classifiers that are more compact and often more accurate than those produced by standard Naive Bayes learner for the Multinomial Model.

2.2 Introduction

An important goal of inductive learning is to generate accurate and compact classifiers from data. In a typical inductive learning scenario, instances to be classified are represented as ordered tuples of attribute values. However, attribute values can be grouped together to reflect assumed or actual similarities among the values in a domain of interest or in the context of a specific application. Such a hierarchical grouping of attribute values yields an attribute value taxonomy (AVT). For example, Figure 2.1 shows a human-made taxonomy associated with the nominal attribute *'Odor'* of the UC Irvine AGARICUS-LEPIOTA mushroom data set (Blake and Merz, 1998).

Hierarchical groupings of attribute values (AVT) are quite common in biological sciences. For example, the Gene Ontology Consortium is developing hierarchical taxonomies for describing many aspects of macromolecular sequence, structure, and function (Ashburner et al., 2000). Undercoffer et al. (Undercoffer et al., 2004) have developed a hierarchical taxonomy which captures the features that are observable or measurable by the target of an attack or

by a system of sensors acting on behalf of the target. Several ontologies being developed as part of the Semantic Web related efforts (Shadbolt et al., 2006; Berners-Lee et al., 2001) also capture hierarchical groupings of attribute values. Kohavi and Provost (Kohavi and Provost, 2001) have noted the need to be able to incorporate background knowledge in the form of hierarchies over data attributes in electronic commerce applications of data mining.

There are several reasons for exploiting AVT in learning classifiers from data, perhaps the most important being a preference for comprehensible and simple, yet accurate and robust classifiers (Pazzani et al., 1997) in many practical applications of data mining. The availability of AVT presents the opportunity to learn classification rules that are expressed in terms of *abstract* attribute values leading to simpler, easier-to-comprehend rules that are expressed in terms of hierarchically related values. Thus, the rule $(odor = pleasant) \rightarrow (class = edible)$ is likely to be preferred over $((odor = a) \wedge (color = brown)) \vee ((odor = l) \wedge (color = brown)) \vee ((odor = s) \wedge (color = brown)) \rightarrow (class = edible)$ by a user who is familiar with the odor taxonomy shown in Figure 2.1.

Another reason for exploiting AVTs in learning classifiers from data arises from the necessity, in many application domains, for learning from small data sets where there is a greater chance of generating classifiers that over-fit the training data. A common approach used by statisticians when estimating from small samples involves *shrinkage* (Duda et al., 2000) or grouping attribute values (or more commonly class labels) into bins, when there are too few instances that match any specific attribute value or class label, to estimate the relevant statistics with adequate confidence. Learning algorithms that exploit AVT can potentially perform *shrinkage* automatically thereby yielding robust classifiers. In other words, exploiting information provided by an AVT can be an effective approach to performing regularization to minimize over-fitting (Zhang and Honavar, 2003).

Consequently, several algorithms for learning classifiers from AVTs and data have been proposed in the literature. This work has shown that AVTs can be exploited to improve the accuracy of classification and in many instances, to reduce the complexity and increase the comprehensibility of the resulting classifiers (Dhar and Tuzhilin, 1993; Han and Fu, 1996;

Hendler et al., 1996; Taylor et al., 1997; Zhang and Honavar, 2003; Zhang et al., 2002). Most of these algorithms exploit AVTs to represent the information needed for classification at different levels of abstraction.

However, in many domains, AVTs specified by human experts are unavailable. Even when a human-supplied AVT is available, it is interesting to explore whether alternative groupings of attribute values into an AVT might yield more accurate or more compact classifiers. Against this background, we explore the problem of automated construction of AVTs from data. In particular, we are interested in AVTs that are useful for generating accurate and compact classifiers.

Furthermore, we extend our exploration to word taxonomy of unstructured data such as text and sequences with the similar arguments.

Word taxonomies present the possibility of learning classification rules that are simpler and easier-to-understand when the terms in the rules are expressed in terms of abstract values. With previous work (Kang et al., 2004; Zhang and Honavar, 2004), abstraction of similar concepts by the means of attribute value taxonomy (AVT) has been shown to be useful in generating concise and accurate classifiers.

Against this background, we introduce word taxonomy guided Naive Bayes learner for the multinomial event model (WTNBL-MN). WTNBL-MN is a word taxonomy based generalization of the standard Naive Bayes learning algorithm for the multinomial model.

Because word taxonomy is not available in many domains, there is a need for automated construction of word taxonomy. Hence, we describe a word taxonomy learner (WTL) that automatically generates word taxonomy from sequence data by clustering of words based on their class conditional distribution.

To evaluate our algorithms, we conducted experiments using two classification tasks: (a) assigning Reuters newswire articles to categories, (b) and classifying protein sequences in terms of their localization. We used Word Taxonomy Learner (WTL) to generate word taxonomy from the training data. The generated word taxonomy was provided to WTNBL-MN to learn concise Naive Bayes classifiers that used abstract words of word taxonomy.

2.3 Learning Taxonomies from Data

We start with definitions of preliminary concepts necessary to describe our algorithms. We then precisely define the problem as learning classifier from taxonomy and data.

2.3.1 Definition of Attribute Value Taxonomy (AVT)

Let $A = \{A_1, A_2, \ldots, A_n\}$ be a set of nominal attributes. Let $V_i = \left\{v_i^1, v_i^2, \ldots, v_i^{m_i}\right\}$ be a finite domain of mutually exclusive values associated with attribute A_i where v_i^j is the j^{th} attribute value of A_i and m_i is the number possible number of values of A_i, that is, $|V_i|$. We say that V_i is the set of primitive values of attribute A_i. Let $C = \{C_1, C_2, \ldots, C_k\}$ be a set of mutually disjoint class labels. A data set is $D \subseteq V_1 \times V_2 \times \ldots \times V_n \times C$.

Let $T = \{T_1, T_2, \ldots, T_n\}$ denote a set of AVT such that T_i is an AVT associated with the attribute A_i, and $Leaves(T_i)$ denote a set of all leaf nodes in T_i. After Haussler (Haussler, 1988), we define a cut δ_i of an AVT T_i as follows:

Definition 1 (Cut over AVT) *A cut δ_i is a subset of nodes in T_i satisfying the following two properties:*

1. *For any leaf $l \in Leaves(T_i)$, either $l \in \delta_i$ or l is a descendant of a node $n \in \delta_i$.*
2. *For any two nodes $f, g \in \delta_i$, f is neither a descendant nor an ancestor of g.*

For example, $\{Bad, a, l, s, n\}$ is a cut through the AVT for *odor* shown in Figure 2.1. Note that a cut through T_i corresponds to a partition of the values in V_i. Let $\Delta = \{\delta_1, \delta_2, \ldots \delta_n\}$ be a set of cuts associated with AVTs in $T = \{T_1, T_2, \ldots T_n\}$.

Definition 2 (Specialization and Abstraction over AVT) *We say that a cut $\hat{\delta}_i$ is a specialization of a cut δ_i if $\hat{\delta}_i$ is obtained by replacing at least one node $v \in \delta_i$ by its descendants. Conversely, δ_i is an abstraction of $\hat{\delta}_i$*

Figure 2.2 illustrates a specialization process in taxonomy T. The cut $\gamma_2 = \{A, B, C, D\}$ in T_2 has been refined to $\delta = \{A, B_1, B_2, C, D\}$ by replacing B with its two children B_1,B_2,

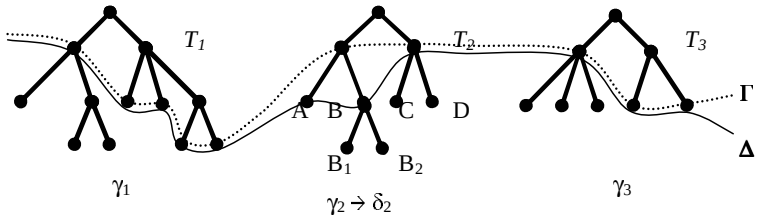


Figure 2.2 Illustration of Cut Specialization over AVT: The cut $\gamma_2 = \{A, B, C, D\}$ in T_2 has been refined to $\delta = \{A, B_1, B_2, C, D\}$ by replacing B with its two children B_1,B_2, and $\delta_1 = \gamma_1$ and $\delta_3 = \gamma_3$. Therefore $\Delta = \{\delta_1, \delta_2, \delta_3\}$ is a specialization of $\Gamma = \{\gamma_1, \gamma_2, \gamma_3\}$.

and $\delta_1 = \gamma_1$ and $\delta_3 = \gamma_3$. Therefore $\Delta = \{\delta_1, \delta_2, \delta_3\}$ is a specialization of $\Gamma = \{\gamma_1, \gamma_2, \gamma_3\}$, and corresponding hypothesis $h(\Delta)$ is a specialization of $h(\Gamma)$.

Definition 3 (Instance Space) *Any choice of Δ defines an input space I_Δ. If there is a node $\in \Delta$ and $\notin$ Leaves(T), the induced input space I_Δ is an abstraction of the original input space I.*

With a data set D, AVT T and corresponding valid cuts, we can extend our definition of instance space to include instance spaces induced from different levels of abstraction of the original input space. Thus, taxonomy guided learning algorithm work on this induced input space.

2.3.2 Definition of Word Taxonomy (WT)

For word taxonomy over the unstructured data such as text documents or sequences, we define abstraction based on the frequency of values associated with the same class label.

Let $\Sigma = \{w_1, w_2, \ldots, w_N\}$ be a dictionary of words, $C = \{c_1, c_2, \ldots, c_M\}$ a finite set of mutually disjoint class labels, and $f_{i,j}$ denote an integer frequency of word w_i in a sequence d_j. Then, sequence d_j is represented as an instance I_j, a frequency vector $< f_{i,j} >$ of w_i, and each sequence belongs to a class label in C. Finally, a data set D is represented as a collection of instance and their associated class label $\{(I_j, c_j)\}$.

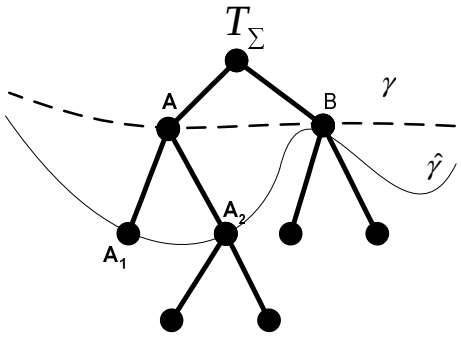


Figure 2.3 Illustration of Cut Specialization: The cut $\gamma = \{A, B\}$ is been refined to $\hat{\gamma} = \{A_1, A_2, B\}$ by replacing A with A_1 and A_2

Let T_Σ be a word taxonomy defined over the possible words of Σ. Let $Nodes(T_\Sigma)$ denote the set of all values in T_Σ and $Root(T_\Sigma)$ denote the root of T_Σ. We represent the set of leaves of T_Σ as $Leaves(T_\Sigma) \subseteq \Sigma$. The internal nodes of the tree correspond to *abstract values* of Σ.

After Haussler (Haussler, 1988), we define a cut γ through a word taxonomy T_Σ as follows.

Definition 4 (Cut over WT) *A cut γ is a subset of nodes in word taxonomy T_Σ satisfying the following two properties:*

1. *For any leaf $l \in Leaves(T_\Sigma)$, either $l \in \gamma$ or l is a descendant of a node in T_Σ.*
2. *For any two nodes f,g $\in \gamma$, f is neither a descendant not an ancestor of g.*

A cut γ induces a partition of words in T_Σ.

Definition 5 (Specialization and Abstraction over WT) *We say that a cut $\hat{\gamma}$ is a specialization of a cut γ if $\hat{\gamma}$ is obtained by replacing at least one node $v \in \gamma$ by its descendants. Conversely, γ is an abstraction of $\hat{\gamma}$*

Figure 2.3 illustrates a specialization process in word taxonomy T_Σ. The cut $\gamma = \{A, B\}$ is been refined to $\hat{\gamma} = \{A_1, A_2, B\}$ by replacing A with A_1 and A_2. Thus, corresponding hypothesis $h_{\hat{\gamma}}$ is a specialization of h_γ.

2.3.3 Algorithms of Learning Taxonomies from Data

2.3.3.1 Learning Attribute Value Taxonomies

Firstly, we describe AVT-Learner, an algorithm for automated construction of AVT from a data set of instances wherein each instance is described by an ordered tuple of N nominal attribute values and a class label.

The problem of learning AVTs from data can be stated as follows: given a data set $D \subseteq V_1 \times V_2 \times \ldots \times V_n \times C$ and a measure of dissimilarity (or equivalently similarity) between any pair of values of an attribute, output a set of AVTs $T = \{T_1, T_2, \ldots, T_n\}$ such that each T_i (AVT associated with the attribute A_i) corresponds to a hierarchical grouping of values in V_i based on the specified similarity measure.

We use hierarchical agglomerative clustering (HAC) of the attribute values according to the distribution of classes that co-occur with them. Let $DM\left(P\left(x\right)||P\left(y\right)\right)$ denote a measure of pairwise divergence between two probability distributions $P\left(x\right)$ and $P\left(y\right)$ where the random variables x and y take values from the same domain. We use the pairwise divergence between the distributions of class labels associated with the corresponding attribute values as a measure of the dissimilarity between the attribute values. Thus, two values of an attribute are considered to be more similar to each other than any other pair of values if their class distributions are more similar to each other than the class distributions associated with any other pair of values for the same attribute. The lower the divergence between the class distributions associated with two attributes, the greater is their similarity. The choice of this measure of dissimilarity between attribute values is motivated by the intended use of the AVT, namely, the construction of accurate, compact, and robust classifiers. If two values of an attribute are indistinguishable from each other with respect to their class distributions, they provide statistically similar information for classification of instances.

The algorithm for learning AVT for a nominal attribute is shown in Figure 2.4. The basic idea behind AVT-Learner is to construct an AVT T_i for each attribute A_i by starting with the primitive values in V_i as the leaves of T_i and recursively add nodes to T_i one at a time by merging two existing nodes. To aid this process, the algorithm maintains a cut δ_i through the

AVT T_i, updating the cut δ_i as new nodes are added to T_i. At each step, the two attribute values to be grouped together to obtain an abstract attribute value to be added to T_i are selected from δ_i based on the divergence between the class distributions associated with the corresponding values. That is, a pair of attribute values in δ_i are merged if they have more similar class distributions than any other pair of attribute values in δ_i. This process terminates when the cut δ_i contains a single value which corresponds to the root of T_i. If $|V_i| = m_i$, the resulting T_i will have $(2m_i - 1)$ nodes when the algorithm terminates.

In the case of continuous-valued attributes, we define intervals based on observed values for the attribute in the data set. We then generate a hierarchical grouping of adjacent intervals, selecting at each step two adjacent intervals to merge using the pairwise divergence measure. A cut through the resulting AVT corresponds to a discretization of the continuous-valued attribute. A similar approach can be used to generate AVT from ordinal attribute values.

2.3.3.2 Learning Word Taxonomy

The problem of learning a word taxonomy from sequence data can be stated as follows: Given a data set represented as a set of instances where an instance is a frequency vector $< f_i, c >$ of a word $w_i \in \Sigma$ and associated class label c, and a similarity measure among the words, output word taxonomy T_Σ such that it corresponds to a hierarchical grouping of words in Σ based on the specified similarity measure.

Since this problem is similar to the problem for learning AVT, we take the similar approach of AVT Learner to Word Taxonomy Learner (WTL).

2.3.4 Pairwise Divergence Measures

There are several ways to measure similarity between two probability distributions. We have tested thirteen divergence measures for probability distributions P and Q, which includes J-divergence. Jensen-Shannon divergence, and Arithmetic and geometric mean divergence.

J-divergence (Topsϕe, 2000) also known as Jeffreys-Kullback-Liebler divergence, is a sym-

AVT-Learner :
begin

1. **Input** : data set D
2. For each attribute A_i:
3. For each attribute value v_i^j :
4. For each class label c_k: estimate the probability $p\left(c_k|v_i^j\right)$
5. Let $P\left(C|v_i^j\right) = \left\{p\left(c_1|v_i^j\right), \ldots, p\left(c_k|v_i^j\right)\right\}$ be the class distribution associated with value .
6. Set $\delta_i \leftarrow V_i$; Initialize T_i with nodes in δ_i.
7. Iterate until $|\delta_i| = 1$:
8. In δ_i, find $(x, y) = argmin\left\{DM\left(P\left(C|v_i^x\right)||P\left(C|v_i^y\right)\right)\right\}$
9. Merge v_i^x and v_i^y $(x \neq y)$ to create a new value v_i^{xy}.
10. Calculate probability distribution $P\left(C|v_i^{xy}\right)$.
11. $\lambda_i \leftarrow \delta_i \cup \{v_i^{xy}\} \setminus \{v_i^x, v_i^y\}$.
12. Update T_i by adding nodes v_i^{xy} as a parent of v_i^x and v_i^y.
13. $\delta_i \leftarrow \lambda_i$.
14. **Output** : $T = \{T_1, T_2, \ldots, T_n\}$

end.

Figure 2.4 Pseudo-code of AVT-Learner

metric version of Kullback-Liebler divergence. J-divergence between two probability distributions P and Q is defined as follows:

$$J\left(P||Q\right) = \left(K\left(P||Q\right) + K\left(Q||P\right)\right) = \sum \left(p_i - q_i\right) log\left(\frac{p_i}{q_i}\right)$$

where Kullback-Liebler divergence, variously known as relative information, directed divergence, relative entropy, function of discrimination, is given by:

$$K\left(P||Q\right) = \sum \left(p_i log\left(\frac{p_i}{q_i}\right)\right)$$

Kullback-Liebler divergence is a natural measure for dissimilarity between distributions. It is non-negative and reflexive, but it is asymmetric and doesn't satisfy triangle inequality.

Jensen-Shannon divergence (Slonim et al., 2006) is weighted information gain, also called Jensen difference divergence, information radius, Jensen difference divergence, and Sibson-Burbea-Rao Jensen Shannon divergence. It is given by:

$$I\left(P||Q\right) = \frac{1}{2}\left[\sum p_i log\left(\frac{2p_i}{p_i + q_i}\right) + \sum q_i log\left(\frac{2q_i}{p_i + q_i}\right)\right]$$

Jensen-Shannon divergence is reflexive, symmetric and bounded. Figure 2.5 shows an AVT of 'odor' attribute generated by AVT-Learner (with binary clustering).

Arithmetic and geometric mean divergence (A & G divergence) (Taneja, 1995), popularly known as Backward Jensen Shannon divergence is given by:

$$T\left(P||Q\right) = \sum \left(\frac{p_i + q_i}{2}\right) log\left(\frac{p_i + q_i}{2\sqrt{p_i q_i}}\right)$$

It is the KL divergence between arithmetic mean and geometric mean of two distributions.

Since the results from different symmetric divergence measures do not make a remarkable difference, we limit the discussion to Jensen-Shannon divergence measure in this dissertation.

2.4 Evaluation of Taxonomies

The intuition behind our approach to evaluating the AVT generated by AVT-Learner is the following: an AVT that captures relevant relationships among attribute values can result in the generation of simple and accurate classifiers from data, just as an appropriate choice of

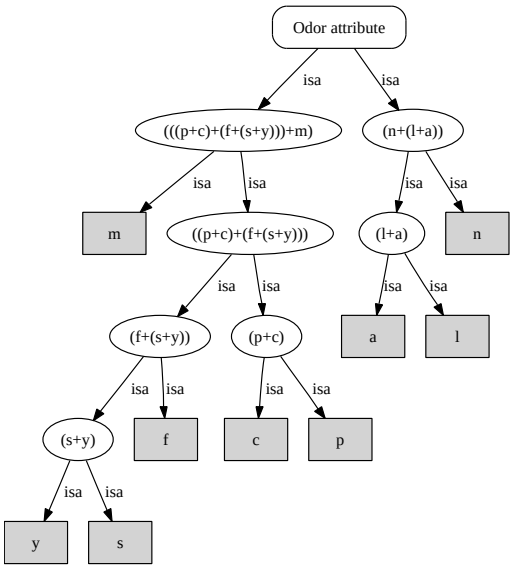


Figure 2.5 AVT of 'odor' attribute of UCI AGARICUS-LEPIOTA mushroom data set generated by AVT-Learner using Jensen-Shannon divergence (binary clustering)

axioms in a mathematical domain can simplify proofs of theorems. Thus, the simplicity and predictive accuracy of the learned classifiers based on alternative choices of AVT can be used to evaluate the utility of the corresponding AVT in specific contexts. Similar arguments are applicable for word taxonomy learner (WTL).

For evaluation, it is necessary to discuss the learning algorithms that can exploit taxonomies. We explain AVT-NBL (Zhang and Honavar, 2004) for structured data and WTNBL-MN for unstructured data.

2.4.1 AVT Guided Variants of Standard Learning Algorithms

It is possible to extend standard learning algorithms in principled ways so as to exploit the information provided by AVT. AVT-DTL (Yamazaki et al., 1995; Zhang et al., 2002; Zhang and Honavar, 2003) and the AVT-NBL (Zhang and Honavar, 2004) which extend the decision tree learning algorithm (Quinlan, 1993) and the Naive Bayes learning algorithm (Langley et al., 1992) respectively are examples such algorithms.

The basic idea behind AVT-NBL is to start with the Naive Bayes Classifier that is based on the most abstract attribute values in AVTs and successively refine the classifier by a scoring function - a Conditional Minimum Description Length (CMDL) score suggested by Friedman et al. (Friedman et al., 1997) to capture trade-off between the accuracy of classification and the complexity of the resulting Naive Bayes classifier.

The experiments reported by Zhang and Honavar (Zhang and Honavar, 2004) using several benchmark data sets show that AVT-NBL is able to learn, using human generated AVT, substantially more accurate classifiers than those produced by Naive Bayes Learner (NBL) applied directly to the data sets as well as NBL applied to data sets represented using a set of binary features that correspond to the nodes of the AVT (PROP-NBL). The classifiers generated by AVT-NBL are substantially more compact than those generated by NBL and PROP-NBL. These results hold across a wide range of missing attribute values in the data sets. Hence, the performance of Naive Bayes classifiers generated by AVT-NBL when supplied with AVT generated by the AVT-Learner provide useful measures of the effectiveness of AVT-Learner in discovering hierarchical groupings of attribute values that are useful in constructing compact and accurate classifiers from data.

2.4.2 WTNBL-MN Algorithm

The problem of learning classifiers from a word taxonomy and sequence data is a natural generalization of the problem of learning classifiers from the sequence data. Original data set D is a collection of labeled instances $< I_j, c_j >$ where $I \in I$. A classifier is a hypothesis in the form of function $h : I \rightarrow C$, whose domain is the instance space I and whose range is the set of class C. A hypothesis space H is a set of hypotheses that can be represented in some hypothesis language or by a parameterized family of functions. Then, the task of learning classifiers from the data set D is induce a hypothesis $h \in H$ that satisfies given criteria.

Hence, the problem of learning classifiers from word taxonomy and data can be described as follows: Given word taxonomy T_Σ over words Σ and a data set D, the aim is induce a classifier $h_{\gamma^*} : I_{\gamma^*} \rightarrow C$ where γ^* is a cut that maximizes given criteria. Of interest here is that

the resulting hypothesis space $H_{\hat{\gamma}}$ of a chosen cut $\hat{\gamma}$ is efficient in searching for both concise and accurate hypothesis.

Word taxonomy guided Naive Bayes Learner is composed of two major components: (a) estimation of parameters of Naive Bayes classifiers based on a cut, (b) and a criterion for specializing or abstracting a cut.

2.4.2.1 Aggregation of Class Conditional Frequency Counts

We can estimate the relevant parameters of a Naive Bayes classifier efficiently by aggregating class conditional frequency counts. For a particular node of a given cut, parameters of the node can be estimated by summing up the class conditional frequency counts of its children (Zhang and Honavar, 2004).

Given word taxonomy T_Σ, we can define a tree of class conditional frequency counts T_f such that there is one-to-one correspondence between the nodes of word taxonomy T_Σ and the nodes of the corresponding T_f. The class conditional frequency counts associated with a non leaf node of T_f is the aggregation of the corresponding class conditional frequency counts associated with its children. Because a cut through word taxonomy corresponds a partition of the set of words, the corresponding cut through T_f specifies a valid class conditional probability table for words. Therefore, to estimate each nodes of T_f, we simply estimate the class conditional frequency counts of primitive words in Σ, which corresponds to the leaves of T_f. Then we aggregate them recursively to calculate the class conditional frequency counts associated with their parent node.

2.4.2.2 Multinomial model for Representing Text/Sequence

In a multinomial model, a sequence is represented as a vector of word occurrence frequencies $f_{i,j}$. The probability of an instance I_j given its class c_j is defined as follows:

$$P(d_j|c_j) = \left\{ \frac{\left(\sum_i^{|\Sigma|} f_{i,j}\right)!}{\prod_i^{|\Sigma|}(f_{i,j})!} \right\} \prod_i^{|\Sigma|} \{p_{i,j}^{f_{i,j}}\} \tag{2.1}$$

The term $\left\{ \frac{\left(\sum_i^{|\Sigma|} f_{i,j}\right)!}{\prod_i^{|\Sigma|}(f_{i,j})!} \right\}$ represents the number of possible combinations of words for the instance I_j.

In equation 2.1, $p_{i,j}$ is basically calculated as follows:

$$p_{i,j} = \frac{Count(c_j, w_i)}{Count(c_j)}$$

$Count(c_j, w_i)$ is the number of times word w_i appears in all the instances that have a class label c_j, and $Count(c_j)$ is the total number of words in a particular class label c_j. With Laplacian smoothing, $p_{i,j}$ will be as follows:

$$p_{i,j} = \frac{1 + Count(c_j, w_i)}{|\Sigma| + Count(c_j)}$$

Or, if we follow the Dirichlet prior, $p_{v_i,c}$ will be as follows:

$$p_{v_i,c} = \frac{\bar{L}/|v| + Count(c, v_i)}{\bar{L} + Count(c)}$$

where, $\bar{L}$ is an average length and $|v|$ is the number of values.

If we consider the number of words in an instance (in other words, document length)(McCallum and Nigam, 1998) but assume that the document length is independent of class for simplicity, we will get the following.

$$P(v|c) = P(d) \left\{ \frac{d!}{\prod_i^{|v|} v_i!} \right\} \prod_i^{|v|} \{p_{v_i,c}^{v_i}\} \tag{2.2}$$

where, $d = \left(\sum_i^{|v|} v_i\right)$, the number of words in a particular instance (document length). In practice, the document length may be class dependent. In practice, we may let the document length class dependent

$$P(v|c) = P(|d||c) \left\{ \frac{d}{\prod_i^{|v|} v_i!} \right\} \prod_i^{|v|} \{p_{v_i,c}^{v_i}\}$$

2.4.2.3 Conditional Minimum Description Length of Naive Bayes Classifier

We use conditional minimum description length (CMDL) (Friedman et al., 1997) score to grade the specialization or abstraction of Naive Bayes classifier for the multinomial model.

Let v_j be a set of attribute values of j^{th} instance $d_j \in D$, and $c_j \in C$ a class label associated with d_j. Then, the conditional log likelihood of the hypothesis B given data D is

$$CLL(B|D) = |D| \sum^{|D|} \log\{P_B(c|v)\} = |D| \sum^{|D|} \log \left\{ \frac{P_B(c)P_B(v|c)}{\sum_{c_i}^{|C|} P_B(c_i)P_B(v|c_i)} \right\} \tag{2.3}$$

For Naive Bayes classifier, this score can be efficiently calculated (Zhang and Honavar, 2004).

$$CLL(B|D) = |D| \sum^{|D|} \log \left\{ \frac{P_B(c) \prod^{v_i \in v} \{P_B(v_i|c)\}}{\sum_{c_i}^{|C|} P_B(c_i) \prod^{v_j \in v} \{P_B(v_j|c_i)\}} \right\}$$

And the corresponding conditional minimum description length (CMDL) score is defined as follows:

$$CMDL(B|D) = -CLL(B|D) + \left\{ \frac{\log |D|}{2} \right\} size(B)$$

where, $size(B)$ is a size of the hypothesis B which corresponds to the number of parameters in conditional probability tables (CPT) of B.

In case of a Naive Bayes classifier with multi-variate Bernoulli model, $size(B)$ is defined as

$$size(B) = (|C| - 1) + |C| \sum_{i=1}^{|v|} (|v_i| - 1)$$

where $|C|$ is the number of class labels, $|v|$ is the number of attributes, and $|v_i|$ is the number of attribute values for an attribute v_i.

2.4.2.4 Conditional Minimum Description Length of a Naive Bayes Classifier for the Multinomial Model

Combining the equations 2.1 and 2.3, we can obtain the conditional log likelihood of the classifier B given data D under the Naive Bayes multinomial model.

$$CLL(B|D) = |D| \sum_{j}^{|D|} \log \left\{ \frac{P(c_j) \left\{ \frac{\left(\sum_i^{|\Sigma|} f_{i,j}\right)!}{\prod_i^{|\Sigma|}(f_{i,j})!} \right\} \prod_i^{|\Sigma|} \{p_{i,j}^{f_{i,j}}\}}{\sum_k^{|C|} \left\{ P(c_k) \left\{ \frac{\left(\sum_i^{|\Sigma|} f_{i,k}\right)!}{\prod_i^{|\Sigma|}(f_{i,k})!} \right\} \prod_i^{|\Sigma|} \{p_{i,k}^{f_{i,k}}\} \right\}} \right\} \tag{2.4}$$

where, $|D|$ is the number of instances, $c_j \in C$ is a class label for instance $d_j \in D$, $f_{i,j}$ is a integer frequency of word $w_i \in \Sigma$ in instance d_j, and $p_{i,j}$ is the estimated probability that word w_i occurred in the instances associated to class label j.

Conditional Minimum Description Length (CMDL) of a Naive Bayes Classifier for the multinomial model is defined as follows:

$$CMDL(B|D) = -CLL(B|D) + \left\{ \frac{\log |D|}{2} \right\} size(B)$$

where, $size(B)$ is a size of the hypothesis B which corresponds to the number of entries in conditional probability tables (CPT) of B.

Therefore, $size(B)$ is estimated as

$$size(B) = (|C| - 1) + |C||\Sigma| \tag{2.5}$$

where $|C|$ is the number of class labels, and $|\Sigma|$ is the cardinality of the vocabulary (i.e. the number of all distinct words).

2.4.2.5 Computation of CMDL score

Because each word is assumed to be independent of others given the class, the search for the word taxonomy guided Naive Bayes classifier can be performed efficiently by optimizing the CMDL criterion independently for each word. Thus, the resulting hypothesis h intuitively trades off the complexity in terms of the number of parameters against the accuracy of classification. The algorithm terminates when none of candidate specialization or abstraction of the classifier yield statistically significant improvement in the CMDL score. Figure 2.6 outlines the algorithm that searches for the optimal cut by specialization in top-down direction.

WTNBL-MN (specialization):
begin

1. **Input** : data set D and word taxonomy T_Σ
2. Initialize a cut γ to the root of T_Σ
3. Estimate probabilities that specify the hypothesis h_γ
4. Repeat until no change in cut γ
5. $\quad \bar{\gamma} \leftarrow \gamma$
6. $\quad$ For each node $v \in \gamma$:
7. $\quad\quad$ Generate a refinement γ^v of γ by replacing v with its children.
8. $\quad\quad$ Construct corresponding hypothesis h_{γ^v}.
9. $\quad\quad$ If $CMDL(h_{\gamma^v}|D) < CMDL(h_{\bar{\gamma}}|D)$, then replace $\bar{\gamma}$ with γ^v.
10. $\quad$ If $\gamma \neq \bar{\gamma}$ then $\gamma \leftarrow \bar{\gamma}$
11. **Output** : h_γ

end.

Figure 2.6 Pseudo-code of Word Taxonomy Guided Naive Bayes Learner for the Multinomial Model(WTNBL-MN)

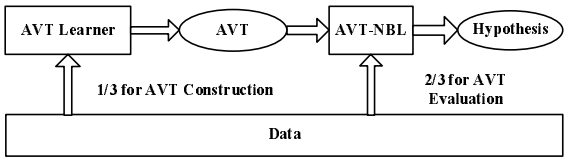


Figure 2.7 Evaluation setup of AVTs with AVT-NBL

2.5 Experiments for AVT

2.5.1 Experimental Setup

Figure 2.7 shows the experimental setup. The AVT generated by the AVT-Learner are evaluated by comparing the performance of the Naive Bayes Classifiers produced by applying

- NBL to the original data set
- AVT-NBL to the original data set (See Figure 2.7).

For the benchmark data sets, we chose 37 data sets from UCI data repository (Blake and Merz, 1998).

Among the data sets we have chosen, AGARICUS-LEPIOTA data set and NURSERY data set have AVT supplied by human experts. AVT for AGARICUS-LEPIOTA data was prepared by a botanist, and AVT for NURSERY data was based on our understanding of the domain. We are not aware of any expert-generated AVTs for other data sets.

The mushroom records in the AGARICUS-LEPIOTA data set were drawn from (Lincoff, 1981) by Jeff Schlimmer in 1987. The data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. The goal of the classifier induced from the data set is discern each species to be definitely edible, definitely poisonous, or of unknown edibility and not recommended. The guide (Lincoff, 1981) stated that there is no simple rule for determining the edibility of a mushroom.

Nursery Database was derived from a hierarchical decision model (Olave et al., 1989) originally developed to rank applications for nursery schools during several years in 1980's when

there was excessive enrollment to these schools in Ljubljana, Slovenia, and the rejected applications frequently needed an objective explanation.

In each experiment, we randomly divided each data set into 3 equal parts and used 1/3 of the data for AVT construction using AVT-Learner. The remaining 2/3 of the data were used for generating and evaluating the classifier. Each set of AVTs generated by the AVT-Learner was evaluated in terms of the error rate and the size of the resulting classifiers (as measured by the number of entries in conditional probability tables). The error rate and size estimates were obtained using 10-fold cross-validation on the part of the data set (2/3) that was set aside for evaluating the classifier. The results reported correspond to averages of the 10-fold cross-validation estimates obtained from the three choices of the AVT-construction and AVT-evaluation. This process ensures that there is no information leakage between the data used for AVT construction, and the data used for classifier construction and evaluation.

10-fold cross-validation experiments were performed to evaluate human expert-supplied AVT on the AVT evaluation data sets used in the experiments described above for the AGARICUS-LEPIOTA data set and the NURSERY data set.

We also evaluated the robustness of the AVT generated by the AVT-Learner by using them to construct classifiers from data sets with varying percentages of missing attribute values. The data sets with different percentages of missing values were generated by uniformly sampling from instances and attributes to introduce the desired percentage of missing values.

2.5.2 Results

AVT generated by AVT-Learner are competitive with human-generated AVT when used by AVT-NBL.

The results of our experiments shown in Figure 2.8 indicate that AVT-Learner is effective in constructing AVTs that are competitive with human expert-supplied AVTs for use in classification tasks with respect to the error rates and the size of the resulting classifiers.

AVT-Learner can generate useful AVT when no human-generated AVT are avail-

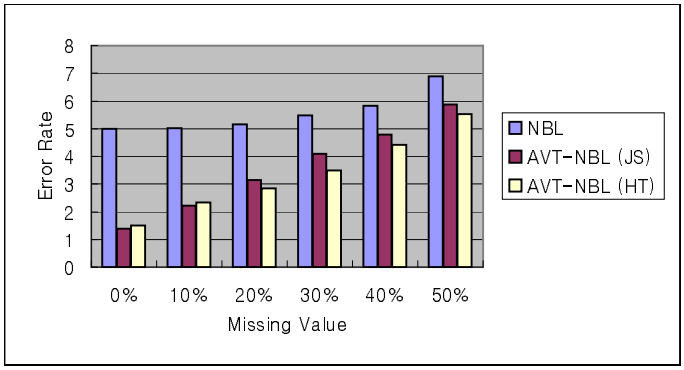


Figure 2.8 The estimated error rates of classifiers generated by NBL and AVT-NBL on AGARICUS-LEPIOTA data with different percentages of missing values. HT stands for human-supplied AVT. JS denotes AVT constructed by AVT-Learner using Jensen-Shannon divergence.

able.

For most of the data sets, there are no human-supplied AVT's available. Figure 2.9 shows the error rate estimates for Naive Bayes classifiers generated by AVT-NBL using AVT generated by the AVT-Learner and the classifiers generated by NBL applied to the DERMATOLOGY data set. The results shown suggest that AVT-Learner, using Jensen-Shannon divergence, is able to generate AVTs that when used by AVT-NBL, result in classifiers that are more accurate than those generated by NBL.

Additional experiments with other data sets produced similar results. Table 2.1 shows the classifier's accuracy on original UCI data sets for NBL and AVT-NBL that uses AVTs generated by AVT-Learner. 10-fold cross-validation is used for evaluation, and Jensen-Shannon divergence is used for AVT generation. The user-specified number for discretization is 10.

Thus, AVT-Learner is able to generate AVTs that are useful for constructing compact and accurate classifiers from data.

AVT generated by AVT-Learner, when used by AVT-NBL, yield substantially more compact Naive Bayes Classifiers than those produced by NBL

Naive Bayes classifiers constructed by AVT-NBL generally have smaller number of param-

Table 2.1 Accuracy of classifiers generated by Naive Bayes Learner (NBL) and AVT Guided Naive Bayes Learner (AVT-NBL) on UCI data sets, calculated by 10-fold cross validation with 95% confidence interval.

Data	NBL	AVT-NBL
Anneal	86.30±2.25	**99.00±0.65**
Audiology	73.45±5.76	**76.99±5.49**
Autos	56.10±6.79	**86.83±4.63**
Balance-scale	90.40±2.31	**91.36±2.20**
Breast-cancer	71.68±5.22	**72.38±5.18**
Breast-w	95.99±1.45	**97.28±1.21**
Car	85.53±1.66	**86.17±1.63**
Colic	77.99±4.23	**83.42±3.80**
Credit-a	77.68±3.11	**86.52±2.55**
Credit-g	75.40±2.67	75.40±2.67
Dermatology	97.81±1.50	**98.09±1.40**
Diabetes	76.30±3.01	**77.99±2.93**
Glass	48.60±6.70	**80.84±5.27**
Heart-c	83.50±4.18	**87.13±3.77**
Heart-h	83.67±4.22	**86.39±3.92**
Heart-statlog	83.70±4.41	**86.67±4.05**
Hepatitis	84.52±5.70	**92.90±4.04**
Hypothyroid	95.28±0.68	**95.78±0.64**
Ionosphere	82.62±3.96	**94.59±2.37**
Iris	**96.00±3.14**	94.67±3.60
Kr-vs-kp	87.89±1.13	**87.92±1.13**
Labor	89.47±7.97	89.47±7.97
Letter	64.12±0.66	**70.54±0.63**
Lymph	83.11±6.04	**84.46±5.84**
Mushroom	95.83±0.43	**99.59±0.14**
Nursery	90.32±0.51	90.32±0.51
Primary-tumor	**50.15±5.32**	47.79±5.32
Segment	80.22±1.62	**90.00±1.22**
Sick	92.68±0.83	**97.83±0.47**
Sonar	67.79±6.35	**99.52±0.94**
Soybean	92.97±1.92	**94.58±1.70**
Splice	95.36±0.73	**95.77±0.70**
Vehicle	44.90±3.35	**67.85±3.15**
Vote	90.11±2.80	90.11±2.80
Vowel	**63.74±2.99**	42.42±3.08
Waveform-5000	**80.00±1.11**	65.08±1.32
Zoo	93.07±4.95	**96.04±3.80**

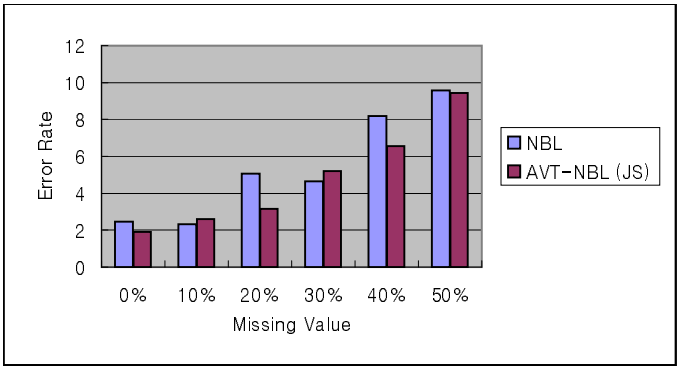


Figure 2.9 The error rate estimates of the Standard Naive Bayes Learner (NBL) compared with that of AVT-NBL on DERMATOLOGY data. JS denotes AVT constructed by AVT-Learner using Jensen-Shannon divergence.

eters than those from NBL (See Figures 2.10 for representative results). Table 2.2 shows the classifier size measured by the number of parameters on selected UCI data sets for NBL and AVT-NBL that uses AVTs generated by AVT-Learner.

These results suggest that AVT-Learner is able to group attribute values into AVT in such a way that the resulting AVT, when used by AVT-NBL, result in compact yet accurate classifiers.

2.6 Experiments for Word Taxonomy

The results of experiments described in this section provide evidence that WTNBL-MN coupled with WTL usually generate more concise and often more accurate classifiers than those of the Naive Bayes classifiers for the multinomial model. We conducted experiments with two sequence classification tasks; text (word sequence) classification and proteins (amino acid sequence) classification. In each case, a word taxonomy is generated using WTL and a classifier is constructed using WTNBL-MN on the resulting WT and sequence data.

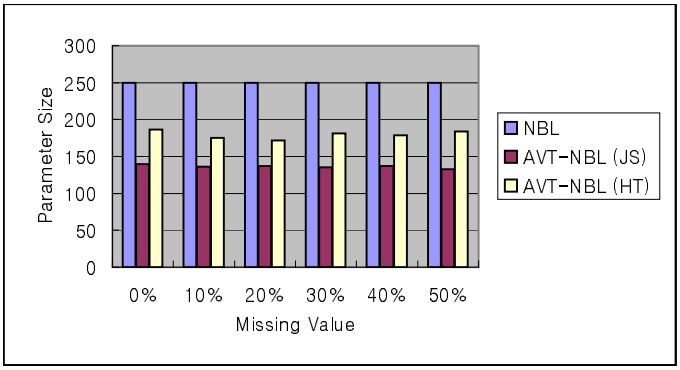


Figure 2.10 The size (as measured by the number of parameters) of classifiers from the standard Naive Bayes learner (NBL) compared with those from AVT-NBL on AGARICUS-LEPIOTA data. HT stands for human-supplied AVT. JS denotes AVT constructed by AVT-Learner using Jensen-Shannon divergence.

Table 2.2 The number of parameters of classifiers from NBL and AVT-NBL on selected UCI data sets

Data	NBL	AVT-NBL
Audiology	3720	**3600**
Breast-cancer	104	**62**
Car	88	**80**
Dermatology	906	**540**
Kr-vs-kp	150	**146**
Mushroom	252	**124**
Nursery	140	**125**
Primary-tumor	836	**814**
Soybean	1919	**1653**
Splice	864	**723**
Vote	66	66
Zoo	259	**238**

2.6.1 Text Classification

Reuters 21587 distribution 1.0 data set[1] consists of 12902 newswire articles in 135 overlapping topic categories.

We build binary classifiers for top ten most populous categories on text classification (Dumais et al., 1998; Joachims, 1998; McCallum and Nigam, 1998; Sandler, 2005; Keerthi, 2005; Joachims, 2005; Gabrilovich and Markovitch, 2005; Carvalho and Cohen, 2005; Rooney et al., 2006; Zhang and Lee, 2006). In our experiment, stop words were not eliminated, and title words were not distinguished with body words. We selected top 300 features based on mutual information with class labels. The mutual information $MI(x, c)$ between a feature x and a category c is defined as:

$$MI(x,c) = \sum^{x} \left\{ \sum^{c} \left\{ P(x,c) log \frac{P(x,c)}{P(x)P(c)} \right\} \right\}$$

We followed the ModApte split (Apté et al., 1994) in which 9603 stories are used for building classifiers and 3299 stories to test the accuracy of the resulting model. We report the break even points, the average of precision and recall when the difference between the two is minimum. Precision and recall of text categorization are defined as:

$$\text{Precision} = \frac{|\text{detected documents in the category (true positives)}|}{|\text{documents in the category (true positives + false negatives)}|}$$

$$\text{Recall} = \frac{|\text{detected documents in the category (true positives)}|}{|\text{detected documents (true positives + false positives)}|}$$

Table 2.3 shows the break even point of precision and recall as well as the size of the classifier (from the equation 2.5) for the ten most frequent categories. WTNBL-MN usually shows similar performance in terms of break even performance, while the classifiers generated by WTNBL-MN have smaller size than those generated by the Naive Bayes Learner (NBL).

Figure 2.11 shows Precision-Recall curves (Fawcett, 2003, 2006) of for the "grain" category. It can be seen that WTNBL-MN generates a Naive Bayes classifier that is more compact than, but has performance comparable to that of the classifier generated from Naive Bayes learner.

[1]This collection is publicly available at
http://www.daviddlewis.com/resources/testcollections/reuters21578/.

Table 2.3 Break even points (BEP) of Classifiers from Naive Bayes Learner with Multinomial Model (NBL-MN) and WTNBL-MN on 10 Largest Categories of Reuters 21586 Data

Data	NBL-MN		WTNBL-MN (Abstraction)		WTNBL-MN (Specialization)		# of documents	
	BEP	size	BEP	size	BEP	size	train	test
earn	**94.94**	602	94.57	376	94.57	**348**	2877	1087
acq	89.43	602	89.43	**446**	89.43	472	1650	719
money-fx	64.80	602	64.80	390	**65.36**	**346**	538	179
grain	74.50	602	74.50	418	**77.85**	**198**	433	149
crude	79.89	602	**80.42**	450	76.72	**182**	389	189
trade	**59.83**	602	**59.83**	406	47.01	**208**	369	118
interest	**61.07**	602	**61.07**	468	59.54	**366**	347	131
ship	82.02	602	82.02	**314**	82.02	348	197	89
wheat	**57.75**	602	**57.75**	402	53.52	**226**	212	71
corn	57.14	602	**58.83**	344	21.43	**106**	182	56
Average (top 5)	80.71	602	80.74	416	**80.79**	**309.20**		
Average (top 10)	72.14	602	**72.32**	401.40	66.75	**280**		

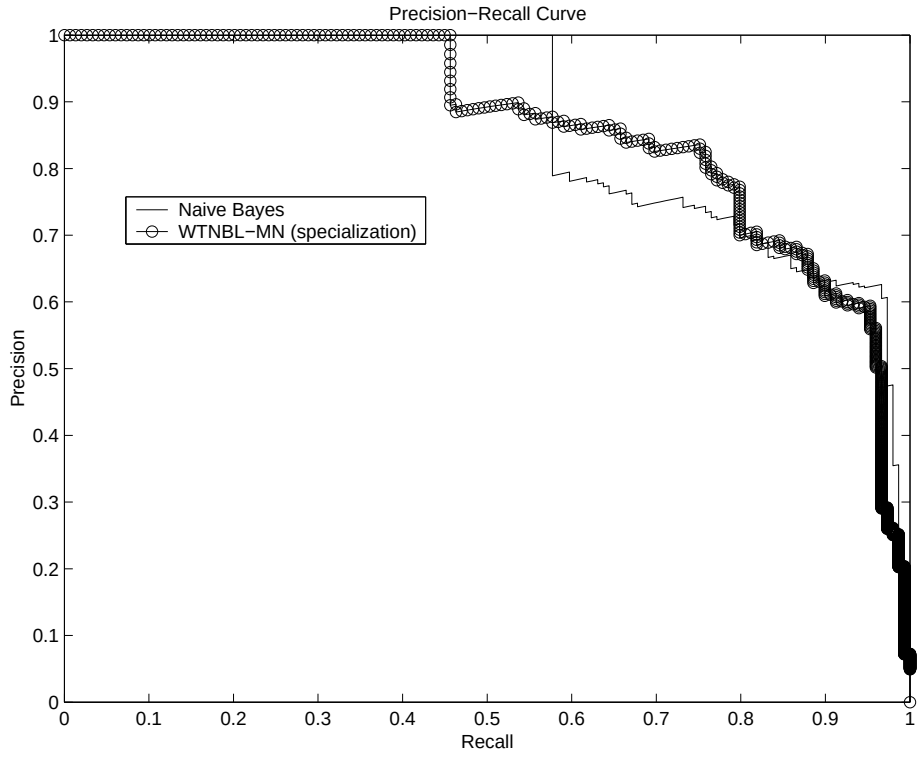


Figure 2.11 Precision-Recall Curves for the "Grain" Category

2.6.2 Protein Sequences

We applied the WTNBL-MN algorithm on two protein data sets with a view to identifying their localization (Reinhardt and Hubbard, 1998; Andorf et al., 2006).

The first data set is 997 prokaryotic protein sequences derived from SWISS-PROT data base (Bairoch and Apweiler, 2000). This data set includes proteins from three different subcellular locations: cytoplasmic (688 proteins), periplasmic (202 proteins), and extracellular (107 proteins).

The second data set is 2427 eukaryotic protein sequences derived from SWISS-PROT data base (Bairoch and Apweiler, 2000). This data set includes proteins from the following four different subcellular locations: nuclear (1097 proteins), cytoplasmic (684 proteins), mitochondrial (321 proteins), extracellular (325 proteins).

For these data sets[2], we conducted ten-fold cross validation. To measure the performance of the following performance measures (Yan et al., 2004) are applied and the results for the data set are reported:

$$\text{Correlation coefficient} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP+FN})(\text{TP+FP})(\text{TN+FP})(\text{TN+FN})}}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP+TN+FP+FN}}$$

$$\text{Sensitivity}^{+} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{Specificity}^{+} = \frac{\text{TP}}{\text{TP+FP}}$$

where, TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives, and FN is the number of false negatives.

Figure 2.12 shows amino acid taxonomy constructed for the Prokaryotic protein sequences. Table 2.4 shows the results in terms of the performance measures for the two protein sequences. For both data sets, the classifiers generated by WTNBL are more concise and show more accurate performance than the classifier generated by the Naive Bayes Learner (NBL) in terms of the measures reported.

Table 2.4 Localization prediction results of Naive Bayes Learner with Multinomial Model (NBL-MN) and WTNBL-MN on Prokaryotic and Eukaryotic protein sequences, calculated by 10-fold cross validation with 95% confidence interval.

(a) Prokaryotic protein sequences

NBL-MN	Correlation Coefficient	Accuracy	Specificity$^+$	Sensitivity$^+$	Size
Cytoplasmic	71.96±2.79	88.26±2.00	89.60±1.89	93.90±1.49	42
Extracellular	**70.57±2.83**	**93.58±1.52**	**65.93±2.94**	83.18±2.32	42
Peripalsmic	51.31±3.10	81.85±2.39	53.85±3.09	72.77±2.76	42
WTNBL-MN (Abstraction)	Correlation Coefficient	Accuracy	Specificity$^+$	Sensitivity$^+$	Size
Cytoplasmic	72.19±2.78	88.37±1.99	89.61±1.89	94.04±1.47	**16**
Extracellular	69.62±2.85	93.18±1.56	63.83±2.98	**84.11±2.27**	**16**
Peripalsmic	50.88±3.10	81.85±2.39	**53.90±3.09**	71.78±2.79	**40**
WTNBL-MN (Specialization)	Correlation Coefficient	Accuracy	Specificity$^+$	Sensitivity$^+$	Size
Cytoplasmic	**72.43±2.77**	**88.47±1.98**	**89.63±1.89**	**94.19±1.45**	20
Extracellular	69.31±2.86	93.18±1.56	64.03±2.98	83.18±2.32	20
Peripalsmic	**51.53±3.10**	81.85±2.39	53.82±3.09	**73.27±2.75**	**40**

(b) Eukaryotic protein sequences

NBL-MN	Correlation Coefficient	Accuracy	Specificity$^+$	Sensitivity$^+$	Size
Nuclear	**61.00±1.94**	**80.72±1.57**	**82.06±1.53**	73.38±1.76	46
Extracellular	36.83±1.92	83.11±1.49	40.23±1.95	**53.85±1.98**	46
Mitochondrial	25.13±1.73	71.69±1.79	25.85±1.74	**61.06±1.94**	46
Cytoplasmic	**44.05±1.98**	**71.41±1.80**	**49.55±1.99**	**81.29±1.55**	46
WTNBL-MN (Abstraction)	Correlation Coefficient	Accuracy	Specificity$^+$	Sensitivity$^+$	Size
Nuclear	58.14±1.96	79.32±1.61	80.51±1.58	71.56±1.79	**20**
Extracellular	34.40±1.89	83.15±1.49	39.60±1.95	49.23±1.99	**32**
Mitochondrial	25.15±1.73	**72.85±1.77**	**26.40±1.75**	58.88±1.96	**26**
Cytoplasmic	43.42±1.97	71.16±1.80	49.29±1.99	80.70±1.57	**26**
WTNBL-MN (Specialization)	Correlation Coefficient	Accuracy	Specificity$^+$	Sensitivity$^+$	Size
Nuclear	60.82±1.94	80.63±1.57	81.70±1.54	**73.66±1.75**	24
Extracellular	**38.21±1.93**	**84.01±1.46**	**42.30±1.97**	53.23±1.99	36
Mitochondrial	**25.48±1.73**	72.35±1.78	26.29±1.75	60.44±1.95	34
Cytoplasmic	43.46±1.97	71.24±1.80	49.37±1.99	80.56±1.57	32

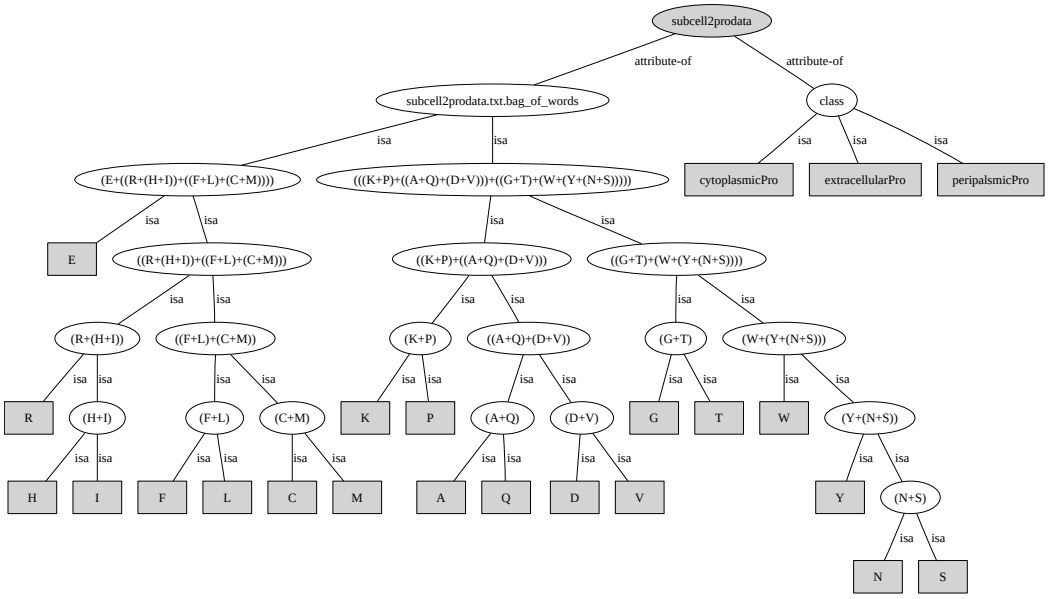


Figure 2.12 Taxonomy from Prokaryotic Protein Localization Sequences constructed by WTL

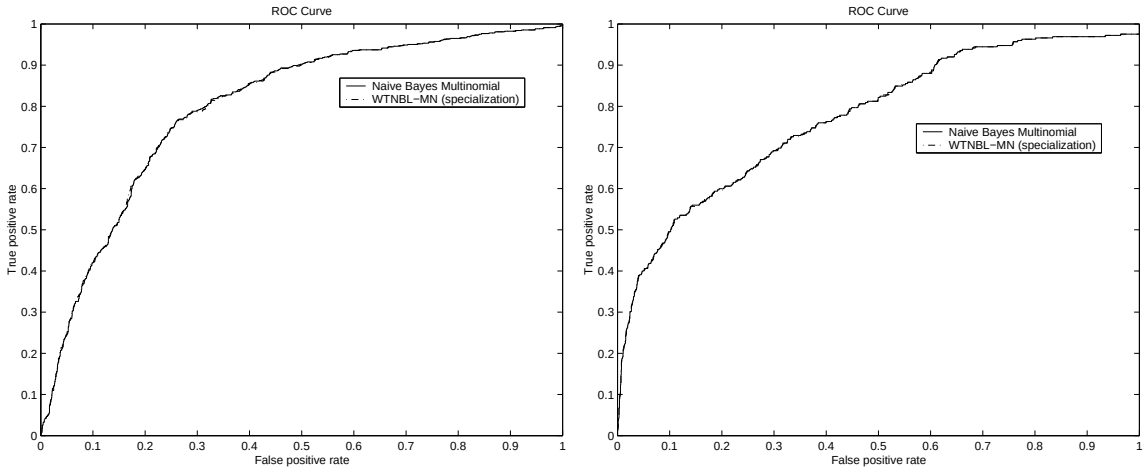
Figure 2.13 shows the ROC curves of the classifiers from Naive Bayes learner and WTNBL-MN on the classification tasks of Cytoplasmic label and Extracellular label respectively. The AUCs of Naive Bayes learner and WTNBL-MN for each class are not very different and the ROC curves of both classifiers nearly overlap each other. This means that either classifier does not show better performance than the other. However, we have seen in the table 2.12 that WTNBL-MN yields more compact classifiers than Naive Bayes learner.

2.7 Summary and Discussion

2.7.1 Summary

In many applications of data mining, there is a strong preference for classifiers that are both accurate and compact (Kohavi and Provost, 2001; Pazzani et al., 1997). Previous work has shown that attribute value taxonomies can be exploited to generate such classifiers from data (Zhang and Honavar, 2003, 2004). However, human-generated taxonomies are unavail-

[2]These datasets are available to download at http://www.doe-mbi.ucla.edu/~astrid/astrid.html.



(a) Cytoplasmic: NB AUC=0.7985 and WTNBL AUC = 0.7983

(b) Extracellular: NB AUC=0.7752 and WTNBL AUC = 0.7753

Figure 2.13 ROC curves of Naive Bayes (NB) and WTNBL-MN on the classification of Cytoplasmic label and Extracellular label

able in many application domains. Manual construction of taxonomies requires a great deal of domain expertise, and in case of large data sets with many attributes and many values for each attribute, manual generation of taxonomies is extremely tedious and hence not feasible in practice. Against this background, we have described AVT-Learner, an algorithm for automated construction of attribute value taxonomies (AVT) from data, and Word Taxonomy Learner (WTL) for automated construction of word taxonomy from text and sequence data. AVT-Learner and WTL recursively groups values (of attributes) based on a suitable measure of divergence between the class distributions associated with the values to construct taxonomies. AVT-Learner is able to generate hierarchical taxonomies of nominal, ordinal, and continuous valued attributes.

The experiments reported in this chapter show that:

- AVT-Learner and WTL is effective in generating taxonomies that when used by AVT-NBL and WTNBL-MN, a principled extension of the standard algorithm for learning Naive Bayes classifiers, result in classifiers that are substantially more compact (and often more accurate) than those obtained by the standard Naive Bayes Learner (that does not use taxonomies).

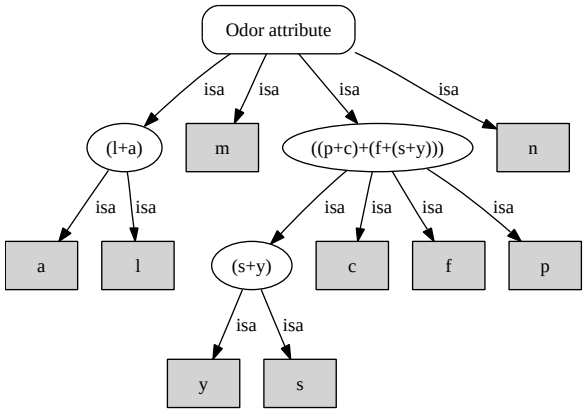


Figure 2.14 AVT of 'odor' attribute of UCI AGARICUS-LEPIOTA mushroom data set generated by AVT-Learner using Jensen-Shannon divergence (with quaternary clustering)

- The taxonomies generated by AVT-Learner and WTL are competitive with human supplied taxonomies (in the case of benchmark data sets where human-generated taxonomies were available) in terms of both the error rate and size of the resulting classifiers.

2.7.2 Discussion

2.7.2.1 Binary vs. K-ary Clustering

The AVTs generated by AVT-Learner are binary trees. Hence, one might wonder if k-ary AVTs yield better results when used with AVT-NBL. Figure 2.14 shows an AVT of 'odor' attribute generated by AVT-Learner (with quaternary clustering). Table 2.5 shows the accuracy of AVT-NBL when k-ary clustering is used by AVT-Learner. It can be seen that AVT-NBL generally works best when binary AVTs are used. It is because reducing internal nodes in AVT-Learner will eventually reduce the search space for possible cuts in AVT-NBL, which leads to generating a less compact classifier.

2.7.2.2 Ordered vs. Orderless

At each step of merging, AVT-Learner considers all possible pairs of values and does not consider the order between the values. However, there are attributes whose values are ordered,

Table 2.5 Accuracy of classifiers generated by AVT Guided Naive Bayes Learner (AVT-NBL) coupled with k-ary AVT-Learner when k is 2,3 and 4 on selected UCI data sets, calculated by 10-fold cross validation with 95% confidence interval..

Data	2-ary	3-ary	4-ary
Nursery	90.32±0.51	90.32±0.51	90.32±0.51
Audiology	**76.99±5.49**	76.55±5.52	**76.99±5.49**
Car	86.17±1.63	86.17±1.63	86.17±1.63
Dermatology	**98.09±1.40**	97.54±1.59	97.54±1.59
Mushroom	99.59±0.14	99.73±0.11	**99.75±0.11**
Soybean	**94.58±1.70**	94.44±1.72	94.44±1.72

such as discretized attributes or ordinal attributes. For such attributes, one might wonder if merging most similar "adjacent" values of an ordered attribute may yields better results. Table 2.6 shows the accuracies of AVT-NBL when the corresponding AVT-Learner maintains the order of values when it builds the AVT, and the accuracies of AVT-NBL when the AVT-Learner does not maintain the order of values. It can be seen that AVT-NBL generally works better when AVT is constructed without concerning the order of attribute values. It is because AVT-Learner needs to compare more possible pairs of values when the algorithm does not consider the order of values.

2.7.3 Related Work

Gibson and Kleinberg (Gibson, 1988) introduced STIRR, an iterative algorithm based on non-linear dynamic systems for clustering categorical attributes. Ganti et. al. (Ganti et al., 1999) designed CACTUS, an algorithm that uses intra-attribute summaries to cluster attribute values. Zaki et al. (Zaki et al., 2005) presented CLICKS algorithm that finds clusters in categorical datasets based on a search for k-partite maximal cliques. However, all of them did not make taxonomies and use the generated for improving classification tasks.

Pereira et. al. (Pereira et al., 1993) described distributional clustering for grouping words based on class distributions associated with the words in text classification. Yamazaki et al., (Yamazaki et al., 1995) described an algorithm for extracting hierarchical groupings from rules learned by FOCL (an inductive learning algorithm) (Pazzani and Kibler, 1992) and

Table 2.6 Accuracy of classifiers generated by AVT Guided Naive Bayes Learner (AVT-NBL) from ordered AVT and orderless AVT on UCI data sets, calculated by 10-fold cross validation with 95% confidence interval.

Data	AVT-NBL (ordered AVT)	AVT-NBL (orderless AVT)
Anneal	98.55±0.78	**99.00±0.65**
Audiology	76.99±5.49	76.99±5.49
Autos	82.93±5.15	**86.83±4.63**
Balance-scale	91.36±2.20	91.36±2.20
Breast-cancer	72.38±5.18	72.38±5.18
Breast-w	97.28±1.21	97.28±1.21
Car	86.17±1.63	86.17±1.63
Colic	82.61±3.87	**83.42±3.80**
Credit-a	85.94±2.59	**86.52±2.55**
Credit-g	75.40±2.67	75.40±2.67
Dermatology	98.09±1.40	98.09±1.40
Diabetes	76.56±3.00	**77.99±2.93**
Glass	78.04±5.55	**80.84±5.27**
Heart-c	83.50±4.18	**87.13±3.77**
Heart-h	85.37±4.04	**86.39±3.92**
Heart-statlog	85.19±4.24	**86.67±4.05**
Hepatitis	89.03±4.92	**92.90±4.04**
Hypothyroid	**95.84±0.64**	95.78±0.64
Ionosphere	94.59±2.37	94.59±2.37
Iris	94.67±3.60	94.67±3.60
Kr-vs-kp	87.92±1.13	87.92±1.13
Labor	87.72±8.52	**89.47±7.97**
Letter	**72.32±0.62**	70.54±0.63
Lymph	84.46±5.84	84.46±5.84
Mushroom	99.59±0.14	99.59±0.14
Nursery	90.32±0.51	90.32±0.51
Primary-tumor	47.79±5.32	47.79±5.32
Segment	87.66±1.34	**90.00±1.22**
Sick	97.80±0.47	**97.83±0.47**
Sonar	92.31±3.62	**99.52±0.94**
Soybean	94.58±1.70	94.58±1.70
Splice	95.77±0.70	95.77±0.70
Vehicle	61.82±3.27	**67.85±3.15**
Vote	90.11±2.80	90.11±2.80
Vowel	42.22±3.08	**42.42±3.08**
Waveform-5000	63.64±1.33	**65.08±1.32**
Zoo	96.04±3.80	96.04±3.80

reported improved performance on learning translation rules from examples in a natural language processing task. Slonim and Tishby (Slonim et al., 2006) described a technique (called the agglomerative information bottleneck method) which extended the distributional clustering approach described by Pereira et al. (Pereira et al., 1993), using Jensen-Shannon divergence for measuring distance between document class distributions associated with words and applied it to a text classification task. Baker and McCallum (Baker and McCallum, 1998) reported improved performance on text classification using a technique similar to distributional clustering and a distance measure, which upon closer examination, can be shown to be equivalent to Jensen-Shannon divergence (Slonim et al., 2005, 2006). Dimitropoulos et al. (Dimitropoulos et al., 2006) augmented Internet Autonomous System (AS) Taxonomy to the data set for ASes classification and successfully classified 95.3% of ASes with expected accuracy of 78.1%.

To the best of our knowledge, there has been little work on the evaluation of techniques for generating hierarchical groupings of attribute values (AVTs) on classification tasks using a broad range of benchmark data sets using algorithms such as AVT-DTL or AVT-NBL that are capable of exploiting AVTs in learning classifiers from data.

2.7.4 Future Work

Some directions for future work include:

- Extending AVT-Learner described in this research to learn AVTs that correspond to tangled hierarchies, which can be represented by directed acyclic graphs (DAG) instead of trees, or complicated graph structure
- Learning AVT from data for a broad range of real world applications such as census data analysis, learning classifiers from relational data (Atramentov et al., 2003), and protein function classification (Wang and Stolfo, 2003), identification of protein-protein interfaces (Terribilini et al., 2006; Yan et al., 2003)
- Developing algorithms for learning hierarchical ontologies based on part-whole and other relations as opposed to ISA relations captured by an AVT

- Developing algorithms for learning hierarchical groupings of values associated with more than one attribute

CHAPTER 3. HOST-BASED INTRUSION DETECTION USING A BAG OF SYSTEM CALLS

This chapter is an extended version of the paper published in IEEE International Conference on Intelligence and Security Informatics in 2005 (Kang et al., 2005c).

3.1 Abstract

In this study, we propose a *bag of system calls* representation for intrusion detection in system call sequences and describe misuse and anomaly detection results with standard machine learning techniques on University of New Mexico (UNM) and MIT Lincoln Lab (MIT LL) system call sequences with the proposed representation. With the feature representation as input, we compare the performance of several machine learning techniques for misuse detection and show experimental results on anomaly detection. The results show that standard machine learning and clustering techniques on simple bag of system calls representation of system call sequences in the operating system's kernel is effective and often performs better than those approaches that use foreign contiguous sequences in detecting intrusive behaviors of compromised processes.

3.2 Introduction

Detection of attempts to compromise the integrity, confidentiality, or availability of computing and communication networks is an extremely challenging problem (Denning, 1987). Most current approaches to the design of intrusion detection systems (IDS) are based on the premise that the actions used in an attempted intrusion can be differentiated from the actions executed by users or processes during the normal operation of the computing and communica-

tion networks (Axelsson, 2000; Murali and Rao, 2005). An effective IDS logs actions executed by users or processes for investigation, alerts the system administrator when the monitored activities are indicative of attempted intrusion, and, if appropriate, takes corrective measures e.g., expelling the intruder.

Intrusion detection and prevention generally refers to a broad range of strategies for defending against malicious attacks. Intrusion detection can be categorized into *misuse* detection and *anomaly* detection. Misuse typically is a known attack, e.g., a hacker attempting to break into an email server in a way that IDS has already trained. A misuse detection system tries to model normal and abnormal behavior from known attacks. It works by comparing network traffic, system call sequences, or other features of known attack patterns. An anomaly is something out of the ordinary, e.g., abnormal network traffic which is actually caused by unknown attacks. An anomaly detection system models *normal* behavior and identifies a behavior as abnormal (or anomalous) if it is sufficiently different from known normal behaviors.

IDS can be classified into those that focus on modeling the behavior of users and those that focus on modeling the behavior of processes (Ghosh and Schwartzbard, 1999). System call data are one of the most common types of data used to model the behavior of processes. Such data can be collected by logging the system calls using operating system utilities e.g. Linux strace or Solaris Basic Security Module (BSM).

There has been a great deal of research on how to design and implement intrusion detection systems. For example, Mukherjee et al (Mukherjee et al., 1994) used a combination of host monitors and network monitors with a centralized director for suspicious system activities in the distributed intrusion detection system (DIDS) project. Because it is difficult to manually specify activities that signal intrusive behavior, there has been much work on adaptive or machine learning or data mining approaches for intrusion detection. Forrest et al (Forrest et al., 1996) worked on the Computer Immunology project and explored approaches inspired by the activities of the immune systems of animals for detecting and defending against intrusions. Subsequently, several groups have explored data mining approaches for intrusion detection (Lee et al., 1999; Helmer et al., 2001; Eskin et al., 2002; Campos and Milenova, 2005).

In most IDS that model the behavior of processes, intrusions are detected by observing fixed-length, contiguous subsequences of system calls. For example, in anomaly detection, subsequences of input traces are matched against normal sequences in database so that foreign sequences (Forrest et al., 1996; Hofmeyr et al., 1998) are detected. One potential drawback of this approach is that the size of the database that contains fixed-length contiguous subsequences increases exponentially with the length of the subsequences. For example, if the number of system calls is 200 and the length of the subsequences is 6, the size of the database is theoretically $200^6 = 64 \times 10^{12}$. In practice, only normal subsequences are stored, so actual database size is smaller, but still considerably bigger than the hypothesis size generated by our approach.

In this study, we explore an alternative representation of system call traces for intrusion detection. Specifically, we use a *bag of system calls* representation of system call sequences. In other words, we consider intrusion detection of system call sequence as a classification problem on a bag of system calls obtained from the system call sequences. With those problem setting, we constructed and evaluated decision tree (Quinlan, 1993), Naive Bayes (McCallum and Nigam, 1998), decision list (Rivest, 1987; Cohen, 1995), Support Vector Machines (SVM) (Cortes and Vapnik, 1995; Platt, 1999), and Logistic Regression (with a ridge estimator) (Cessie and Houwelingen, 1992) classifiers using bag of system calls representation of system calls for misuse detection. We also explored an approach to anomaly detection using a one class Naive Bayes classifier as well as K-means clustering (Bishop, 1996) using the same representation of system call sequences. Bag of words model is already popular in text classification and categorization area (Mitchell, 1997), and our motivation is to investigate the usefulness of the model in intrusion detection tasks.

The results show that the proposed approach for misuse detection yields comparable or sometimes better performance than the methods previously reported in the literature in terms of detection rate and false positive over widely used benchmark data sets such as University of New Mexico (UNM) and MIT Lincoln Lab (MIT LL) system call sequences.

The rest of the chapter is organized as follows. Section 3.3 describes two different repre-

sentations of system call sequences. Section 3.4 describes the benchmark data sets used in our study. Section 3.5 describes the experimental setup and results. Section 3.6 concludes with a summary and discussion.

3.3 Alternative Representations of System Call Sequences

We describe two feature representations of system call sequences that intrusion detection algorithms deal with. The first one is a contiguous subsequence with fixed length k from original input traces, and the second is bag of system calls, which is our approach.

One of the main questions in sequence-based intrusion detection is how to define "intrusion" in an input sequence. Most intrusion detection algorithms such as *STIDE* (Warrender et al., 1999) regard that intrusion is related with fixed-length subsequence that only happens in intrusive traces.

In our approach, we convert the input sequence to bag of system calls. Thus, the ordering information between system calls is lost and only the frequency of each system call is preserved for each input sequence. Intrusion is represented according to the machine learning algorithm applied.

Formally, the intrusion detection problem on system call or command sequences can be defined as follows:

Let $\Sigma = \{s_1, s_2, s_3, \ldots, s_m\}$ be a set of system calls where $m = |\Sigma|$ is the number of system calls. Data set D can be defined as a set of labeled sequences $\{< Z_i, c_i > | Z_i \in \Sigma^*, c_i \in \{0, 1\}\}$ where Z_i is an input sequence and c_i is a corresponding class label denoting 0 for "normal" label and 1 for "intrusion" label. Given the data set D, the goal of the learning algorithm is to find a classifier $h : \Sigma^* \rightarrow \{0, 1\}$ that maximizes given criteria. Such criteria are accuracy, detection rate and false positive rate.

Since it is difficult to deal with sequences directly, each sequence $Z \in \Sigma^*$ is mapped into a finite dimensional feature vector by a feature representation $\Phi : \Sigma^* \rightarrow \mathbf{X}$. Thus, the classifier is defined as $h : \mathbf{X} \rightarrow \{0, 1\}$ for data set $\{< X_j, c_j > | X \in \mathbf{X}, c_j \in \{0, 1\}\}$.

3.3.1 Contiguous Foreign Subsequences

In this approach, a feature is defined as $X_j = x_1x_2x_3\ldots x_l$, a substring of Z_i, where $x_k \in \Sigma$ and l is a constant. The number of possible features is $\left|\Sigma^l\right| \geq j$ and each feature X_j is assigned a class label c_i according to the original sequence Z_i.

STIDE uses sliding windows with length l over an original input trace to generate fixed-length substrings as features and constructs a database of the features in the training stage, and decides a test sequence is anomalous if the number of mismatches in the user-specified locality frame (locality frame count), which is composed of adjacent features in the frame, is more than the user-specified threshold. Empirically, it is widely accepted that, for effective intrusion detection, the minimal value of k is six (Tan and Maxion, 2002).

3.3.2 Bag of System Calls

"Bag of system calls" representation is an integer-frequency based method. In our approach, a feature is defined as an ordered list $X_i = \langle c_1, c_2, c_3, \ldots, c_m \rangle$ where $m = |\Sigma|$ and c_j is the number of occurrence of system call s_j in the input sequence Z_i.

Thus, the original trace is converted to a bag of system calls, and the ordering information of adjacent system calls in the input sequence is lost and only the frequency of each system call in the bag is preserved. Intrusion in this feature representation is defined according to the machine learning algorithm applied.

One of the main issues in this study is whether the bag of system calls representation, which is already popular in text classification and categorization, can effectively represent intrusion. We will show experimental results over University of New Mexico (UNM) and Massachusetts Institute of Technology Lincoln Lab (MIT LL) data in later sections for this issue. It will be shown that frequency information is effective enough to discriminate between normal sequences and abnormal sequences. As an example for this, figure 3.1 shows a histogram of the average frequency of selected system calls in normal sequences and abnormal sequences in UNM denial of service (DoS) trace data set (also known as stide data set). We found a similar phenomenon for other data sets including MIT LL data sets.

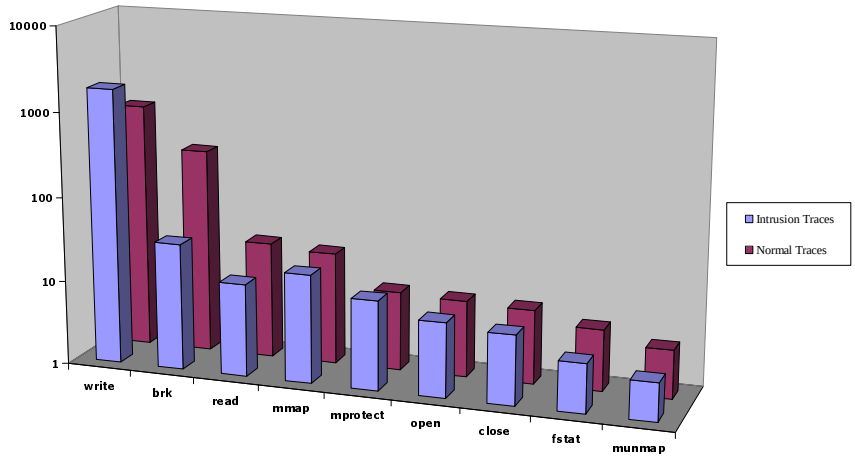


Figure 3.1 Average frequency of selected system calls in normal traces and intrusion traces in UNM denial of service trace data set

3.4 Data Sets

For experiments, we choose publicly available system call sequences from UNM and MIT LL data.

3.4.1 UNM System System Call Sequences

The University of New Mexico (UNM) provides a number of system call data sets. The data sets we tested are "live lpr", "live lpr MIT", "synthetic sendmail", "synthetic sendmail CERT", and "denial of service"(DoS).

In UNM system call traces, each trace is an output of one program. Sometimes, one trace has multiple processes. In such cases, we have made one sequence per process in the original trace. Thus, multiple sequences of system calls are made from one trace if the input trace has multiple processes in it. However, most traces have only one process and usually one sequence is created for each trace. Table 3.1 shows the number of original traces and the number of sequences for each program.

There are three different mapping files in UNM call traces. One is Sun (synthetic sendmail, synthetic sendmail CERT, synthetic lpr, live lpr and live lpr.MIT) , another is Linux (live named, login, ps, inet and DoS), and the third is new Linux (synthetic ftp and xlock). There

Table 3.1 The number of original traces and generated sequences in UNM data sets

Program	# of original traces	# of sequences
live lpr (normal)	1232	1232
live lpr (exploit)	1001	1001
live lpr MIT (normal)	2704	2704
live lpr MIT (exploit)	1001	1001
synthetic sendmail (normal)	7	346
synthetic sendmail (exploit)	10	25
synthetic sendmail CERT (normal)	2	294
synthetic sendmail CERT (exploit)	6	34
denial of service (normal)	13726	13726
denial of service (exploit)	1	105

are old and new Sun mapping files but only one system call is added to the new mapping file so both can be easily converted. The Sun mapping file has a few duplicate system calls (e.g. 'fstat', 'stat', etc.), but we changed them so that each system call is unique.

3.4.2 MIT Lincoln Lab System Call Sequences

We used data sets provided by the MIT Lincoln Lab (Lippmann et al., 1999).

The fourth week (starting at 6/22/98) training data set of year 1998 is used for the experiments in this study. This training data is comprised of a detailed set of data files representing the state of a particular system over eight-hour daytime periods over the course of the week beginning on 6/22/98. Of interest here is the omnibus data file containing all system calls made during the collection period and the network traffic analysis file (distilled from raw network data) that identifies inbound network connection attempts.

We explain the issues with cross-indexing the data files. MIT Lincoln Labs datasets included an omnibus file containing all system call traces along with a separate, network traffic analysis data file indicating inbound network connections to the system. Attack attempts are logged with the network data, so labeling of the training data requires cross-indexing this file with the system call trace file. The system call trace file identifies the source of each call using the process ID. Therefore, cross-indexing requires tracking the argument to the 'exec' system

call identifying the binary to be executed. Additionally, the timestamps from the network traffic analyzer do not exactly correspond to the execution timestamps from the operating system kernel. A tolerance of one second was arbitrarily chosen and seems to permit the matching of a large majority of connection attempts with their corresponding server processes run on the target system.

All processes detected that do not correspond to some network connection attempt identified in the trace are removed from consideration (since they cannot be classified), as are all calls attributed to a process ID for which an 'exec' system call is not found. The resulting data are available at

`http://www.cs.iastate.edu/~dkkang/IDS_Bag/`.

3.5 Experiments and Results

We use different approaches for three different types of intrusion detection experiments. The approaches will be explained at each respective section.

The data sets we have tested are "live lpr", "live lpr MIT", "synthetic sendmail", "synthetic sendmail CERT", and "denial of service attack" of UNM, and the fourth week training data set of year 1998 in MIT LL.

For the evaluation of classifiers generated in the experiment, 10-fold cross validation is used, so no training information is reused in the test stage. In 'x'-fold cross-validation, the data set is divided into x subsets of approximately equal size. One of the subsets is picked for testing and the rest subsets are used for training. In other words, a classifier is generated from 'x-1' subsets and the classifier is tested over the rest subset. This routine is applied for each of x different subsets, and then accuracy, detection rate and false positive rate are averaged respectively over each of x different subsets tested. This is to ensure that no information used for classifier generation is reused as test data.

Accuracy, detection rate, and false positive rate are defined as follows:

$$\text{accuracy} = \frac{\#\text{ of true positives} + \#\text{ of true negatives}}{\#\text{ of input sequences}}$$

$$\text{detection rate} = \frac{\#\text{ of true positives}}{\#\text{ of true positives} + \#\text{ of false negatives}}$$

$$\text{false positive rate} = \frac{\#\text{ of false positives}}{\#\text{ of true negatives} + \#\text{ of false positives}}$$

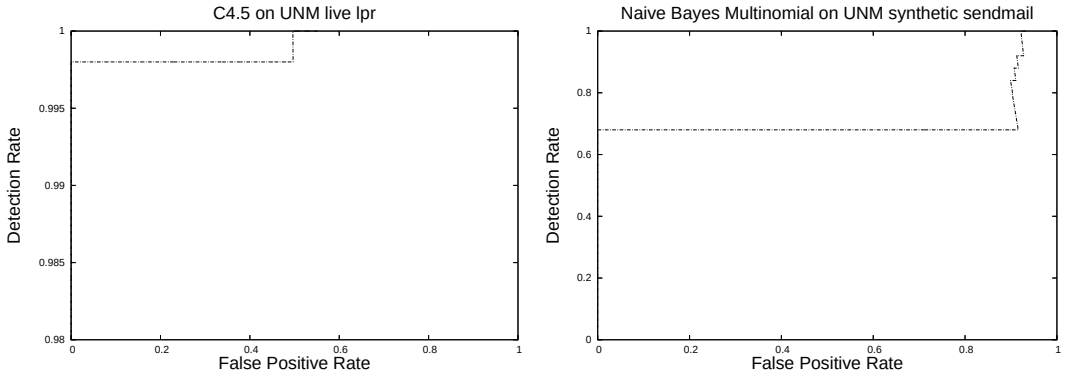
3.5.1 Experimental Results on Misuse Detection

For misuse detection, we use several machine learning techniques. We tested Naive Bayes Multinomial (McCallum and Nigam, 1998), C4.5 (Quinlan, 1993), RIPPER (Cohen, 1995), SVM (Cortes and Vapnik, 1995; Platt, 1999) (with two class labels), and Logistic Regression. For SVM, Sequential Minimal Optimization (SMO) (Platt, 1999) with a linear kernel was used for training, and for logistic regression, a multinomial logistic regression model with a ridge estimator (Cessie and Houwelingen, 1992) was used. Table 3.2 shows the accuracy, detection rate, and false positive rate of the data sets with 95% confidence intervals by doing t-test (Duda et al., 2000). The detection rate is a fraction of the intrusions identified and the false positive rate is a fraction of normal data mis-identified as intrusion.

The results in table 3.2 show that standard machine learning techniques are effective in misuse detection with simple bag of system calls representation. For example, with SMO using a linear kernel, an SVM can perfectly detect both normal and intrusion sequences in the "UNM live lpr" data set.

In the MIT LL results in table 3.2, it is interesting that all machine learning algorithms got the same results on each day tested. We did not try to detect the type of intrusion and assign a corresponding score for the intrusion as was intended in the original evaluation in 1998. Instead we just tried to detect intrusion. Perhaps, the reason that all algorithms have the same results for each day is that the normal sequences and intrusion sequences in the data set are already highly different. Wednesday data set was not tested because no intrusions were in the network traffic analysis file.

One problem is that the machine learning algorithms will not work well when data is not quite balanced, which is common in intrusion detection practice. For example, "UNM synthetic sendmail" and "UNM synthetic sendmail CERT" data sets in the table are such data sets, and that's why their detection rate or false positive rate is not quite optimal. There



(a) C4.5 decision tree induction on UNM live lpr (b) Naive Bayes Multinomial on UNM synthetic sendmail

Figure 3.2 ROC Curve of "UNM live lpr" and "UNM synthetic sendmail" data sets in misuse detection

are several ways to deal with this imbalanced data problem. Here, we use a cost matrix which assigns different weights for each misclassification. Figure 3.2 shows the Receiver Operating Characteristic (ROC) Curve of "UNM live lpr" and "UNM synthetic sendmail" data sets using C4.5 and Naive Bayes Multinomial algorithms respectively.

From figure 3.2(a), we can see that the classifier generated from "UNM live lpr" data set is very effective because it has sufficient number of intrusion data for training. The "UNM live lpr" data set has 183 attributes, 1232 normal sequences, and 1001 intrusion sequences. From figure 3.2(b), standard machine learning techniques have limitations in this case, because the data sets themselves are small. The "UNM synthetic sendmail" data set has 182 attributes, 346 normal sequences, and only 25 intrusion sequences. Since we tested the algorithm with 10 fold cross-validation, in some folds, the algorithm did not have enough intrusion samples.

3.5.2 Detecting intrusion from the generated rules

One of the problems in our bag of system calls representation is that, with some machine learning algorithms, the classification cannot be done until the end of the process (Warrender et al., 1999). However, with the machine learning algorithms that generate comprehensive hypotheses, we can use very simple rules to detect a process that has exhibited intrusive behavior before it is terminated.

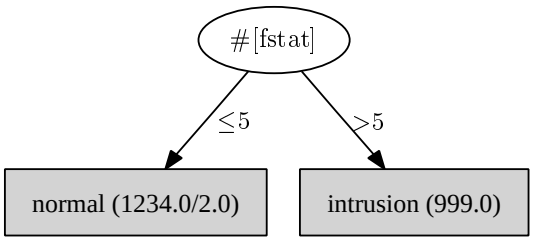


Figure 3.3 C4.5 decision tree for UNM live lpr

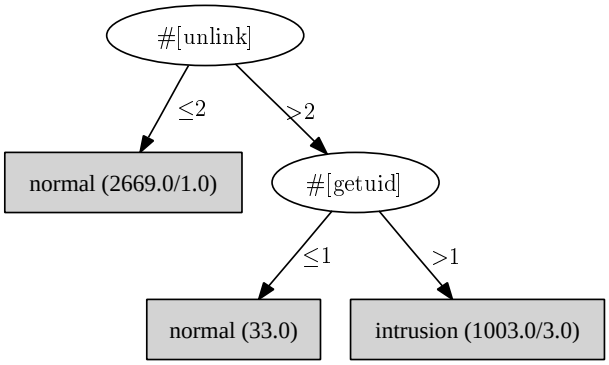


Figure 3.4 C4.5 decision tree for UNM live lpr MIT

Figure 3.3 is the decision tree by C4.5 for the "UNM live lpr" data set.

Though this simple rule does not have a perfect detection rate, it says that "we can guess the input lpr program trace is an intrusion sequence if the number of occurrences of 'fstat' system calls is more than 5". Therefore, a simple counter program that counts the number of certain system calls can detect intrusion before the process ends. However, unlike the approach that detects foreign contiguous subsequences, the counter program may not detect intrusion just after foreign subsequences are executed.

Figure 3.4 is the decision tree produced by C4.5 for "UNM live lpr MIT" data set. Though both the "UNM live lpr" data and the "UNM live lpr MIT" data contain intrusion snapshots by "*lprcp*" scripts, the generated rule may not always be the same because of different system environments.

The reason that this difference in frequency matters in classification is that the programs compromised by the intruder will have more codes (intrusion codes) which will be executed

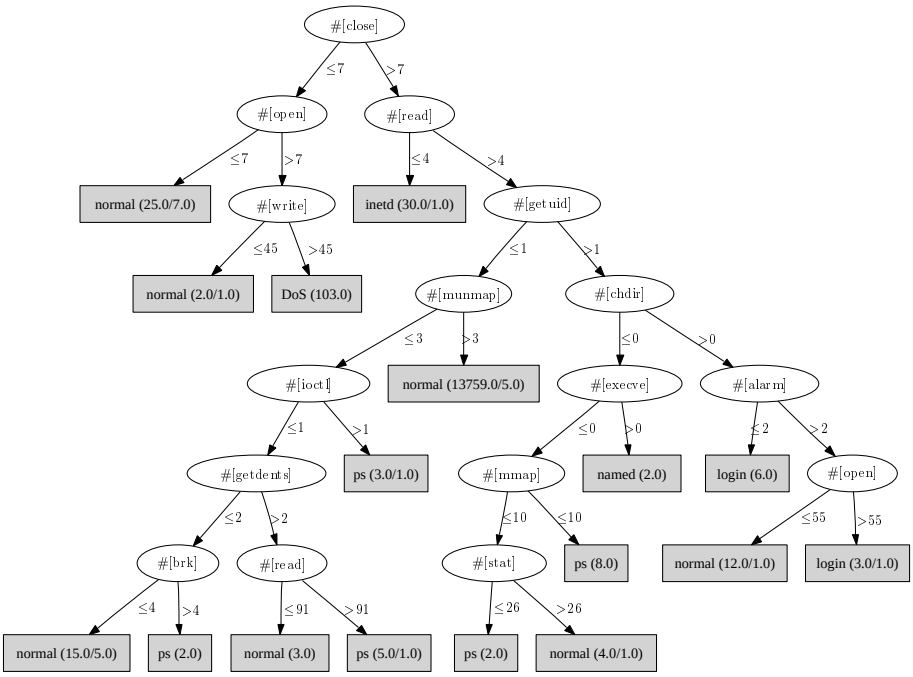


Figure 3.5 C4.5 decision tree for multiple intrusions in Linux from UNM data

during the routine execution of the programs, causing a change in the distribution of system calls. In the decision trees of figure 3.3 and 3.4, it can be seen that intrusion lies under 'greater than (>)' arc. It is because adding intrusion codes in the original program increases the counts of those system calls ('fstat', 'unlink', and 'getuid') in the decision trees.

Figure 3.5 shows the decision tree produced by C4.5 for multiple intrusions. The intrusions are "inted", "denial of service attack", "ps", "login", and "named" in UNM data. The figure indicates that this kind of decision tree can be an intrusion detector for composite attacks.

3.5.3 Experimental Results on Supervised Anomaly Detection

For supervised anomaly detection, we used one class Naive Bayes algorithm. In one class Naive Bayes, we calculate the probability distribution of the training data instead of the class label conditional probability distribution. For test sequences, we calculated symmetric Kullback-Liebler divergence (Kullback and Leibler, 1951; Li and Vitanyi, 1993) between the learned distribution and the distribution of test sequence in bag of system calls representation.

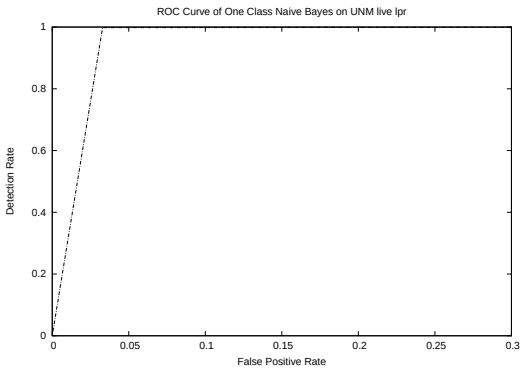


Figure 3.6 ROC Curve of One class Naive Bayes on UNM live lpr (θ = 0.43)

If the divergence is under a user-specified threshold θ, then the test sequence is considered to be similar to the learned distribution.

In figure 3.6, we show the result of the one class Naive Bayes algorithm in a bag of system calls representation on "UNM live lpr" data.

One class Naive Bayes performs effectively on the "UNM live lpr" data set, but does not perform effectively on some of other data sets, especially when the data set is imbalanced.

3.5.4 Experimental Results on Unsupervised Anomaly Detection

In unsupervised anomaly detection, the learning algorithm assumes that the input data set is composed of normal sequences and intrusion sequences. Therefore, it assumes the data distribution is a mixture of the distribution of normal seqeunces and the intrusion seqeunces. We use k-Means clustering with k set to 2 for clustering normal and intrusion distributions. We evaluated the clustering based approach on "UNM live lpr" and "UNM synthetic sendmail" data. The results are shown in table 3.3.

From the results in the table, k-means clustering is effective in unsupervised anomaly detection on 'UNM live lpr' data but not on 'UNM synthetic sendmail' data. Hence, there is a need for more sophisticated approaches for unsupervised anomaly detection.

3.6 Summary and Discussion

In this study, we have explored the use of a simple *bag of system calls* representation of system call sequences for intrusion detection. We constructed decision tree, Naive Bayes, decision list, and SVM and Logistic Regression classifiers for misuse detection. We constructed one class Naive Bayes algorithm and K-Means clustering for anomaly detection. In addition to the fact that we can use those standard machine learning methods, the proposed 'bag of system calls' representation has significant computational advantages over other approaches that have been reported in the literature.

Results of our experiments using widely used benchmark data sets - the University of New Mexico (UNM) and MIT Lincoln Lab (MIT LL) system call sequences show that the performance of the proposed approach in terms of detection rate and false positive rate is comparable or superior to that of previously reported data mining approaches to misuse detection. In particular, as shown in table 3.2, the proposed methods achieve nearly 100% detection rate with almost 0% false positive rate on all the data sets studied with the exception of two synthetic data sets ('UNM synthetic sendmail' and 'UNM synthetic sendmail CERT'). It is important to note that the reported performance measures were estimated using 10 fold cross-validation which ensures no overlap between training data and test data.

3.6.1 Discussion

When compared with the widely used fixed-length contiguous subsequence models, the *bag of system calls* representation explored in this study may seem somewhat simple. It may be argued that much more sophisticated models that take into account the identity of the user or perhaps the order in which the calls were made. But our experiments show that a much simpler approach may be adequate in many scenarios. The results of experiments described in this study show that it is possible to achieve nearly perfect detection rates and false positive rates using a data representation that discards the relationship between system call and originating process as well as the sequence structure of the calls within the traces.

Forrest et al. (Forrest et al., 1996; Warrender et al., 1999) showed that it is possible to

achieve accurate anomaly detection using fixed-length contiguous subsequence representation of input data. In their approach, the detector will find anomalous subsequences right after they are executed depending on user-specified thresholds. The proposed '*bag of system calls* representation has advantages that learning is faster, memory requirements are significantly lower, and simple counter program can discriminate normal sequences and abnormal sequences very quickly, before the process is terminated.

In these respects, a bag of system call representation is very suitable for protecting well known attacks and trivially modified attacks for IDS under time and space constraints. If the IDS needs to be built in real-time and the built system must be as light as possible to be able to work over limited resources such as sensor networks (Akyildizy et al., 2005), our approach will be a perfect fit because the generated IDS is simple and powerful to detect well known attacks. However, if the attacker knows the intrusion detection mechanism, our approach can be deceived by mimicry attacks (Wagner and Soto, 2002). Our future work will be focused on addressing this problem.

3.6.2 Related Work

Liao and Vermuri (Liao and Vemuri, 2002) used k-Nearest Neighbor (kNN) algorithm to classify normal and intrusive system call traces. They tested the kNN classifier on 1998 MIT Lincoln Lab BSM data and obtained effective detection rate and low false positive rate. However, they did not perform the detailed analysis with various machine learning techniques and multiple data sets. Their algorithm did not generate comprehensible rule sets for intrusive programs, which is very important for intrusion detection system and analysis.

Warrender, Forrest, and Pearlmutter (Warrender et al., 1999) have presented several intrusion detection methods based upon system call trace data. They tested a method that utilizes sliding windows to determine a database of *normal* sequences to form a database for testing against test instances. They then used a similar method to compare windows in the test instances against the database and classify instances according to a function of the similarity of these sequences to those in the *normal* sequence database. The function requires

sequential analysis of a window of system calls for each call made by a process. This requires the maintenance of a large database of *normal* system call trace sequences.

The same authors have described a rule-based classification method that requires alterations to the training data to learn. This model involves prediction of the next system call to be made by a process given some number of calls made immediately before. This method requires enumeration of all unique system call traces within a given program. This is quite demanding on a learner, especially in a situation where the datasets are quite large indeed. Even the space requirements are quite large relative to the input dataset. Finally, classification time is high for such methods because (in the worst case) each rule needs to be checked for each input instance.

Warrender et al. have presented Hidden Markov Model (HMM) methods for intrusion detection. Although this method does not require modification of the input dataset, it does require individual examination of each dataset to determine the optimal HMM to attempt to learn in each case. While this requirement does not seem overly demanding, we would prefer a method which allows classification of multiple input datasets in the same format if possible. Additionally construction of accurate HMM models can be quite demanding in terms the amount of training data as well as computational effort. Warrender, et al. observe that, for a process that makes S system calls, S states (and thus $2S^2$ values) must be computed. Datasets of interest in practice contain large amounts of processes (eight hours per day worth in the case of the MIT Lincoln Labs datasets), and each process makes a large number of system calls throughout its lifetime. Computing even polynomially many values for each instance becomes a problem at this scale.

Normalized frequency of audit data was used in SRI NIDES (Anderson et al., 1995). In NIDES, probability distribution of long term behavior of a program is generated and maintained as its profile. For detecting the anomalous behavior of the program, the profile is compared with short term behavior of the program, which is also maintained as probability distribution, using a statistical test similar to χ^2 test. The behavior of a program is characterized by its audit data such as file access, CPU usage, etc. We maintain the raw count of

system calls that are sequentially observed from the program as its profile, but this approach can be applied to other types of audit data. In some machine learning algorithms, raw counts are normalized and statistically compared with new behavior of the program. The Naive Bayes learning algorithm, which is one of the learning algorithms reported in this study, generates class-conditional probability distributions and prior distributions of the raw counts and statistically compares them with new distribution from the new behavior of the program. Moreover, as we showed, our profile representation can be used effectively with various machine learning algorithms.

One of the most popular rule induction techniques used in IDS is Repeated Incremental Pruning to Produce Error Reduction (RIPPER) rule learning algorithm (Cohen, 1995). Lee et al. (Lee and Stolfo, 1998) used RIPPER on a set of substrings of length 7 generated by the sliding window from *sendmail* system call traces. The generated rules are based on the insight that intrusion can be captured from the fixed-length substrings. For example, the rule *'normal:* $p_2 = 104, p_7 = 112$*'* means *'if* p_2 *is 104 and* p_7 *is 112 then the substring is normal'.* This approach, as in the case of *STIDE*, employs a user-supplied threshold to determine if the input trace is normal or intrusive. We applied RIPPER on a bag of system calls representation, and we obtained rules based on counts such as *'(count(fcntl)* $\geq$ *1) and (count(rename)* $\leq$ *0) and (count(read)* $\geq$ *5)* $\rightarrow$ *class=intrusion'* where *count(X)* returns the number of occurrence of system call X in the input trace. The rules generated by our method apply to the entire system call trace (as opposed to fixed length substring of traces). In our case, the relevant thresholds are learned directly from the training data, thereby avoiding the necessity of user-supplied thresholds.

Supervised learning techniques like Multi Layer Perceptron (MLP) with Error Back Propagation (Ghosh and Schwartzbard, 1999) have been investigated, as have unsupervised learning techniques like Self-Organized Feature Map (SOM) (Gunes Kayacik, 2003). Kang et al. (Kang et al., 2005a) used principal component analysis (PCA) and time-delay neural network (TDNN) for mutated attacks. In their model, a network packet is considered as a gray level image where each byte of a packet is represented a pixel. Chebrolua et al. (Chebrolua et al., 2005) used

Bayesian Network and Classification and Regression Trees (CART) to detect important features for intrusion detection. Spencer (Spencer, 2005) used artificial neural network to detect anomalies in wireless devices. Jiang et al. (Jiang et al., 2005) combined neural network with hidden Markov model (HMM) for efficient intrusion detection. Cha et al. (Cha et al., 2005) designed neural network system using Soundex algorithm to find anomalous behavior patterns. Xu and Xie (Xu and Xie, 2005) introduced Markov reward process model for the behavior of the system call sequences and converted the intrusion detection to predicting the value function of the Markov reward process. Yang et al. (Yang et al., 2005) designed intrusion detection system based on radial basis function (RBF) and compared the performance with back propagation network. Yang (Yang, 2005) viewed intrusion detection task as a case of data mining applied to time series. He used autoregressive moving average (ARMA) and Hopfield models to analyze the time series. Gao et al. (Gao et al., 2005) proposed a method of applying principal component neural networks for intrusion feature extraction. The extracted features are employed by SVM for classification. Using neural networks generally requires the specification of hidden nodes, and the generated model from learning neural network is hard to comprehend. Sy (Sy, 2005) defined the access signature as the collection of the statistically significant association patterns of 4th order using mutual information from the sequence of UNIX command data and used the signature for masquerader detection. Lu et al. (Lu et al., 2005) used several data mining techniques such as clustering, classification, and association rules to maximize the effectiveness in identifying attacks, thereby helping the users to construct more secure information systems. Jiang et al. (Jiang et al., 2006) proposed a novel method to compute the cluster radius threshold. They perform the data classification by an improved nearest neighbor (INN) method and presented a powerful clustering-based method for the unsupervised intrusion detection (CBUID).

All of these approaches use n-gram representation for modeling intrusion, but our approaches uses a bag of system calls representation.

Peddabachigaria et al. (Peddabachigaria et al., 2005) modeled intrusion detection system using decision tree and support vector machines (SVM). Their hybrid system combined clas-

sifiers to maximize the accuracy and had more accurate results. One class support vector machines (OCSVM) which can be useful in learning unlabeled data sets, are used for supervised anomaly detection by a few researchers. Heller et al. (Heller et al., 2003) compared OCSVM with probabilistic anomaly detection (PAD) algorithm for Windows Registry data, and concluded that well-defined kernels are important to enhance the performance of OCSVM. Lee et al. (Lee et al., 2005) proposed Multi-step Multi-class Intrusion Detection System (MMIDS), which alleviates some drawbacks associated with misuse detection and anomaly detection. The MMIDS consists of a hierarchical structure of one-class SVM, novel multi-class SVM, and incremental clustering algorithm: Fuzzy-ART. Yilmazel et al. (Yilmazel et al., 2005) compared bag of words representation (BOW) and NLP based representation for both typical and one-class classification problem using SVM algorithm. Ma et al. (Ma et al., 2005) implemented multi-class SVMs (one-versus-rest, one-versus-rest method and a new Decision Tree (DT) SVM) for intrusion detection. They also applied a support vector (SV) reduction algorithm and found that it decreases the training time dramatically while improves the detection rate. Steinwart et al. (Steinwart et al., 2005) interpreted anomaly detection as a binary classification problem of finding level sets for the data generating density. They compared the corresponding classification risk with the standard performance measure for the density level problem, and found that the empirical classification risk can serve as an empirical performance measure for the anomaly detection. According to the interpretation, they proposed a support vector machine (SVM) for anomaly detection and compared their SVM to other commonly used methods including the standard one-class SVM.

For protein classification, Leslie et al. introduced spectrum kernel (Leslie et al., 2002a) and mismatch kernel (Leslie et al., 2002b). Spectrum kernel is for k-length continuous subsequences, and mismatch kernel is similar to spectrum kernel but mismatches are allowed. Tian et al. (Tian et al., 2004) developed string kernel for intrusion detection. Their kernel penalizes non-continuous occurrences and feature map is indexed by all possible subsequences. Our future work includes one class SVM experiment on a bag of words feature representation.

Liu et al. (Liu et al., 2005a) investigated three system-call-based feature representations

for "insider threat" and "external threat": n-grams of system call names, histograms of system call names, and individual system calls with associated parameters, and found that none of these representations consistently performs as well when dealing with the internal threat as previous results show for external threat detection.

Recently, string alignment techniques has been used for intrusion detection and worm detection. Using string alignment techniques for intrusion detection is one of our important future work.

Coull et al. (Coull et al., 2003) first proposed bio-informatics techniques for intrusion detection. They used a semi-global alignment and unique scoring function for detecting intrusive sequences. Takeda (Takeda, 2005) also applied bio-informatics techniques for network intrusion detection. Tripp (Tripp, 2005) describes a finite state machine approach to string matching for an intrusion detection system. To obtain high performance, he designed a hardware for a parallel string matching. Newsome et al. (Newsome et al., 2005) used an adaptation of the Smith-Waterman (Smith and Waterman, 1981) algorithm to find an alignment for generating signatures, which are applied to match polymorphic worm payloads. Tang and Chen (Tang and Chen, 2005) introduced position-aware distribution signature (PADS), which fits in the gap between the traditional signatures and the anomaly-based systems, and proposed two algorithms based on Expectation-Maximization (EM) and Gibbs Sampling for efficient computation of PADS from polymorphic worm samples. Jiang and Xu (Jiang and Xu, 2005) proposed behavioral footprinting. They modeled each infection step as a behavior phenotype and the entire infection session as a sequential behavioral footprint, and presented advanced sequence analysis techniques to extract a worm's behavioral footprint from its infection traces

3.6.3 Future Work

Some directions for future work include:

- Further formalizing intrusive behaviors of multiple processes as a *multi-bag* in IDS framework. A process often fork child processes during the execution and intrusion can be a cooperative work between the processes in the same group. Thus, it is a more natural

idea than the conventional approaches to model an IDS to consider cooperative attacks. However, the research area of a host-based IDS that monitors multiple process' cooperative behavior has not been explored. Considering this observation, we will propose a theoretical framework for IDS that models multiple processes as a multi-bag.

- Extending the supervised anomaly detection experiments with one-class support vector machines (Scholkopf et al., 2001; Leslie et al., 2002a,b; Tian et al., 2004) or other anomaly detection techniques in SVM (Tax and Duin, 2004)

- Extending feature representation so that subsequences rather than system calls can be dealt with by the existing machine learning techniques in an efficient way. We believe it will show better performance in terms of accuracy/detection rate/false positive rate in supervised anomaly detection

- Modeling multiple processes' behavior in one trace. Current intrusion detection system assumes one process produce intrusions in the intrusion model. Modeling multiple processes that are cooperative is more probable for future intrusion detection system

- Performing experiments on different operating system such as Microsoft Windows. Anomaly detection of spyware in Windows is a good example

- Applying generalized global alignment (Huang and Chao, 2003; Takeda, 2005; Coull et al., 2003) of system call sequences

- Using system call arguments (Mutz et al., 2006) with abstraction (Kang et al., 2004) for anomaly detection

Table 3.2 Experimental results of misuse detection estimated using 10 fold cross-validation with 95% confidence interval

Program	Naive Bayes Multinomial	C4.5	RIPPER	SVM	Logistic Regression
UNM live lpr					
accuracy	83.43±1.54	99.91±0.12	99.91±0.12	100.00±0.00	99.91±0.12
detection rate	100.00±0.00	99.80±0.19	99.80±0.19	100.00±0.00	100.00±0.00
false positive rate	30.03±1.90	0.00±0.00	0.00±0.00	0.00±0.00	0.16±0.17
UNM live lpr MIT					
accuracy	54.52±1.60	99.89±0.11	99.86±0.12	99.83±0.13	99.97±0.06
detection rate	100.00±0.00	99.90±0.10	99.80±0.14	99.80±0.14	99.90±0.10
false positive rate	62.31±1.56	0.11±0.11	0.11±0.11	0.14±0.12	0.00±0.00
UNM synthetic sendmail					
accuracy	20.21±4.09	94.87±2.24	94.33±2.35	95.68±2.07	95.41±2.13
detection rate	92.00±2.76	40.00±4.99	48.00±5.08	40.00±4.99	64.00±4.88
false positive rate	84.97±3.64	1.15±1.08	2.31±1.53	0.28±0.54	2.31±1.53
UNM synthetic sendmail CERT					
accuracy	24.39±4.65	96.64±1.95	95.42±2.26	96.03±2.11	96.03±2.11
detection rate	100.00±0.00	85.29±3.83	82.35±4.13	64.70±5.17	82.35±4.13
false positive rate	84.35±3.93	2.04±1.53	3.06±1.86	0.34±0.63	2.38±1.65
UNM denial of service					
accuracy	98.70±0.19	99.97±0.03	99.96±0.03	99.98±0.02	99.97±0.03
detection rate	44.76±0.83	99.04±0.16	98.09±0.23	100.00±0.00	99.04±0.16
false positive rate	0.88±0.16	0.02±0.02	0.02±0.02	0.01±0.02	0.01±0.02
MIT LL 1998 4^{th} Week					
Monday					
accuracy	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
detection rate	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
false positive rate	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
Tuesday					
accuracy	99.55±0.62	99.55±0.62	99.55±0.62	99.55±0.62	99.55±0.62
detection rate	98.60±1.09	98.60±1.09	98.60±1.09	98.60±1.09	98.60±1.09
false positive rate	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
Thursday					
accuracy	99.73±0.53	99.73±0.53	99.73±0.53	99.73±0.53	99.73±0.53
detection rate	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
false positive rate	0.04±0.20	0.04±0.20	0.04±0.20	0.04±0.20	0.04±0.20
Friday					
accuracy	98.80±1.35	98.80±1.35	98.80±1.35	98.80±1.35	98.80±1.35
detection rate	89.28±3.83	89.28±3.83	89.28±3.83	89.28±3.83	89.28±3.83
false positive rate	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

Table 3.3 Results of K-means clustering for unsupervised anomaly detection, estimated using 10 fold cross-validation with 95% confidence interval

Program	Accuracy	Detection Rate	False Positive
UNM live lpr	99.28±0.35	100.00±0.00	1.29±0.47
UNM synthetic sendmail	80.32±4.05	40.00±4.99	16.76±3.80

CHAPTER 4. RECURSIVE NAIVE BAYES LEARNER

This chapter is based on the paper published in the Tenth Pacific-Asia Conference on Knowledge Discovery and Data Mining (Kang et al., 2006). But, there have been added considerably more contents and experimental results in this chapter.

4.1 Abstract

Naive Bayes (NB) classifier relies on the assumption that the instances in each class can be described by a *single* generative model. This assumption can be restrictive in many real world classification tasks. We describe recursive Naive Bayes learner (RNBL), which relaxes this assumption by constructing a tree of Naive Bayes classifiers for sequence classification, where each individual NB classifier in the tree is based on an event model (one model for each class at each node in the tree). In our experiments on protein sequences, Reuters newswire documents and UC-Irvine benchmark data sets, we observe that RNBL substantially outperforms NB classifier. Furthermore, our experiments on the protein sequences and the text documents show that RNBL outperforms C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion) and yields accuracies that are comparable to those of support vector machines (SVM) using similar information.

4.2 Introduction

Naive Bayes (NB) classifiers, due to their simplicity and modest computational and training data requirements, are among the most widely used classifiers on many classification tasks, including text classification tasks (McCallum and Nigam, 1998) and macromolecular sequence classification tasks that arise in bio-informatics applications (Andorf et al., 2004). NB classifiers

belong to the family of generative models (a model for generating data given a class) for classification. Instances of a class are assumed to be generated by a random process which is modeled by a generative model. The parameters of the generative model are estimated (in the case of NB) assuming independence among the attributes given the class. New instances to be classified are assigned to the class that is the most probable for the instance.

NB classifier relies on the assumption that the instances in each class can be described by a *single* generative model (i.e., probability distribution). According to Langley (Langley, 1993), this assumption can be restrictive in many real world classification tasks. One way to overcome this limitation while maintaining some of the computational advantages of NB classifiers is to construct a tree of NB classifiers. Each node in the tree (a NB classifier) corresponds to one set of generative models (one generative model per class), with different nodes in the tree corresponding to different generative models for a given class. Langley described a recursive NB classifier (RBC) for classifying instances that are represented by ordered tuples of nominal attribute values. RBC works analogous to a decision tree learner (Quinlan, 1993), recursively partitioning the training set at each node in the tree until the NB classifier of the node simply cannot partition the corresponding data set. Unlike in the case of the standard decision tree, the branches out of each node correspond to the most likely class lebels assigned by the NB classifier at that node. In cases where each class cannot be accurately modeled by a single Naive Bayes generative model, the subset of instances routed to one or more branches belong to more than one class. RBC models the distribution of instances in a class at each node using a Naive Bayes generative model. However, according to Langley's reports of experiments on some of the UC-Irvine benchmark data sets, the recursive NB classifier did not yield significant improvements over standard NB classifier (Langley, 1993).

In this study, we revisit the idea of recursive NB classifier in the context of text/sequence classification tasks and most of the UC-Irvine benchmark data sets. We describe RNBL, an algorithm for constructing a tree of Naive Bayes classifiers for sequence classification with two different event models and two stopping criteria. For text and sequence classification, each NB classifier in the tree is based on a multinomial event model (McCallum and Nigam, 1998)

(one for each class at each node in the tree). Our choice of the multinomial event model is influenced by its reported advantages over the multivariate event model of sequences (McCallum and Nigam, 1998) in text classification tasks. For UC-Irvine benchmark data sets, RNBL uses multivariate event model. RNBL works in a manner similar to Langley's RBC, recursively partitioning the training set of labeled sequences at each node in the tree until a stopping criterion is satisfied. The branches out of each node correspond to the most likely class assigned by the NB classifier at that node. As for the stopping criterion, RNBL uses either a conditional minimum description length (CMDL) score for the classifier (Friedman et al., 1997) or area under the ROC curve (AUC). The CMDL score is specifically adapted to the case of RNBL based on the CMDL score for the NB classifier using the multinomial event model for sequences (Kang et al., 2005d). Previous reports by Langley (Langley, 1993) in the case of a recursive NB classifier (RBC) for data sets whose the instances are represented as tuples of nominal attribute values (such as the UC-Irvine benchmark data), suggested that the tree of NB classifiers offered little improvement in accuracy over the standard NB classifier. In our experiments on protein sequence and text classification tasks, we observe that RNBL substantially outperforms NB classifier. Also, contrary to Langley (Langley, 1993) report, RNBL mostly outperforms the standard Naive Bayes learner for the UC-Irvine benchmark data sets. Furthermore, our experiments show that, for the text and sequence classification tasks, RNBL outperforms C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion) and yields accuracies that are comparable to those of SVM using similar information.

The rest of the chapter is organized as follows: Section 4.3 briefly introduces the event models for sequence classification; Section 4.4 presents RNBL (recursive Naive Bayes learner) algorithm in detail; Section 4.5 presents our experimental results; Section 4.6 concludes with summary and discussion.

4.3 Event Models for Naive Bayes Sequence Classification

4.3.1 Multi-variate Bernoulli model

In a multi-variate Bernoulli model, a sequence d_j is represented as an instance I_j by a vector of binary values $b_{i,j} \in \{0, 1\}$ where $b_{i,j}$ denotes the presence or absence of a word w_i in the sequence. The number of occurrence of word is not preserved in the vector. The probability of sequence d_j given its class c_j is as follows:

$$P(d_j|c_j) = \prod_{i=1}^{|\Sigma|} (b_{i,j}p_{i,j} + (1 - b_{i,j})(1 - p_{i,j})) \tag{4.1}$$

4.3.2 Multinomial Event Model

Consider sequences defined over a finite alphabet $\Sigma = \{w_1 \cdots w_d\}$ where $d = |\Sigma|$. For example, in the case of protein sequences, Σ can be the 20-letter amino acid alphabet ($\Sigma = \{A_1, A_2, \ldots, A_{20}\}$). In the case of text, Σ corresponds to the finite vocabulary of words. Typically, a sequence $S_j \in \Sigma^\star$ is mapped into a finite dimensional feature space D through a mapping $\Phi : \Sigma^\star \rightarrow D$.

In a multinomial event model, a sequence S_j is represented by a *bag* of elements from Σ. That is, S_j is represented by a vector D_j of frequencies of occurrences in S_j of each element of Σ. Thus, $D_j =< f_{1j}, f_{2j}, \ldots, f_{dj}, c_j >$, where $f_{ij} \in \mathbb{Z}^*$ denotes the number of occurrences of w_i (the ith element of the alphabet Σ) in the sequence S_j. Thus, we can model the sequence S_j as a sequence of random draws from a multinomial distribution over the alphabet Σ. If we denote the probability of picking an element w_i given the class c_j by $P(w_i|c_j)$, the probability of sequence S_j given its class c_j under the multinomial event model is defined as follows:

$$P(X_1 = f_{1j}, \ldots, X_d = f_{dj}|c_j) = \left\{ \frac{\left(\sum_i^d f_{ij}\right)!}{\prod_i^d (f_{ij})!} \right\} \prod_{i=1}^{d} P(w_i|c_j)^{f_{ij}} \tag{4.2}$$

(Note: To be fully correct, we would need to multiply the right hand side of the above equation by $P(N|c_j)$, the probability of drawing a sequence of a specific length $N = (\sum_i^d f_{ij})$ given the class c_j, but this is hard to do in practice.)

Given a training set of sequences, it is straightforward to estimate the probabilities $P(w_i|c_j)$ using the Laplace estimator as $\hat{P}(w_i|c_j) = p_{ij} = \frac{Count_{ij}+1}{Count_j+d}$, where $Count_{ij}$ is the number of occurrences of w_i in sequences belonging to class c_j and $Count_j$ is the total number of words in training set sequences belonging to class c_j.

4.4 Recursive Naive Bayes Learner

4.4.1 RNBL Algorithm

As noted above, RNBL, analogous to the decision tree learner, recursively partitions the training data set using Naive Bayes classifiers at each node of the tree. The root of the tree is a Naive Bayes classifier constructed from the entire data set. The outgoing branches correspond to the different class labels, assigned by the Naive Bayes classifier.

For a given input training data set $D_0(= D_{current})$, we create a Naive Bayes classifier n_0. We compute the stopping criterion $Score_{current}$ for the classifier n_0 (See the later sections for details of the calculation of CMDL score and AUC score for recursive Naive Bayes classifier). The classifier n_0 partitions the data set D_0 into $|C|$ subsets based on the class labels assigned to the instances $\in D_0$ by the classifier n_0. Each such subset is in turn used to train additional Naive Bayes classifiers. At each step, the score for the resulting tree of Naive Bayes classifiers is computed and compared with the score of the classifier from the previous step. This recursive process terminates when additional refinements of the classifier yield no significant improvement in the score. Fig. 4.1 shows the pseudo-code of RNBL algorithm.

Analogous to a decision tree, the resulting classifier predicts a class label for a new instance as follows: starting at the root of the tree, the instance is routed along the outgoing branches of successive Naive Bayes classifiers, at each node following the branch corresponding to the most likely class label for the instance, until a leaf node is reached. Finally, The instance is assigned the label predicted by the classifier at the leaf node.

RNBL($D_{current}$) :
begin

1. **Input** : data set $D_0 = D_{current}$ // *data set*
2. Estimate probabilities based on D_0 that specify the Naive Bayes classifier n_0
3. Add n_0 to the current classifier $h_{current}$ if $n_0 \notin h_{current}$
4. $Score_{current} \leftarrow CMDL(h_{current}|D_0)$ or $AUC(h_{current}|D_0)$ // *CMDL/AUC score of the current classifier.*
5. Partition $D_{current}$ into $\mathbb{D} = \{D_1, D_2, \ldots, D_{|C|} | \forall_{S \in D_i} \forall_{j \neq i}, P(c_i|S) > P(c_j|S)\}$
6. For each $D_i \in \mathbb{D}$, estimate probabilities that specify the corresponding Naive Bayes classifiers n_i
7. $h_{potential} \leftarrow$ refinement of $h_{current}$ given D_i with the classifiers corresponding to each n_i based on the corresponding D_i in the previous step // *see Fig. 4.2 for details*
8. $Score_{potential} \leftarrow CMDL(h_{potential}|\sum_{i=0}^{|C|} D_i)$ or $AUC(h_{potential}|\sum_{i=0}^{|C|} D_i)$ // *CMDL/AUC score resulting from the refined classifier*
9. If $Score_{potential} > Score_{current}$ then // *accept the refinement*
10. Add each n_i to $h_{current}$
11. For each child node n_i
12. **RNBL**(D_i) // *recursion*
13. End For
14. End If
15. **Output** : $h_{current}$

end.

Figure 4.1 Recursive Naive Bayes Learner

4.4.2 CMDL for a Recursive Naive Bayes Classifier

RNBL employs the conditional minimum description length (CMDL) score (Friedman et al., 1997), specifically adapted to the case of RNBL, based on the CMDL score for NB classifier using multivariate and multinomial event models (Kang et al., 2005d) as the stopping criterion.

Let v_j be a set of attribute values of j^{th} instance $d_j \in D$, and $c_j \in C$ a class label associated with d_j. Then, the conditional log likelihood of the hypothesis B given data D is

$$CLL(B|D) = |D| \sum^{|D|} \log\{P_B(c|v)\} = |D| \sum^{|D|} \log \left\{ \frac{P_B(c)P_B(v|c)}{\sum_{c_i}^{|C|} P_B(c_i)P_B(v|c_i)} \right\} \tag{4.3}$$

For Naive Bayes classifier, this score can be efficiently calculated (Zhang and Honavar, 2004).

$$CLL(B|D) = |D| \sum^{|D|} \log \left\{ \frac{P_B(c) \prod^{v_i \in v} \{P_B(v_i|c)\}}{\sum_{c_i}^{|C|} P_B(c_i) \prod^{v_j \in v} \{P_B(v_j|c_i)\}} \right\}$$

This is the conditional log likelihood of Naive Bayes Multi-variate Bernoulli model (Zhang and Honavar, 2004).

And the corresponding conditional minimum description length (CMDL) score is defined as follows:

$$CMDL(B|D) = -CLL(B|D) + \left\{ \frac{\log |D|}{2} \right\} size(B)$$

where, $size(B)$ is a size of the hypothesis B which corresponds to the number of entries in conditional probability tables (CPT) of B.

In case of a Naive Bayes classifier with multi-variate Bernoulli model, $size(B)$ is defined as

$$size(B) = (|C| - 1) + |C| \sum_{i=1}^{|v|} (|v_i| - 1)$$

where $|C|$ is the number of class labels, $|v|$ is the number of attributes, and $|v_i|$ is the number of attribute values for an attribute v_i.

Combining the equations 4.2 and 4.3, we can obtain the conditional log likelihood of the classifier B given data D under the Naive Bayes multinomial model.

$$CLL(B|D) = |D| \sum_{j}^{|D|} \log \left\{ \frac{P(c_j) \left\{ \frac{\left(\sum_i^{|\Sigma|} f_{i,j}\right)!}{\prod_i^{|\Sigma|}(f_{i,j})!} \right\} \prod_i^{|\Sigma|} \{p_{i,j}^{f_{i,j}}\}}{\sum_k^{|C|} \left\{ P(c_k) \left\{ \frac{\left(\sum_i^{|\Sigma|} f_{i,k}\right)!}{\prod_i^{|\Sigma|}(f_{i,k})!} \right\} \prod_i^{|\Sigma|} \{p_{i,k}^{f_{i,k}}\} \right\}} \right\}$$

where, $|D|$ is the number of instances, $c_j \in C$ is a class label for instance $d_j \in D$, $f_{i,j}$ is a integer frequency of word $w_i \in \Sigma$ in instance d_j, and $p_{i,j}$ is the estimated probability that word w_i occurred in the instances associated to class label j.

Conditional Minimum Description Length (CMDL) of a Naive Bayes Classifier for the multinomial model is defined as follows:

$$CMDL(B|D) = -CLL(B|D) + \left\{ \frac{\log |D|}{2} \right\} size(B)$$

where, $size(B)$ is a size of the hypothesis B which corresponds to the number of entries in conditional probability tables (CPT) of B.

Therefore, $size(B)$ is estimated as

$$size(B) = (|C| - 1) + |C||\Sigma|$$

where $|C|$ is the number of class labels, and $|\Sigma|$ is the cardinality of the vocabulary (i.e. the number of all distinct words).

For more detailed description on CMDL score, please refer to section 2.4.2.3 and 2.4.2.4.

We observe that in the case of a recursive Naive Bayes classifier, $CLL(h|D)$ can be decomposed in terms of the CLL scores of the individual Naive Bayes classifiers at the leaves of the tree of classifiers. Consequently, the CMDL score for the composite tree-structured classifier can be written as follows:

$$CMDL(h|D) = \sum_{node \in Leaves(h)} CLL(h_{node}|D_{node}) - \left\{ \frac{\log |D|}{2} \right\} size(h),$$

where $size(h) = ((|C| - 1) + |C||\Sigma|)|h|$, denoting $|h|$ the number of nodes in h.

For example, Fig. 4.2 shows a Recursive Naive Bayes classifier consisting of 5 individual Naive Bayes classifiers. $\hat{c_+}$ and $\hat{c_-}$ are the predicted outputs of each hypothesis.

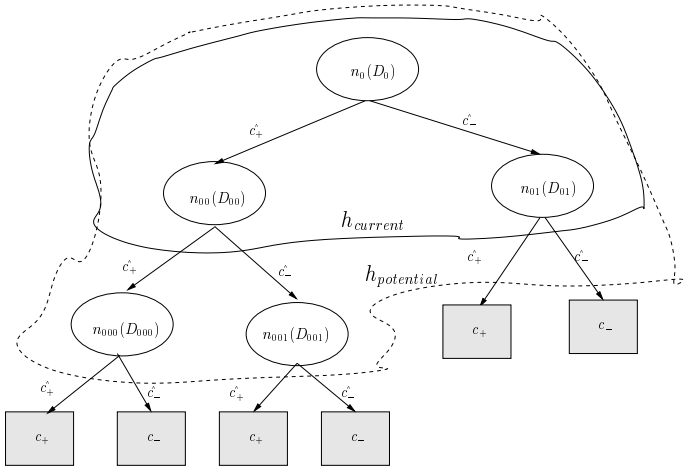


Figure 4.2 Recursion tree of classifiers. Note that $h_{potential}$ is the refinement of $h_{current}$ by adding nodes $n_{000}(D_{000})$ and $n_{001}(D_{001})$ as children of $n_{00}(D_{00})$.

In the figure,

$$CLL(h_{current}|D) = CLL(n_{00}|D_{00}) + CLL(n_{01}|D_{01})$$

and

$$CLL(h_{potential}|D) = CLL(n_{000}|D_{000}) + CLL(n_{001}|D_{001}) + CLL(n_{01}|D_{01}),$$

where $|C|$=2, $|h_{current}| = 3$, and $|h_{potential}| = 5$.

Using the CMDL score, we can choose the hypothesis h that effectively trades off the complexity, measured by the number of parameters, against the accuracy of classification. As is described in Fig. 4.1, the algorithm terminates when none of the refinements of the classifier (splits of the tree nodes) yields statistically significant improvement in the overall CMDL score.

4.4.3 Area Under the Curve (AUC) score for Naive Bayes Classifier

The receiver operating characteristics (ROC) curve was originally used in signal detection theory (Mason and Graham, 2002). ROC curve is usually drawn for soft binary classifiers on two dimension of true positive rate and false positive rate. A binary classifier is not soft if it assigns score $\in \{0, 1\}$ for each class label when it classify an instance. Thus, a soft classifier

can assign score $\in$ [0..1] for each class label, and the scores assigned to one instance can be different from those to other instances. From a binary classifier, we can easily draw ROC curve with the following definition of true positive rate and false positive rate.

$$\text{True positive rate} = \frac{\#\text{ of True positives}}{\#\text{ of True positives} + \#\text{ of False negatives}}$$
$$\text{False positive rate} = \frac{\#\text{ of False positives}}{\#\text{ of True negatives} + \#\text{ of False positives}}$$

ROC analysis (Provost and Fawcett, 1997) has been used in machine learning because it can deals with skew sensitivity when cost parameters are not known.

Area under the ROC curve (AUC) (Hand and Till, 2001) is a calculated area under the ROC curve. According to (Hand and Till, 2001), AUC, $\hat{A}$, is equivalent to Mann-Whitney-Wilcoxon sum of ranks test which is described as follows:

$$\hat{A} = \frac{S_{(+)} - \frac{Pos \times (Pos+1)}{2}}{Pos \times Neg}$$

where $S_{(+)}$ is the sum of the ranks of the positive class points, Pos is the number of instances with positive class, and Neg is the number of instances with negative class.

4.5 Experiments

To evaluate RNBL, recursive Naive Bayes learner, we conducted experiments using three classification tasks: (a) assigning Reuters newswire articles to categories, (b) classifying protein sequences in terms of their cellular localization, (c) and classification tasks on UC-Irvine benchmark data sets. The results of the experiments described in this section show that the classifiers generated by RNBL are typically more accurate than Naive Bayes classifiers, and that RNBL sometimes yields more accurate classifiers than C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion). Especially for text/sequence classification, RNBL yields accuracies that are comparable to those of linear kernel based SVM trained with the SMO algorithm (Platt, 1999) on a bag of letters (words) representation of sequences (text).

4.5.1 Reuters 21587 Text Categorization Test Collection

Reuters 21587 distribution 1.0 data set[1] consists of 12902 newswire articles in 135 overlapping topic categories. We followed the ModApte split (Apté et al., 1994) in which 9603 stories are used to train the classifier and 3299 stories to test the accuracy of the resulting classifier. We eliminated the stories that do not have any topic associated with them (i.e., no class label). As a result, 7775 stories were used for training and 3019 stories for testing the classifier.

Because each story has multiple topics (class labels), we built binary classifiers for the top ten most populous categories following the setup used in previous studies by other authors (Dumais et al., 1998; Joachims, 1998; McCallum and Nigam, 1998; Sandler, 2005; Keerthi, 2005; Joachims, 2005; Gabrilovich and Markovitch, 2005; Carvalho and Cohen, 2005; Rooney et al., 2006; Zhang and Lee, 2006).

In our experiments, stop words were not eliminated, and title words were not distinguished from body words. Following the widely used procedure for text classification tasks with large vocabularies, we selected top 300 features based on mutual information with class labels. The mutual information $MI(x, c)$ between a feature x and a category c is defined as follows:

$$MI(x,c) = \sum^{x} \left\{ \sum^{c} \left\{ P(x,c) log \frac{P(x,c)}{P(x)P(c)} \right\} \right\}$$

For evaluation of the classifiers, following the standard practice in text classification literature, we report the break-even points, which is the average of precision and recall when the difference between the two is minimum. Precision and recall are defined as follows:

$$\text{Precision} = \frac{|\text{detected documents in the category}|}{|\text{documents in the category}|} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{Recall} = \frac{|\text{detected documents in the category}|}{|\text{detected documents}|} = \frac{\text{TP}}{\text{TP+FP}}$$

Table 4.1 shows the break-even points of precision and recall as a performance measure for the ten most frequent categories. The results in the table show that, RNBL outperforms the

[1]This collection is publicly available at http://www.daviddlewis.com/resources/testcollections/reuters21578/.

Table 4.1 Break-even point of precision and recall (a standard accuracy measure for ModApte split of Reuters 21587 data set) on the 10 largest categories of Reuters 21587 data set.

	Data		NBL	RNBL	C4.5	SVM
name	# train (+/−)	# test (+/−)	point	point	point	point
earn	2877 / 4898	1087 / 1932	94.94	96.50	95.58	**97.24**
acq	1650 / 6125	719 / 2300	89.43	**93.32**	89.29	92.91
money-fx	538 / 7237	179 / 2840	64.80	69.83	69.27	**72.07**
grain	433 / 7342	149 / 2870	74.50	**89.26**	85.23	**89.26**
crude	389 / 7386	189 / 2830	79.89	77.78	76.19	**86.77**
trade	369 / 7406	117 / 2902	59.83	70.09	61.54	**71.79**
interest	347 / 7428	131 / 2888	61.07	70.99	64.89	**73.28**
ship	197 / 7578	89 / 2930	**82.02**	**82.02**	65.17	80.90
wheat	212 / 7563	71 / 2948	57.75	73.24	**87.32**	80.28
corn	181 / 7594	56 / 2963	57.14	67.85	**92.86**	76.79

other algorithms, except SVM, in terms of classification accuracy for Reuters 21587 text data set.

Figure 4.3 shows Precision-Recall curve (Fawcett, 2003, 2006) for the "Earn" category. It can also be seen that RNBL compares favorably with Naive Bayes and C4.5 decision tree learner.

4.5.2 Protein Subcellular Localization Prediction

We applied RNBL to two protein sequence data sets, where the goal is to predict the subcellular localization of the proteins (Reinhardt and Hubbard, 1998; Andorf et al., 2006).

The first data set consists of 997 prokaryotic protein sequences derived from SWISS-PROT database (release 33.0) (Bairoch and Apweiler, 2000). This data set includes proteins from three different subcellular locations: cytoplasmic (688 proteins), periplasmic (202 proteins), and extracellular (107 proteins).

The second data set contains 2427 eukaryotic protein sequences derived from SWISS-PROT database (release 33.0) (Bairoch and Apweiler, 2000). This data set includes proteins from the following four different subcellular locations: nuclear (1097 proteins), cytoplasmic (684

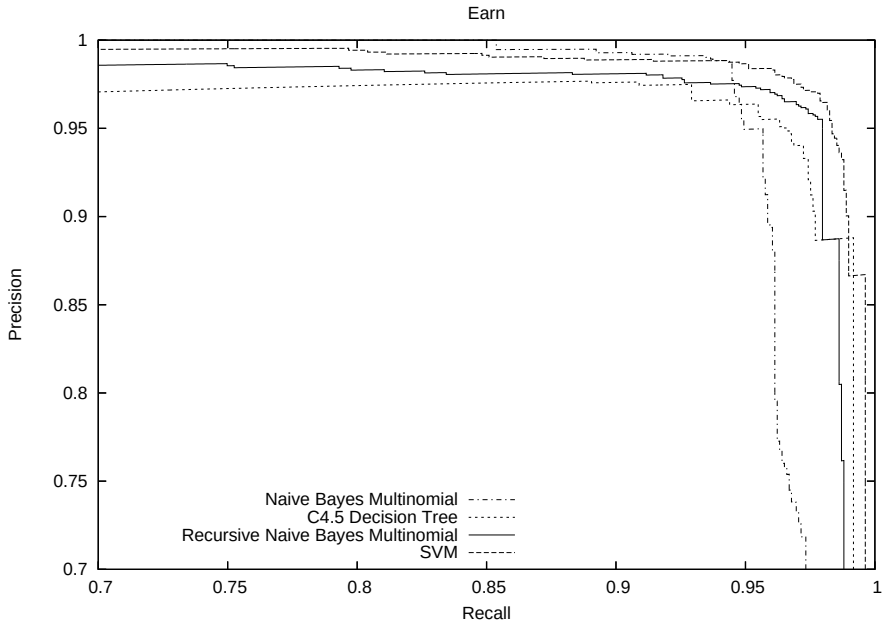


Figure 4.3 Precision-Recall Curves of "Earn" Category

proteins), mitochondrial (321 proteins), extracellular (325 proteins).

The accuracy, sensitivity, and specificity of the classifiers (estimated using 10-fold cross-validation) on the two data sets [2] are shown in Table 4.2. Each measure is defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP+TN+FP+FN}}$$

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

$$\text{Specificity} = \frac{\text{TP}}{\text{TP+FP}}$$

where, TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives, and FN is the number of false negatives.

The results show that RNBL generally outperforms C4.5, and compares favorably with SVM. Specificity of SVM for 'Mitochondrial' is "N/A", because the SVM classifier always outputs negative when most of the instances in the data set have negative class label (imbalanced), which leads its specificity to be undefined.

[2] These two datasets are available to download at http://www.doe-mbi.ucla.edu/~astrid/astrid.html.

Figure 4.4 shows receiver operating characteristic (ROC) curves of Naive Bayes, recursive Naive Bayes, C4.5, and SVM classifiers applied to 'Peripalsmic Prokaryotic' and 'Cytoplasmic Eukaryotic' protein sequences for the same task (predicting localization) described in the table 4.2. The areas under the curve (AUC) for Naive Bayes, recursive Naive Bayes, C4.5, and SVM classifiers for 'Peripalsmic Prokaryotic' protein sequences are 0.8494, 0.9311, 0.7379, and 0.8708 respectively (shown in figure 4.4(a)), and for 'Cytoplasmic Eukaryotic' protein sequences, 0.7985, 0.9689, 0.7320, and 0.7949 respectively (shown in figure 4.4(b)). The graphs in the figure 4.4 show that AUC for recursive Naive Bayes classifier is higher than other classifiers, which means that RNBL is a promising algorithm for the task.

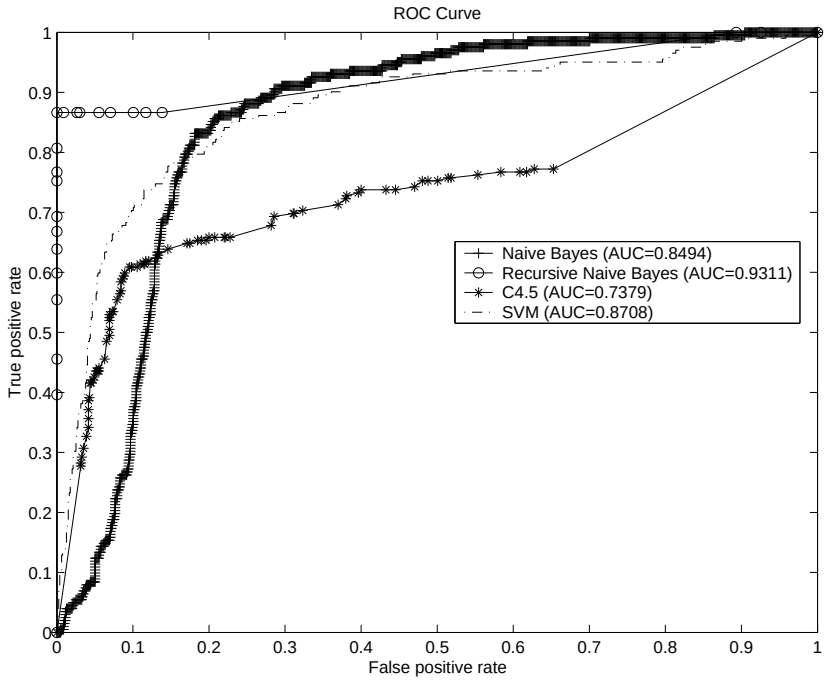
The third data set[3] comprises a total of 7589 eukaryotic proteins derived from SWISS-PROT data base (realease 39.0) (Bairoch and Apweiler, 2000). The data set includes proteins from the following twelve different subcellular locations (number of proteins in parentheses): chloroplast (671), cytoplasmic (1245), cytoskeleton (41), endoplasmic reticulum (114), extracellular (862), golgi apparatus (48), lysosomal (93), mitochondrial (727), nuclear (1932), peroxisomal (125), plasma membrane (1677), and vacuolar(54). This dataset is adapted from the protein localization prediction studies of (Cai et al., 2002; Park and Kanehisa, 2003).

Table 4.3 indicates that, in most cases, RNBL generates classifiers that have higher accuracy than those of C4.5 decision tree learner, and are comparable to those of SVM.

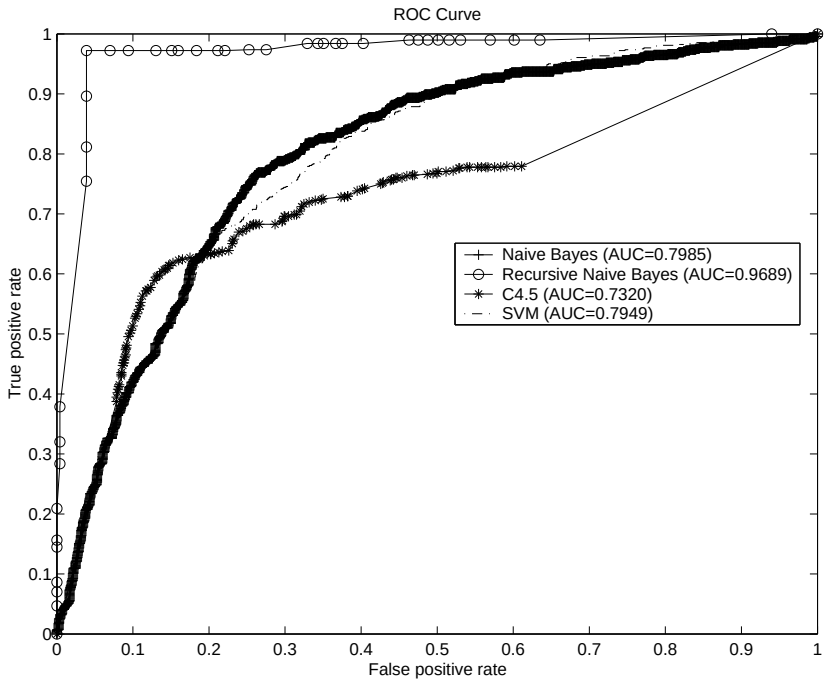
'Vacuolar' localization entry of C4.5 is "N/A" because both TP and FP are zero, which leads to undefined Specificity. The reason is that the classifier always outputs negative because the data set is imbalanced. Most Specificity entries for SVM in the table are also "N/A" for the same reason.

Figure 4.5 shows ROC curves of Naive Bayes, recursive Naive Bayes, C4.5, and SVM classifiers applied to 'Extracellular' and 'Nuclear' protein sequences for the same task (predicting localization) described in the table 4.3. The areas under the curve (AUC) for Naive Bayes, recursive Naive Bayes, C4.5, and SVM classifiers for 'Extracellular' protein sequences are 0.7600, 0.8228, 0.7673, and 0.8139 respectively (shown in figure 4.5(a)), and for 'Nuclear'

[3]These datasets are available to download at http://web.kuicr.kyoto-u.ac.jp/~park/Seqdata/.



(a) Peripalsmic Prokaryotic



(b) Cytoplasmic Eukaryotic

Figure 4.4 ROC Curve of classifiers for "Peripalsmic Prokaryotic" and "Cytoplasmic Eukaryotic" Protein Sequences

protein sequences, 0.8494, 0.9532, 0.7727, and 0.9027 respectively (shown in figure 4.5(b)).

Similarly to the second data set, the results shown in the table 4.3 and figure 4.5 indicate that RNBL generally outperforms or at least shows comparable performance over other algorithms we have compared on protein sequence classification.

4.5.3 UC Irvine Benchmark Data Sets

For the benchmark data sets, we chose 35 data sets from UCI data repository[4] (Blake and Merz, 1998). The data sets in the repository are widely used to evaluate machine learning algorithms.

Table 4.4 shows the accuracy of Naive Bayes Classifier (NBC), C4.5 decision tree, support vector machines (SVM), and recursive Naive Bayes Classifier (RNBC) regularized with conditional minimum description length (CMDL) and area under the ROC curve (AUC) respectively on UC-Irvine benchmark data sets, calculated by 10-fold cross validation with 95% confidence interval.

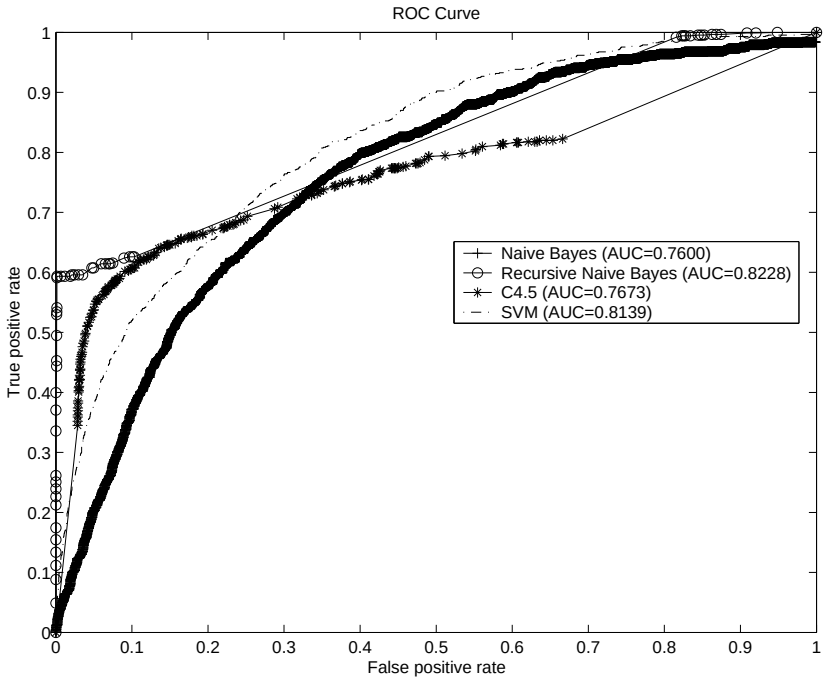
From the table 4.4, it is not easy to find one superior learning algorithm that dominates the other algorithms for most data sets. Overall, there is no one superior algorithm that dominates others. However, as for the comparison between NBC and RNBC, RNBC are more or equally accurate than NBC for 26 out of 35 data sets. Thus, we can see that RNBC mostly yields higher accuracy over NBC.

4.6 Related Work and Summary

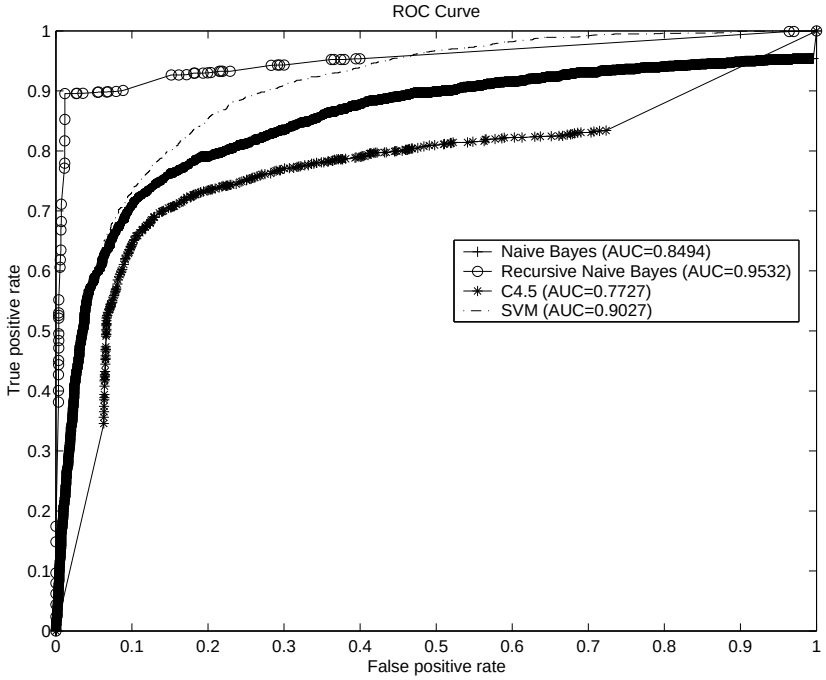
4.6.1 Related Work

As noted earlier, Langley (Langley, 1993) investigated recursive Bayesian classifiers for the instances described by tuples of nominal attribute values. RNBL reported in this study applies to not only the data of such kind, but also text/sequence data with multivariate/multinomial event models.

[4]This collection is publicly available at http://www.ics.uci.edu/~mlearn/MLRepository.html.



(a) Extracellular



(b) Nuclear

Figure 4.5 ROC Curve of classifiers for "Extracellular" and "Nuclear" Protein Sequences

There have been research work on relaxing the independence assumption of a Naive Bayes learning algorithm. Kohavi (Kohavi, 1996) introduced NBTree algorithm, a hybrid of a decision tree and Naive Bayes classifiers for instances represented using tuples of nominal attributes. NBTree evaluates the attributes available at each node to decide whether to continue building a decision tree or to terminate with a Naive Bayes classifier. In contrast, RNBL algorithm, like Langley's RBC, builds a decision tree, whose nodes are all Naive Bayes Classifiers.

Webb et al. (Webb et al., 2005) proposed an approach to improve the accuracy of Naive Bayes by weakening its attribute independence assumption by averaging all of a constrained class of classifiers. Langseth and Nielsen (Langseth and Nielsen, 2006) focus on a relatively new set of models, termed Hierarchical Naive Bayes models. Hierarchical Naive Bayes models extend the modeling flexibility of Naive Bayes models by introducing latent variables to relax some of the independence statements in these models. Liu et al. (Liu et al., 2005b) propose an algorithm named Graph-NB, which upgrades Naive Bayesian classifier to deal with multiple tables directly. In order to take advantage of linkage relationships among tables, and treat different tables linked to the target table differently, a semantic relationship graph is developed to describe the relationship and to avoid unnecessary joins.

Gama and Brazdil (Gama and Brazdil, 2000) proposed an algorithm that generates a cascade of classifiers. Their algorithm combines Naive Bayes, C4.5 decision tree and linear discriminants, and introduces a new attribute at each stage of the cascade. Gama (Gama, 2001) proposed an algorithm for multivariate tree learning that combines a univariate decision tree with a discriminant function by means of constructive induction. The algorithm uses Linear Bayes classifier for constructing new attributes. For growing trees, the algorithm builds multivariate decision nodes, and for pruning, the algorithm builds functional decision nodes. In both approaches, they performed experiments on several UC-Irvine benchmark data sets (Blake and Merz, 1998) for classifying instances represented as tuples of nominal attribute values. RNBL also recursively applies only the Naive Bayes classifier based on the multivariate/multinomial event models for text and sequences.

Area under the curve (AUC) has been used for the evaluation of prediction ability of the

learning algorithms. Huang and Ling (Huang and Ling, 2005) established formal criteria for comparing AUC and accuracy for learning algorithms and showed theoretically and empirically that AUC is a better measure (defined precisely) than accuracy. Brefeld and Scheffer (Brefeld and Scheffer, 2005) presented a rigorous derivation of an AUC maximizing Support Vector Machine (SVM). In our research, we maximize AUC for recursive Naive Bayes learner.

4.6.2 Summary

RNBL algorithm described in this study relaxes the *single generative model per class* assumption of NB classifiers, while maintaining some of their computational advantages. RNBL constructs a tree of Naive Bayes classifiers for UC Irvine benchmark data, text documents and biological sequences. It works in a manner similar to Langley's RBC (Langley, 1993), recursively partitioning the training set of labeled sequences at each node in the tree until a stopping criterion is satisfied. RNBL employs both the conditional minimum description length (CMDL) score (Friedman et al., 1997) and AUC score for the classifier. The CMDL score for RNBL is specifically adapted to the case of RNBL classifier based on the CMDL score for the Naive Bayes classifier using the multivariate (Zhang and Honavar, 2004) and multinomial (Kang et al., 2005d) event model as its stopping criterion. Previous reports by Langley (Langley, 1993) in the case of a recursive NB classifier (RBC) on selected UC-Irvine benchmark data sets whose instances were represented by tuples of nominal attribute values (such as the UC-Irvine benchmark data) had suggested that the tree of NB classifiers offered little improvement in accuracy over the standard NB classifier. In contrast, we observe that on protein sequence and text classification tasks, RNBL substantially outperforms the NB classifier. Furthermore, our experiments show that RNBL outperforms C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion) and yields accuracies that are comparable to those of SVM using similar information. As for UC-Irvine benchmark data, the classifiers from RNBL still mostly outperforms Naive Bayes classifiers, as shown in table 4.4.

Given the relatively modest computational requirements of RNBL relative to SVM, RNBL

is an attractive alternative to SVM in training classifiers on extremely large data sets of protein sequences or text documents, and works favorably over Naive Bayes learner on multivariate data sets such as UC-Irvine benchmark data.

4.6.3 Future Work

Some directions for future work include the following:

- Exploration of the use of abstraction hierarchies (discussed in chapter 2) over letters of the alphabet (or words of the vocabulary) (Kang et al., 2005d; Zhang and Honavar, 2004) with RNBL
- Investigation of alternative model selection measures as split stopping criteria for RNBL-MN rather than conditional minimum description length (CMDL) and Area Under the ROC Curve (AUC)
- Formal analysis and comparison of information gains obtained by selecting one attribute and obtained by choosing the predicted label of Naive Bayes classifier over the attributes
- Experimental analysis of the independence among the attributes affects the accuracy of decision tree induction algorithms and RNBL, using artificially created data sets

Table 4.2 Localization prediction results of RNBL and other learning algorithms on Prokaryotic and Eukaryotic protein sequences, calculated by 10-fold cross validation with 95% confidence interval.

(a) Prokaryotic protein sequences

Algorithm	Measure	Cytoplasmic	Extracellular	Peripalsmic
NBL-MN	accuracy	88.26±2.00	93.58±1.52	81.85±2.39
	specificity	89.60±1.89	65.93±2.94	53.85±3.09
	sensitivity	93.90±1.49	**83.18±2.32**	**72.77±2.76**
RNBL (CMDL)	accuracy	**90.67±1.81**	**94.58±1.41**	87.76±2.03
	specificity	91.61±1.72	75.73±2.66	73.53±2.74
	sensitivity	95.20±1.33	72.90±2.76	61.88±3.01
RNBL (AUC)	accuracy	**90.67±1.81**	93.78±1.50	**87.96±2.02**
	specificity	**91.73±1.71**	69.91±2.85	**73.84±2.73**
	sensitivity	95.06±1.35	73.83±2.73	62.87±3.00
C4.5	accuracy	84.15±2.27	91.98±1.69	84.65±2.24
	specificity	88.58±1.97	63.37±2.99	64.00±2.98
	sensitivity	88.32±1.99	59.81±3.04	55.45±3.09
SVM	accuracy	87.26±2.07	93.78±1.50	79.74±2.49
	specificity	84.67±2.24	**89.47±1.91**	50.00±3.10
	sensitivity	**99.56±0.41**	47.66±3.1	0.50±0.44

(b) Eukaryotic protein sequences

Algorithm	Measure	Cytoplasmic	Extracellular	Mitochondrial	Nuclear
NBL-MN	accuracy	71.41±1.80	83.11±1.49	71.69±1.79	80.72±1.57
	specificity	49.55±1.99	40.23±1.95	25.86±1.74	82.06±1.53
	sensitivity	**81.29±1.55**	53.85±1.98	**61.06±1.94**	73.38±1.76
RNBL (CMDL)	accuracy	78.12±1.64	**92.13±1.07**	**87.72±1.31**	**83.48±1.48**
	specificity	60.24±1.95	75.97±1.70	**54.44±1.98**	84.30±1.45
	sensitivity	65.79±1.89	**60.31±1.95**	43.93±1.97	**78.09±1.65**
RNBL (AUC)	accuracy	77.13±1.67	90.73±1.15	86.53±1.36	83.40±1.48
	specificity	58.57±1.96	67.12±1.87	48.93±1.99	84.29±1.45
	sensitivity	64.47±1.90	**60.31±1.95**	42.68±1.97	77.76±1.65
C4.5	accuracy	**78.99±1.62**	91.18±1.13	86.57±1.36	79.85±1.60
	specificity	63.51±1.92	69.89±1.83	49.03±1.99	77.94±1.65
	sensitivity	59.80±1.95	60.00±1.95	39.25±1.94	77.30±1.67
SVM	accuracy	71.98±1.79	86.69±1.35	86.77±1.35	79.36±1.61
	specificity	**83.33±1.48**	**100.00±0.00**	N/A	**87.53±1.31**
	sensitivity	0.73±0.34	0.62±0.31	0.00±0.00	63.35±1.92

Table 4.3 Localization prediction results on 7589 Eukaryotic protein sequences, calculated by 10-fold cross validation with 95% confidence interval.

Algorithm	Measure	Chloroplast	Cytoplasmic	Cytoskeleton	ER
NBL-MN	accuracy	78.11±0.93	77.58±0.94	95.08±0.49	91.44±0.63
	specificity	17.11±0.85	38.62±1.10	6.30±0.55	9.65±0.66
	sensitivity	**38.30±1.09**	**62.61±1.09**	60.00±1.10	**56.14±1.12**
Recursive	accuracy	90.71±0.65	**84.64±0.81**	99.50±0.16	**98.56±0.27**
NBL-MN	specificity	**44.59±1.12**	**53.79±1.12**	51.92±1.12	**53.42±1.12**
(CMDL)	sensitivity	20.27±0.91	44.00±1.12	**67.50±1.05**	34.21±1.07
Recursive	accuracy	90.26±0.67	83.55±0.83	97.95±0.32	97.99±0.32
NBL-MN	specificity	39.76±1.10	49.71±1.13	15.98±0.82	33.62±1.06
(AUC)	sensitivity	19.37±0.89	41.10±1.11	**67.50±1.05**	34.21±1.07
C4.5	accuracy	90.12±0.67	83.39±0.84	**99.56±0.15**	98.44±0.28
	specificity	42.86±1.11	49.15±1.13	**66.67±1.06**	45.45±1.12
	sensitivity	34.87±1.07	41.90±1.11	35.00±1.07	17.54±0.86
SVM	accuracy	**91.15±0.64**	83.63±0.83	99.47±0.16	98.50±0.27
	specificity	N/A	N/A	N/A	N/A
	sensitivity	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

Algorithm	Measure	Extracellular	Golgi	Lysosomal	Mitochondrial
NBL-MN	accuracy	81.59±0.87	94.29±0.52	90.88±0.65	78.28±0.93
	specificity	30.31±1.03	2.70±0.36	7.53±0.59	15.43±0.81
	sensitivity	47.74±1.12	**23.40±0.95**	**56.99±1.11**	**28.20±1.01**
Recursive	accuracy	**91.36±0.63**	99.18±0.20	98.54±0.27	89.09±0.70
NBL-MN	specificity	**71.73±1.01**	**25.81±0.99**	**40.63±1.11**	**31.48±1.05**
(CMDL)	sensitivity	39.49±1.10	17.02±0.85	41.94±1.11	11.69±0.72
Recursive	accuracy	90.26±0.67	98.88±0.24	97.94±0.32	88.82±0.71
NBL-MN	specificity	62.84±1.09	16.07±0.83	27.97±1.01	30.13±1.03
(AUC)	sensitivity	34.96±1.07	19.15±0.89	43.01±1.11	12.52±0.75
C4.5	accuracy	90.46±0.66	99.37±0.18	98.59±0.27	87.83±0.74
	specificity	58.98±1.11	0.00±0.00	36.54±1.08	31.29±1.04
	sensitivity	**52.61±1.12**	0.00±0.00	20.43±0.91	22.42±0.94
SVM	accuracy	88.64±0.71	**99.38±0.18**	**98.77±0.25**	**90.41±0.66**
	specificity	N/A	N/A	N/A	N/A
	sensitivity	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

Algorithm	Measure	Nuclear	Peroxisomal	Plasma	Vacuolar
NBL-MN	accuracy	85.88±0.78	92.15±0.61	92.43±0.60	94.92±0.49
	specificity	70.70±1.02	4.46±0.46	78.86±0.92	6.56±0.56
	sensitivity	**76.19±0.96**	**18.40±0.87**	**89.78±0.68**	**46.30±1.12**
Recursive	accuracy	**86.74±0.76**	98.03±0.31	**93.88±0.54**	98.92±0.23
NBL-MN	specificity	74.91±0.98	12.50±0.74	85.50±0.79	**20.83±0.91**
(CMDL)	sensitivity	72.15±1.01	3.20±0.40	87.04±0.76	18.52±0.87
Recursive	accuracy	86.00±0.78	97.59±0.35	93.76±0.54	98.47±0.28
NBL-MN	specificity	73.33±1.00	7.35±0.59	85.26±0.80	11.25±0.71
(AUC)	sensitivity	70.86±1.02	4.00±0.44	86.74±0.76	16.67±0.84
C4.5	accuracy	83.45±0.84	98.26±0.29	90.74±0.65	**99.29±0.19**
	specificity	68.85±1.04	**34.78±1.07**	80.11±0.90	N/A
	sensitivity	64.08±1.08	6.40±0.55	77.24±0.94	0.00±0.00
SVM	accuracy	82.32±0.86	**98.35±0.29**	93.47±0.56	**99.29±0.19**
	specificity	**83.87±0.83**	N/A	**95.66±0.46**	N/A
	sensitivity	37.94±1.09	0.00±0.00	73.78±	0.00±0.00

Table 4.4 Accuracy of Naive Bayes Classifier (NBC), Recursive Naive Bayes Classifier (RNBC) regularized with conditional minimum description length (CMDL) and area under the ROC curve (AUC), C4.5 decision tree, and support vector machines (SVM) respectively on UC-Irvine benchmark data sets, calculated by 10-fold cross validation with 95% confidence interval.

Data	NBC	RNBC (CMDL)	RNBC (AUC)	C4.5	SVM
Anneal	86.30±2.25	98.00±0.92	98.00±0.92	**98.44±0.81**	97.44±1.03
Audiology	73.45±5.76	73.45±5.76	73.45±5.76	77.88±5.41	**81.86±5.02**
Autos	56.10±6.79	72.20±6.13	72.20±6.13	**81.95±5.26**	71.22±6.20
Balance-scale	**90.40±2.31**	**90.40±2.31**	**90.40±2.31**	76.64±3.32	87.68±2.58
Breast-cancer	71.68±5.22	69.93±5.31	69.93±5.31	**75.52±4.98**	69.58±5.33
Breast-w	95.99±1.45	95.28±1.57	95.28±1.57	94.56±1.68	**97.00±1.27**
Colic	77.99±4.23	82.34±3.90	82.34±3.90	**85.33±3.62**	82.61±3.87
Credit-a	77.68±3.11	81.45±2.90	81.59±2.89	**86.09±2.58**	84.93±2.67
Credit-g	**75.40±2.67**	74.60±2.70	74.70±2.69	70.50±2.83	75.10±2.68
Dermatology	**97.81±1.50**	**97.81±1.50**	**97.81±1.50**	93.99±2.44	94.81±2.27
Diabetes	76.30±3.01	74.35±3.09	73.96±3.10	73.83±3.11	**77.34±2.96**
Glass	48.60±6.70	**66.82±6.31**	**66.82±6.31**	**66.82±6.31**	56.07±6.65
Heart-c	83.50±4.18	80.86±4.43	81.19±4.40	77.56±4.70	**84.16±4.11**
Heart-h	**83.67±4.23**	82.99±4.29	82.99±4.29	80.95±4.49	82.65±4.33
Heart-statlog	83.70±4.41	82.59±4.52	82.59±4.52	76.67±5.05	**84.07±4.36**
Hepatitis	84.52±5.70	84.52±5.70	84.52±5.70	83.87±5.79	**85.16±5.60**
Hypothyroid	95.28±0.68	98.17±0.43	98.17±0.43	**99.58±0.21**	93.61±0.78
Ionosphere	82.62±3.96	**93.45±2.59**	**93.45±2.59**	91.45±2.92	88.60±3.51
Iris	**96.00±3.14**	92.67±4.17	92.67±4.17	**96.00±3.14**	**96.00±3.14**
Kr-vs-kp	87.89±1.13	92.37±0.92	92.30±0.92	**99.44±0.26**	95.43±0.72
Labor	**89.47±7.97**	**89.47±7.97**	**89.47±7.97**	73.68±11.43	**89.47±7.97**
Letter	64.12±0.66	**89.53±0.42**	85.41±0.49	87.98±0.45	82.34±0.53
Lymph	83.11±6.04	79.05±6.56	78.38±6.63	77.03±6.78	**86.49±5.51**
Mushroom	95.83±0.43	99.94±0.05	99.94±0.05	**100.00±0.00**	**100.00±0.00**
Primary-tumor	**50.15±5.32**	**50.15±5.32**	**50.15±5.32**	39.82±5.21	46.90±5.31
Segment	80.22±1.62	95.84±0.81	92.77±1.06	**96.93±0.70**	93.07±1.04
Sick	92.60±0.84	97.38±0.51	97.24±0.52	**98.81±0.35**	93.85±0.77
Sonar	67.79±6.35	**79.33±5.50**	**79.33±5.50**	71.15±6.16	75.96±5.81
Soybean	92.97±1.92	92.97±1.92	92.97±1.92	91.51±2.09	**93.85±1.80**
Splice	95.30±0.73	**95.55±0.72**	**95.55±0.72**	94.08±0.82	93.45±0.86
Vehicle	44.80±3.35	71.75±3.03	71.51±3.04	72.46±3.01	**74.35±2.94**
Vote	90.11±2.81	96.09±1.82	**96.32±1.77**	**96.32±1.77**	96.09±1.82
Vowel	63.74±2.99	**87.37±2.07**	85.96±2.16	81.52±2.42	71.41±2.81
Waveform-5000	80.00±1.11	83.16±1.04	83.12±1.04	75.08±1.20	**86.68±0.94**
Zoo	95.05±4.23	**96.04±3.80**	**96.04±3.80**	92.08±5.27	**96.04±3.80**
# of wins	7	11	10	13	**14**

CHAPTER 5. SUMMARY AND DISCUSSION

5.1 Summary

An important goal of inductive learning is to generate accurate and compact classifiers from data. In a typical inductive learning scenario, instances in a data set are simply represented as ordered tuples of attribute values. There are many possibilities for the improvement of the machine learning algorithm. In our research, we explore three methods (abstraction, aggregation, and recursion) to improve the accuracy and compactness of the classifiers.

- Abstraction method in our research is aimed at the design and analysis of algorithms that generate and deal with taxonomies for construction of compact and robust classifiers. We introduce algorithms for automated construction of taxonomies inductively from both structured (such as UCI Repository) and unstructured (such as text and biological sequences) data. We invented AVT-Learner, an algorithm for automated construction of attribute value taxonomies (AVT) from data, and Word Taxonomy Learner (WTL) for automated construction of word taxonomy from text and sequence data. The experimental results show that the AVTs generated by AVT-Learner are competitive with human-generated AVTs (in cases where such AVTs are available). AVT-NBL using AVTs generated by AVT-Learner achieves classification accuracies that are comparable to or higher than those obtained by NBL; and the resulting classifiers are significantly more compact than those generated by NBL. Similarly, our experimental results of WTL and WTNBL on protein localization sequences and Reuters text show that the proposed algorithms can generate Naive Bayes classifiers that are more compact and often more accurate than those produced by standard Naive Bayes learner for the Multinomial

Model.

- We apply aggregation method to construct features as a multiset of values for intrusion detection task. More precisely, we propose a bag of system calls representation for system call traces for the intrusion detection task and describe misuse and anomaly detection results with standard machine learning techniques on University of New Mexico (UNM) and MIT Lincoln Lab (MIT LL) system call sequences with the proposed representation. With the feature representation as input, we compare the performance of several machine learning techniques for misuse detection and show experimental results on anomaly detection. The results show that standard machine learning and clustering techniques on simple bag of system calls representation of system call sequences in the operating system's kernel is effective and often performs better than those approaches that use foreign contiguous sequences in detecting intrusive behaviors of compromised processes.

- We describe recursive Naive Bayes learner (RNBL), which relaxes this assumption by constructing a tree of Naive Bayes classifiers for sequence classification, where each individual NB classifier in the tree is based on an event model (one model for each class at each node in the tree). In our experiments on protein sequences, Reuters newswire documents and UC-Irvine benchmark data sets, we observe that RNBL substantially outperforms NB classifier. Furthermore, our experiments on the protein sequences and the text documents show that RNBL outperforms C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion) and yields accuracies that are comparable to those of support vector machines (SVM) using similar information.

5.2 Contributions

The main contributions of this dissertation include:

1. AVT-Learner and WTL is effective in generating taxonomies that when used by AVT-NBL and WTNBL-MN, a principled extension of the standard algorithm for learning Naive Bayes classifiers, result in classifiers that are substantially more compact (and

often more accurate) than those obtained by the standard Naive Bayes Learner (that does not use taxonomies).

2. The taxonomies generated by AVT-Learner and WTL are competitive with human supplied taxonomies (in the case of benchmark data sets where human-generated taxonomies were available) in terms of both the error rate and size of the resulting classifiers.

3. From the experiments on UNM and MIT Lincoln Lab data sets for the evaluation of intrusion detection systems, we show that a simple approach like a bag of system calls may be more adequate than k-gram in many scenarios. The results of experiments described in this paper show that it is possible to achieve nearly perfect detection rates and false positive rates using a data representation that discards the relationship between system call and originating process as well as the sequence structure of the calls within the traces.

4. Contrary to the previous report by Langley (Langley, 1993) in the case of a recursive NB classifier (RBC) on selected UC-Irvine benchmark data sets whose instances were represented by tuples of nominal attribute values (such as the UC-Irvine benchmark data), we observe that on protein sequence and text classification tasks, RNBL substantially outperforms the NB classifier, and our experiments show that RNBL outperforms C4.5 decision tree learner (using tests on sequence composition statistics as the splitting criterion) and yields accuracies that are comparable to those of SVM using similar information. As for UC-Irvine benchmark data, the classifiers from RNBL still mostly outperforms Naive Bayes classifiers, as shown in table 4.4.

5.3 Future Work

Some promising directions of the future work include:

1. **Learning Taxonomies**

(a) Extending AVT-Learner described in this research to learn AVTs that correspond to tangled hierarchies, which can be represented by directed acyclic graphs (DAG) instead of trees, or complicated graph structure

(b) Learning AVT from data for a broad range of real world applications such as census data analysis, learning classifiers from relational data (Atramentov et al., 2003), and protein function classification (Wang and Stolfo, 2003), identification of protein-protein interfaces (Terribilini et al., 2006; Yan et al., 2003)

(c) Developing algorithms for learning hierarchical ontologies based on part-whole and other relations as opposed to ISA relations captured by an AVT

(d) Developing algorithms for learning hierarchical groupings of values associated with more than one attribute

2. **Intrusion Detection Using a Bag of System Calls**

(a) Further formalizing intrusive behaviors of multiple processes as a *multi-bag* in IDS framework. A process often fork child processes during the execution and intrusion can be a cooperative work between the processes in the same group. Thus, it is a more natural idea than the conventional approaches to model an IDS to consider cooperative attacks. However, the research area of a host-based IDS that monitors multiple process' cooperative behavior has not been explored. Considering this observation, we will propose a theoretical framework for IDS that models multiple processes as a multi-bag.

(b) Extending the supervised anomaly detection experiments with one-class support vector machines (Scholkopf et al., 2001; Leslie et al., 2002a,b; Tian et al., 2004) or other anomaly detection techniques in SVM (Tax and Duin, 2004)

(c) Extending feature representation so that subsequences rather than system calls can be dealt with by the existing machine learning techniques in an efficient way. We believe it will show better performance in terms of accuracy/detection rate/false positive rate in supervised anomaly detection

(d) Modeling multiple processes' behavior in one trace. Current intrusion detection system assumes one process produce intrusions in the intrusion model. Modeling multiple processes that are cooperative is more probable for future intrusion detection system

(e) Performing experiments on different operating system such as Microsoft Windows. Anomaly detection of spyware in Windows is a good example

(f) Applying generalized global alignment (Huang and Chao, 2003; Takeda, 2005; Coull et al., 2003) of system call sequences

(g) Using system call arguments (Mutz et al., 2006) with abstraction (Kang et al., 2004) for anomaly detection

3. **Recursive Naive Bayes Learner**

(a) Exploration of the use of abstraction hierarchies (discussed in chapter 2) over letters of the alphabet (or words of the vocabulary) (Kang et al., 2005d; Zhang and Honavar, 2004) with RNBL

(b) Investigation of alternative model selection measures as split stopping criteria for RNBL-MN rather than conditional minimum description length (CMDL) and Area Under the ROC Curve (AUC)

(c) Formal analysis and comparison of information gains obtained by selecting one attribute and obtained by choosing the predicted label of Naive Bayes classifier over the attributes

(d) Experimental analysis of the independence among the attributes affects the accuracy of decision tree induction algorithms and RNBL, using artificially created data sets

BIBLIOGRAPHY

Akyildizy, I. F., Vuran, M. C., Akan, O. B., and Su, W. (2005). Wireless sensor networks: A survey revisited. *Computer Networks Journal*.

Anderson, D., Lunt, T. F., Javitz, H., Tamaru, A., and Valdes, A. (1995). Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system (NIDES). Technical Report SRI-CSL-95-06, Computer Science Laboratory, SRI International, Menlo Park, CA.

Andorf, C., Dobbs, D., and Honavar, V. (2006). Learning classifiers for assigning protein sequences to subcellular localization families. In *Proceedings of the Annual Meeting of the International Society for Computational Biology (ISMB 2006)*, Fortaleza, Brazil.

Andorf, C., Silvescu, A., Dobbs, D., and Honavar, V. (2004). Learning classifiers for assigning protein sequences to gene ontology functional families. In *Proceedings of the Fifth International Conference on Knowledge Based Computer Systems (KBCS 2004)*, pages 256–265.

Apté, C., Damerau, F., and Weiss, S. M. (1994). Towards language independent automated learning of text categorization models. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval*, pages 23–30, New York, NY, USA. Springer-Verlag New York, Inc.

Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., Harris, M., Hill, D., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J., Richardson, J., Ringwald, M., Rubin, G., and Sherlock, G. (2000). Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, 25(1):25–29.

Atramentov, A., Leiva, H., and Honavar, V. (2003). A multi-relational decision tree learning algorithm - implementation and experiments. In Horváth, T. and Yamamoto, A., editors, *ILP03*, volume 2835 of *LNAI*, pages 38–56. Springer-Verlag.

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers Univ.

Bairoch, A. and Apweiler, R. (2000). The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.*, 28:45–48.

Baker, L. D. and McCallum, A. K. (1998). Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM Press.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*.

Bishop, C. M. (1996). *Neural networks for pattern recognition*. Oxford University Press.

Blake, C. and Merz, C. (1998). UCI repository of machine learning databases.

Brefeld, U. and Scheffer, T. (2005). AUC maximizing support vector learning. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*.

Cai, Y.-D., Liu, X.-J., and Chou, K.-C. (2002). Artificial neural network model for predicting protein subcellular location. *Computers & Chemistry*, 26(2):179–182.

Campos, M. and Milenova, B. (2005). Creation and deployment of data mining-based intrusion detection systems in oracle database l0g. In *Proceedings of the Fourth International Conference on Machine Learning and Applications*.

Carvalho, V. R. and Cohen, W. W. (2005). On the collective classification of email "speech acts". In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05)*, pages 345–352, New York, NY, USA. ACM Press.

Cessie, S. L. and Houwelingen, J. V. (1992). Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201.

Cha, B., Vaidya, B., and Han, S. (2005). Anomaly intrusion detection for system call using the soundex algorithm and neural networks. In *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)*.

Chebrolua, S., Abrahama, A., and Thomas, J. P. (2005). Feature deduction and ensemble design of intrusion detection systems. *Computers and Security*.

Cohen, W. W. (1995). Fast effective rule induction. In Prieditis, A. and Russell, S., editors, *Proc. of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA. Morgan Kaufmann.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

Coull, S., Branch, J., Szymanski, B., and Breimer, E. (2003). Intrusion detection: A bioinformatics approach. In *19th Annual Computer Security Applications Conference*, Las Vegas, Nevada.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232.

Dhar, V. and Tuzhilin, A. (1993). Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):926–938.

Dimitropoulos, X., Krioukov, D., Riley, G., and Claffy, K. (2006). Revealing the autonomous system taxonomy: The machine learning approach. In *Passive and Active Measurement (PAM) Workshop*.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh in-*

ternational conference on Information and knowledge management, pages 148–155. ACM Press.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Data Mining for Security Applications*.

Fawcett, T. (2003). ROC graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Labs.

Fawcett, T. (2006). ROC graphs with instance-varying costs. *Pattern Recognition Letters*, 27(8):882 – 891.

Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. (1996). A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128. IEEE Computer Society.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163.

Gabrilovich, E. and Markovitch, S. (2005). Feature generation for text categorization using world knowledge. In *Proceedings of The Nineteenth International Joint Conference for Artificial Intelligence*, pages 1048–1053, Edinburgh, Scotland.

Gama, J. (2001). Functional trees for classification. In *IEEE International Conference on Data Mining*. IEEE Computer Society.

Gama, J. and Brazdil, P. (2000). Cascade generalization. *Machine Learning*, 41(3):315–343.

Ganti, V., Gehrke, J., and Ramakrishnan, R. (1999). CACTUS - clustering categorical data using summaries. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 73–83. ACM Press.

Gao, H.-H., Yang, H.-H., and Wang, X.-Y. (2005). Principal component neural networks based intrusion feature extraction and detection using SVM. In *Proceedings of the first*

International Conference on Advances in Natural Computation (ICNC 2005), pages 21–27, Changsha, China. Springer Berlin / Heidelberg.

Ghosh, A. and Schwartzbard, A. (1999). A study in using neural networks for anomaly and misuse detection. In *8th USENIX Security Symposium*, pages 141–151, Washington, D.C.

Gibson, E. (1988). Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge. *Annual Review of Psychology*, 39:1–41.

Gunes Kayacik, Nur Zincir-Heywood, M. H. (2003). On the capability of an SOM based intrusion detection system. In *The IEEE International Joint Conference on Neural Networks, IJCNN03*.

Han, J. and Fu, Y. (1996). Exploration of the power of attribute-oriented induction in data mining. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*. AIII Press/MIT Press.

Hand, D. and Till, R. (2001). A simple generalization of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial intelligence*, 36:177–221.

Heller, K. A., Svore, K. M., Keromytis, A. D., and Stolfo, S. J. (2003). One class support vector machines for detecting anomalous window registry accesses. In *The 3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security*, Florida.

Helmer, G., Wong, J., Slagell, M., Honavar, V., Miller, L., and Lutz, R. (2001). A software fault tree approach to requirement analysis of an intrusion detection system. In *Symposium on Requirements Engineering for Information Security*.

Hendler, J., Stoffel, K., and Taylor, M. (1996). Advances in high performance knowledge representation. Technical Report CS-TR-3672, University of Maryland Institute for Advanced Computer Studies Dept. of Computer Science.

Hofmeyr, S. A., Forrest, S., and Somayaji, A. (1998). Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6(3):151–180.

Huang, J. and Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299 – 310.

Huang, X. and Chao, K.-M. (2003). A generalized global alignment algorithm. *Bioinformatics*, 19(2):228–233.

Jiang, S., Song, X., Wang, H., Han, J.-J., and Li, Q.-H. (2006). A clustering-based method for unsupervised intrusion detections. *Pattern Recognition Letters*, 22(7):802 – 810.

Jiang, W., Xu, Y., and Xu, Y. (2005). A novel intrusions detection method based on HMM embedded neural network. In *Proceedings of the first International Conference on Advances in Natural Computation (ICNC 2005)*, Changsha, China.

Jiang, X. and Xu, D. (2005). Behavioral footprinting: a new dimension to characterize self-propagating worms. Technical Report CERIAS TR 2005-80, CERIAS and Department of Computer Science, Purdue University.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In Nédellec, C. and Rouveirol, C., editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine Learning*, pages 377–384, Bonn, Germany. ACM Press.

Kang, B.-D., Lee, J.-W., Kim, J.-H., Kwon, O.-H., Seong, C.-Y., and Kim, S.-K. (2005a). An intrusion detection system using principal component analysis and time delay neural network. In *Proceedings of 7th International Workshop on Enterprise networking and Computing in Healthcare Industry (HEALTHCOM 2005)*, pages 442– 445.

Kang, D.-K., Fuller, D., and Honavar, V. (2005b). Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In *Proceedings of 6th IEEE Systems Man and Cybernetics Information Assurance Workshop (IAW)*, West Point, NY, USA.

Kang, D.-K., Fuller, D., and Honavar, V. (2005c). Learning classifiers for misuse detection using a bag of system calls representation. In *Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI-2005)*, volume 3495, pages 511–516, Atlanta, GA, USA. Springer-Verlag.

Kang, D.-K., Silvescu, A., and Honavar, V. (2006). RNBL-MN: A recursive naive Bayes learner for sequence classification. In *10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006)*, volume 3918 of *Lecture Notes in Artificial Intelligence*, Singapore. Springer Verlag.

Kang, D.-K., Silvescu, A., Zhang, J., and Honavar, V. (2004). Generation of attribute value taxonomies from data for data-driven construction of accurate and compact classifiers. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 130–137.

Kang, D.-K., Zhang, J., Silvescu, A., and Honavar, V. (2005d). Multinomial event model based abstraction for sequence and text classification. In *Abstraction, Reformulation and Approximation, 6th International Symposium, SARA 2005, Edinburgh, Scotland, UK, July 26-29, 2005, Proceedings*, Lecture Notes in Computer Science, pages 134–148. Springer.

Keerthi, S. S. (2005). Generalized LARS as an effective feature selection tool for text classification with SVMs. In *Proceedings of the 22nd international conference on Machine Learning*, pages 417–424, Bonn, Germany. ACM Press.

Kohavi, R. (1996). Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207.

Kohavi, R. and Provost, F. (2001). Applications of data mining to electronic commerce. *Data Mining and Knowledge Discovery*, 5(1-2):5–10.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Statist.*, 22:79–86.

Langley, P. (1993). Induction of recursive Bayesian classifiers. In *ECML '93: Proceedings of the European Conference on Machine Learning*, pages 153–164, London, UK. Springer-Verlag.

Langley, P., Iba, W., and Thompson, K. (1992). An analysis of Bayesian classifiers. In *National Conference on Artificial Intelligence*, pages 223–228.

Langseth, H. and Nielsen, T. D. (2006). Classification using hierarchical naive Bayes models. *Machine Learning*, 63(2):135–159.

Lee, H., Song, J., and Park, D. (2005). Intrusion detection system based on multi-class SVM. In *Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing (RSFDGrC 2005)*, pages 511–519, Regina, Canada. Springer Berlin / Heidelberg.

Lee, W. and Stolfo, S. (1998). Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX.

Lee, W., Stolfo, S. J., and Mok, K. W. (1999). A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*, pages 120–132.

Leslie, C., Eskin, E., and Noble, W. S. (2002a). The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing 2002 (PSB 2002)*, pages 564–575.

Leslie, C., Eskin, E., Weston, J., and Noble, W. S. (2002b). Mismatch string kernels for SVM protein classification. In *Neural Information Processing Systems 2002 (NIPS 2002)*.

Li, M. and Vitanyi, P. M. B. (1993). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin.

Liao, Y. and Vemuri, V. R. (2002). Using text categorization techniques for intrusion detection. In *Proceedings of the 11th USENIX Security Symposium*, pages 51–59, Berkeley, CA, USA. USENIX Association.

Lincoff, G. H. (1981). *The Audubon Society Field Guide to North American Mushrooms*. Alfred A. Knopf, New York, NY.

Lippmann, R., Cunningham, R. K., Fried, D. J., Graf, I., Kendall, K. R., Webster, S. E., and Zissman, M. A. (1999). Results of the darpa 1998 offline intrusion detection evaluation. In *Recent Advances in Intrusion Detection*.

Liu, A., Martin, C., Hetherington, T., and Matzner, S. (2005a). A comparison of system call feature representations for insider threat detection. In *Proceedings of 6th IEEE Systems Man and Cybernetics Information Assurance Workshop (IAW)*, West Point, NY, USA.

Liu, H., Yin, X., and Han, J. (2005b). An efficient multi-relational naive Bayesian classifier based on semantic relationship graph. In *Proceedings of the 4th international workshop on Multi-relational mining (MRDM '05)*, pages 39–48, New York, NY, USA. ACM Press.

Lu, C.-T., Boedihardjo, A. P., and Manalwar, P. (2005). Exploiting efficient data mining techniques to enhance intrusion detection systems. In *IEEE International Conference on Information Reuse and Integration (IRI -2005)*, pages 512– 517.

Ma, Z., Zhen, L., and Liao, X. (2005). On the efficiency of support vector classifiers for intrusion detection. In *International Conference on Neural Networks and Brain (ICNN&B '05)*, pages 935–940.

Mason, S. J. and Graham, N. E. (2002). Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical signi cance and interpretation. *Q.J.R. Meteorol. Soc.*, 128:2145–2166.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Mukherjee, B., Heberlein, H., and Levitt, K. (1994). Network intrusion detection. *IEEE Network*, 8(3):26–41.

Murali, A. and Rao, M. (2005). A survey on intrusion detection approaches. In *First International Conference on Information and Communication Technologies (ICICT 2005)*, pages 233– 240.

Mutz, D., Valeur, F., Vigna, G., and Kruegel, C. (2006). Anomalous system call detection. *ACM Trans. Inf. Syst. Secur.*, 9(1):61–93.

Newsome, J., Karp, B., and Song, D. (2005). Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of IEEE Symposium on Security and Privacy (S&P'05)*, pages 226–241.

Olave, M., Rajkovic, V., and Bohanec, M. (1989). An application for admission in public school systems. In Snellen, I. T. M., van de Donk, W. B. H. J., and Baquiast, J.-P., editors, *Expert Systems in Public Administration*, pages 145–160. Elsevier Science Publishers, North Holland.

Park, K.-J. and Kanehisa, M. (2003). Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. *Bioinformatics*, 19(13):1656–1663.

Pazzani, M. and Kibler, D. (1992). The role of prior knowledge in inductive learning. *Machine Learning*, 9:54–97.

Pazzani, M. J., Mani, S., and Shankle, W. R. (1997). Beyond concise and colorful: Learning intelligible rules. In *Knowledge Discovery and Data Mining*, pages 235–238.

Peddabachigaria, S., Abrahamb, A., Grosanc, C., and Thomasa, J. (2005). Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer Applications*.

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of English words. In *31st Annual Meeting of the ACL*, pages 183–190.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, pages 185–208.

Provost, F. and Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distribution. In Heckerman, D., Mannila, H., and Pregibon, D., editors, *Proceedings of the 3rd Intl. Conf, on Knowledge Discovery and Data Mining (KDD-97)*, pages 43–48, Menlo Park, CA. AAAI Press.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Reinhardt, A. and Hubbard, T. (1998). Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26(9):2230–2236.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, 2(3):229–246.

Rooney, N., Patterson, D., Galushka, M., and Dobrynin, V. (2006). A scaleable document clustering approach for large document corpora. *Information Processing and Management*, 42(5):1163 – 1175.

Sandler, M. (2005). On the use of linear programming for unsupervised text classification. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 256 – 264, Chicago, Illinois. ACM Press.

Scholkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1472.

Shadbolt, N., Berners-Lee, T., Hendler, J., Hart, C., and Benjamins, R. (2006). The next wave of the web. In *Proceedings of the 15th international conference on World Wide Web*, Edinburgh, Scotland.

Slonim, N., Atwal, G. S., Tkaik, G., and Bialek, W. (2005). Information-based clustering. In *Proceedings of the National Academy of Sciences of United States of America*, Joseph Henry Laboratories of Physics, and Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, NJ 08544.

Slonim, N., Friedman, N., and Tishby, N. (2006). Multivariate information bottleneck. *Neural Computation*, 18(8):1739–1789.

Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197.

Spencer, J. (2005). Use of an artificial neural network to detect anomalies in wireless device location for the purpose of intrusion detection. In *Proceedings of IEEE SoutheastCon*.

Steinwart, I., Hush, D., and Scovel, C. (2005). A classification framework for anomaly detection. *The Journal of Machine Learning Research*, 5:211–232.

Sy, B. K. (2005). Signature-based approach for intrusion detection. In *Proceedings of the 4th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM 2005)*, Leipzig, Germany.

Takeda, K. (2005). The application of bioinformatics to network intrusion detection. In *39th Annual 2005 International Carnahan Conference on Security Technology (CCST '05)*, pages 130– 132.

Tan, K. M. C. and Maxion, R. A. (2002). "Why 6?" Defining the operational limits of STIDE, an anomaly-based intrusion detector. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 188. IEEE Computer Society.

Taneja, I. (1995). New developments in generalized information measures. *Advances in Imaging and Electron Physics*, 91:37–135.

Tang, Y. and Chen, S. (2005). Defending against internet worms: A signature-based approach. In *Proc. of IEEE INFOCOM 05*, Miami, Florida.

Tax, D. M. J. and Duin, R. P. W. (2004). Support vector data description. *Machine Learning*, 54(1):45–66.

Taylor, M., Stoffel, K., , and Hendler, J. (1997). Ontology based induction of high level classification rules. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*.

Terribilini, M., Lee, J.-H., Yan, C., Jernigan, R., Honavar, V., and Dobbs, D. (2006). Prediction of rna-binding sites in proteins based on amino acid sequence. *RNA*, 12:1450–1462.

Tian, S., Yu, J., and Yin, C. (2004). Anomaly detection using support vector machines. In *International Symposium on Neural Networks (ISNN 2004)*.

Topsϕe, F. (2000). Some inequalities for information divergence and related measures of discrimination. *IEEE Transactions on Information Theory*, 46:1602–1609.

Tripp, G. (2005). A parallel "string matching engine" for use in high speed network intrusion detection systems. *Journal in Computer Virology*, 2(1):21–34.

Undercoffer, J. L., Joshi, A., Finin, T., and Pinkston, J. (2004). A Target Centric Ontology for Intrusion Detection: Using DAML+OIL to Classify Intrusive Behaviors. *Knowledge Engineering Review*.

Wagner, D. and Soto, P. (2002). Mimicry attacks on host based intrusion detection systems. In *Proc. Ninth ACM Conference on Computer and Communications Security*.

Wang, K. and Stolfo, S. J. (2003). One class training for masquerade detection. In *ICDM Workshop on Data Mining for Computer Security (DMSEC 03)*, Melbourne, FL.

Warrender, C., Forrest, S., and Pearlmutter, B. A. (1999). Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145.

Webb, G. I., Boughton, J. R., and Wang, Z. (2005). Not so naive Bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24.

Xu, X. and Xie, T. (2005). A reinforcement learning approach for host-based intrusion detection using sequences of system calls. In *Proceedings of the International Conference on Intelligent Computing (ICIC 2005)*, pages 995–1003, Hefei, China.

Yamazaki, T., Pazzani, M. J., and Merz, C. J. (1995). Learning hierarchies from ambiguous natural language data. In *International Conference on Machine Learning*, pages 575–583.

Yan, C., Dobbs, D., and Honavar, V. (2003). Identification of surface residues involved in protein-protein interaction – a support vector machine approach. In Abraham, A., Franke, K., and Koppen, M., editors, *Intelligent Systems Design and Applications (ISDA-03)*, pages 53–62. Springer-Verlag.

Yan, C., Dobbs, D., and Honavar, V. (2004). A two-stage classifier for identification of protein-protein interface residues. In *Proceedings Twelfth International Conference on Intelligent Systems for Molecular Biology / Third European Conference on Computational Biology (ISMB/ECCB 2004)*, pages 371–378.

Yang, T. (2005). A time series data mining based on arma and hopfield model for intrusion detection. In *International Conference on Neural Networks and Brain (ICNN&B '05)*, pages 1045– 1049.

Yang, Z., Wei, X., Bi, L., Shi, D., and Li, H. (2005). An intrusion detection system based on RBF neural network. In *Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design*, pages 873– 875.

Yilmazel, O., Symonenko, S., Balasubramanian, N., and Liddy, E. D. (2005). Leveraging one-class SVM and semantic analysis to detect anomalous content. In *Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI 2005)*, pages 381–388, Atlanta, GA. Springer.

Zaki, M. J., Peters, M., Assent, I., and Seidl, T. (2005). CLICKS: an effective algorithm for mining subspace clusters in categorical datasets. In *Proceeding of the eleventh ACM*

SIGKDD international conference on Knowledge discovery in data mining, pages 736–742, Chicago, Illinois, USA. ACM Press.

Zhang, D. and Lee, W. S. (2006). Extracting key-substring-group features for text classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '06)*, pages 474–483, New York, NY, USA. ACM Press.

Zhang, J. and Honavar, V. (2003). Learning decision tree classifiers from attribute value taxonomies and partially specified data. In *the Twentieth International Conference on Machine Learning (ICML 2003)*, Washington, DC.

Zhang, J. and Honavar, V. (2004). AVT-NBL: An algorithm for learning compact and accurate naive bayes classifiers from attribute value taxonomies and data. In *International Conference on Data Mining (ICDM 2004)*.

Zhang, J., Silvescu, A., and Honavar, V. (2002). Ontology-driven induction of decision trees at multiple levels of abstraction. In *Proceedings of Symposium on Abstraction, Reformulation, and Approximation 2002. Vol. 2371 of Lecture Notes in Artificial Intelligence : Springer-Verlag.*