An Investigation of Classification and Clustering of Sequential Data

Submitted By
Pradeep Kumar

For the Degree of

Doctor of Philosophy
in
Computer Science

Department of Computer & Information Sciences
School of Mathematics and Computer/Information Sciences
University of Hyderabad
Hyderabad - 500046
INDIA
August 2006

To my parents and teachers,

without whose support this work

would not be possible.

DECLARATION

I, **Pradeep Kumar**, hereby declare that the work presented in this thesis has been carried out by me under the supervision of **Dr. S. Bapi Raju**, Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, India and **Dr. P. Radha Krishna**, IDRBT, Hyderabad, India, as per the PhD ordinances of the University. I declare, to the best of my knowledge, that no part of this thesis has been submitted for the award of a research degree of any other University.

Pradeep Kumar

Regn No: 03MCPC11

Department of Computer and Information Sciences
School of Mathematics and Computer/Information Sciences
University of Hyderabad, Hyderabad, INDIA

C E R T I F I C A T E

This is to certify that the thesis work entitled **'An Investigation of Classification and Clustering of Sequential Data'** being submitted by **Mr. Pradeep Kumar** (Reg. No. **03MCPC11)** in partial fulfillment of the requirement for the award of degree of **Doctor of Philosophy (Computer Science)** of the University of Hyderabad, is a record of *bona fide* work carried out by him under our supervision.

The matter embodied in this thesis has not been submitted for the award of any other research degree.

Dr. S. Bapi Raju
(Supervisor)
Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, INDIA

Dr. P. Radha Krishna
(Supervisor)
Institute for Development and Research in Banking Technology, Hyderabad, INDIA

Prof. Arun Agarwal
(Head)
Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, INDIA

Prof. Arun K. Pujari
(Dean)
School of Mathematics and Computer/Information Sciences, University of Hyderabad, Hyderabad, INDIA

ACKNOWLEDGEMENTS

There are many people who supported me while I was working on my thesis and I am sorry that I cannot mention all of them in the following. I want to express my deep gratitude to all of them.

First of all, I would like to thank my supervisors, Dr. S. Bapi Raju and Dr. P. Radha Krishna. They made this work possible by offering me the opportunity to work on my own choice of problems in their excellent research guidance. I benefitted a lot from the opportunities they provided for me and enjoyed the inspiring working atmosphere they created.

I thank Prof. Arun Agarwal, Head, Department of Computer and Information Sciences (DCIS) for making available all the facilities required for this research work. I thank Shri. Arvind Sharma, Director, IDRBT, Hyderabad and Dr. V. P. Gulati, ex-Director, IDRBT, Hyderabad to provide me with all necessary infrastructure to carry out this research work smoothly at IDRBT.

I want to extend my warmest thanks to Prof. Arun. K. Pujari, a member of Doctoral research committee. He shared a lot of his knowledge about scientific work and data mining with me. His encouragement and cooperation during my research time helped me a lot in doubtful times.

My thanks are due to Dr. Atul Negi, second member of Doctoral research committee for patiently listening to my problems and providing me the fruitful way to come out of it.

I would like to pay my gratitude to Dr. S. K. De, XLRI, Jamshedpur, India and Mr. A. Laha, Faculty, IDRBT, Hyderabad, INDIA for their helpful discussions during my research. They have been very friendly and informative.

Most of the solutions in this thesis were developed in a team and I want to especially thank them. I know that working with me sometimes demands a lot of endurance and often the willingness to follow my rather broad excursions. I am trying to improve. My

special thanks are to M. Venketesawara Rao, V. Sandhya, P. Seema and P. Deepthi. Scientific research lives in discussions and therefore I want to thank all of my colleagues for many interesting conversations and arguments, not to mention the good times we had. The list goes long but to name some team members - M. L. Das, Sanjay Rawat, G. Geetha Kumari, T. M. Padmaja. They directly or indirectly provided the impetus for my research. If I have missed some name I am sorry but my heartiest thanks are due to them.

I would like to thank faculty members of IDRBT, Hyderabad, INDIA and University of Hyderabad, Hyderabad, INDIA for their useful discussion and advice throughout my research. I would also like to thank IDRBT for its fellowship, without which it would have been very difficult to pursue this research.

I want to thank my parents for their affection and their help for managing my life in busy times. Without you, it would have been very difficult to focus on my research. At last, I want to thank the rest of my family members and my friends. Their belief in me was a driving force behind my efforts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOL

Symbol	Description
L	Length of the sliding window
U	Universe
X	Set X, subset of U
R	Equivalence relation
τ	Tolerance relation
$\overline{R}X$	Upper approximation of set X
$\underline{R}X$	Lower approximation of set X
δ	Threshold Value in rough set algorithm
σ	Relative similarity
τ	Threshold Value in kNN algorithm
p	Sequence Similarity
q	Set Similarity
A	Approximation Space
k	Total number of clusters in clustering algorithm and number of nearest neighbors in classification algorithm
$S^3M(A,B)$	Similarity between sequences A and B using S^3MSequence Similarity
$\hat{C_j}$	Cluster representative of j^{th} cluster

$LD(A, B)$ Levensthein distance between sequences A and B

$|C_j|$ Size of the j^{th} cluster

C_{j_l} l^{th} object of j^{th} cluster

List of Algorithms

ABSTRACT

Research on sequence data concentrates on the discovery of frequently occurring patterns. However, comparatively less amount of work has been carried out in the area of sequence data classification and clustering.

The contribution of this thesis is in the development of new methods for classification and clustering of sequential data. The thesis introduces solutions for real-world applications dealing with classification of system call sequences in intrusion detection and clustering of web navigational sequences in web usage mining. We have chosen $DARPA'98$ IDS benchmark dataset for the sequence classification experiments. For sequence clustering experiments, *msnbc* web navigational benchmark dataset was chosen.

Initially, we established the hypothesis that while comparing two sequences, considering only the order or the content information embedded in the two sequences results in poor performance in classification and clustering tasks. kNN classification and PAM clustering algorithms are utilized for classification and clustering tasks, respectively and sliding window technique is used for extracting subsequences. To establish the hypothesis four distance/similarity measures namely, $Euclidean$, $Cosine$, $Jaccard$ and $Binary\,Weighted\,Cosine$ measures were used along with various sliding window sizes.

Based on the results from our hypothesis testing, we designed a new similarity metric, S^3M which considers both the order of occurrence as well as the content information of the two sequences being compared. Better classification accuracy and clustering quality were achieved with the newly devised measure.

In clustering of sequences, the goodness of the clusters was measured using average levensthein distance. We propose a new partitional clustering algorithm for sequence data, $SeqPAM$. $SeqPAM$ differs from PAM in mediod selection as well as the optimization function. The superiority of the proposed algorithm was demonstrated over PAM clustering algorithm.

Further, a new indiscernibility-based rough agglomerative hierarchical clustering algorithm for sequential data is proposed. Here, the indiscernibility relation has been extended to a tolerance relation with the transitivity property being relaxed. Initial clusters are formed using a similarity upper approximation. Subsequent clusters are formed using the concept of constrained-similarity upper approximation wherein a condition of relative similarity is used as a merging criterion. We compared the results of the proposed approach with that of the traditional hierarchical clustering algorithm using vector coding of sequences.

CHAPTER I

INTRODUCTION

The person who makes a success of living is the one who sees his goal steadily and aims for it unswervingly. That is dedication. - *Cecil B. De Mille*

In recent years advanced information systems have enabled collection of increasingly large amounts of data. To analyze huge amounts of sequential data, the interdisciplinary field of Knowledge Discovery in Databases (KDD) has emerged. KDD comprises of many steps namely, data selection, data preprocessing, data transformation, data mining and data interpretation and evaluation. Data mining forms a core activity in KDD. Data appear in various formats either as sequences of events or non-sequential. Data Mining applies efficient algorithms to extract interesting patterns and regularities from these data. Besides the sheer size of available data sources, the complexity of data has increased as well. Thus, new data mining methods are necessary to draw maximum benefit from this additional information. In this chapter, the taxonomy of data has been introduced and described in detail. Then data mining and its key tasks are discussed. Clustering and classification are discussed in detail as these are the primary focus of the thesis. Important application areas of sequence classification and clustering are discussed. In the subsequent section, we present an outline of the methods used for validating the proposed algorithms for the several benchmark datasets. Finally, the chapter concludes with an outline of the thesis.

1.1 Introduction

In recent years the amount of data that is collected by advanced information systems has increased tremendously. The resulting data volume is too large to be examined manually and even the methods for automatic data analysis based on classical statistics and machine learning often face problems when processing large,

dynamic data sets consisting of complex data. The interdisciplinary field of Knowledge Discovery in Databases (KDD) helps in analyzing these large volumes of data. KDD employs methods at the crosspost of machine learning, statistics and database systems. Farad et al. [56] define KDD as follows :

"Knowledge Discovery in Databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data."

According to this definition, data is a set of facts that is somehow accessible in electronic form. The term patterns indicates models and regularities which can be observed within the data. Patterns have to be valid, i.e. they should be true on new data with some degree of certainty. A novel pattern is not previously known or trivially true. The potential usefulness of patterns refers to the possibility that they lead to an action providing a benefit. A pattern is understandable if it is interpretable by a human user. KDD is a process comprising several steps that are perhaps repeated over several iterations.

The core step of KDD called *Data Mining*. Data mining can be defined as an activity that extracts new and nontrivial information contained in large databases. The goal is to discover hidden patterns, unexpected trends or other non-obvious relationships in the data using a combination of techniques such as, machine learning, statistics and databases. Data Mining finds application in a wide and diverse range of business, scientific and engineering scenarios. For example, large databases of loan applications are available which record different kinds of demographic and transactional information about the customers. These databases can be mined for interesting and hidden patterns leading to defaulting of loans which can in turn help in determining whether a new loan application should be accepted or rejected. Data mining can be helpful in the field of medical science by analyzing the medical records of the patients of a region by making early predictions for the potential outbreak of infectious disease or analysis of customer transactions for market research applications, etc. The list of application areas for data mining is large and is bound to grow rapidly in the years to come. A growing body of literature is available discussing the techniques for data mining and its applications [72, 80, 97].

A huge amount of data is collected every day in the form of sequences. These

sequential data are valuable sources of information not only to search for a particular value or event at a specific time, but also to analyze the frequency of certain events or sets of events related by particular temporal/sequential relationship. Examples includes, web server logs, online transaction logs, alarms/events and performance measurements generated by distributed computer systems and by telecommunication networks and financial transaction data. In general, sequential data can be either categorical (event streams) or numerical (time series). Both categorical as well as numerical data have been studied exhaustively in data mining. Many aspects of sequential data, like considering their order of occurrence, scalar nature and so on have proven their importance in emerging applications and posed several challenges calling for more research.

Sequential data arise in many areas of science and engineering. The data may either be a time series, generated by a dynamical system, or a sequence generated by a 1-dimensional spatial process, e.g., bio-sequences.

Important definitions of the terms being used in this thesis are presented below.
An **alphabet** generally denoted by $\sum$, is a set of letters, or a set of numbers, etc.
A **sequence** is a list of symbols (or events) from the alphabet arranged in a linear fashion, such that the order of the members is well defined. For example, sequences $< B, K, M >$ and $< K, M, B >$ defined over set of letters (alphabet) are different as the order of the symbols in both sequences is not the same. Sequences can be finite, as in the above example, or infinite, such as the sequence of all even positive integers $< 2, 4, 6, \cdots >$. The number of symbols (or events) determines the length of the sequence.
A **Subsequence** of a sequence is a new sequence which is formed from the original sequence by deleting some of the elements without disturbing the relative positions of the remaining elements. For example, suppose that $\sum$ is an alphabet and S is a sequence generated from $\sum$, denoted as $< s_1, s_2, s_3, \ldots, s_k >$ where $k \in N$ denotes the index position of symbols in sequence S. A subsequence of S is a sequence formed from it by considering symbols in strictly increasing order in the index set k. For example, $< s_1, s_3 >$ and $< s_2, s_3 >$ are subsequences of S.

1.2 Data Taxonomy

Data is a representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or by automated means.

Data may be *sequential* or *non sequential* in nature. Non-sequential data are those data where order of occurrence is not important. Ex: Market Basket data, value on dice in a binomial experiment. Sequential data are those data where order of occurrence is important to consider. Sequential data can be ordered with respect to time or some other dimension such as, space. For example, web logs, protein and DNA sequences, system calls recorded by an intrusion detection system, speech and so on. Further, sequential data can be classified as *temporal* or *non-temporal*. Temporal data are those data which have time stamp attached to it. Examples of such data are stock market data, meteorological data, and so on. Non- temporal data are those which are ordered with respect to some other dimension other than time such as for example, space. Examples of such data are web logs, sequence of system calls, and so on. Both temporal and non-temporal data can be further classified as *discrete* and *continuous*. Discrete data describes observations/elements that have a finite possible domain. Figure 1 shows the representation of discrete data generated from the set of alphabet of letters. Examples of discrete temporal sequential data includes, time logs of caller in a call center. Examples of discrete non-temporal data include system calls, web logs. Continuous data are observa-

Symbols from the set of letter alphabet, Σ	T	T	A	T	E	C	T	E	T	A	P	Y	R	T	T	C	H	G	B	D	E	W	S	L	K
Index position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Figure 1: Representation of discrete data

tions/ elements that take value from a continuous domain. Generally continuous data come from measurements. Figure 2 shows the representation of continuous data. Continuous data can accept any value between the defined range where as discrete data accepts defined set from the domain. Examples of continuous temporal data includes stock market data and weather data. Examples of continuous

non-temporal data includes image data and handwriting.

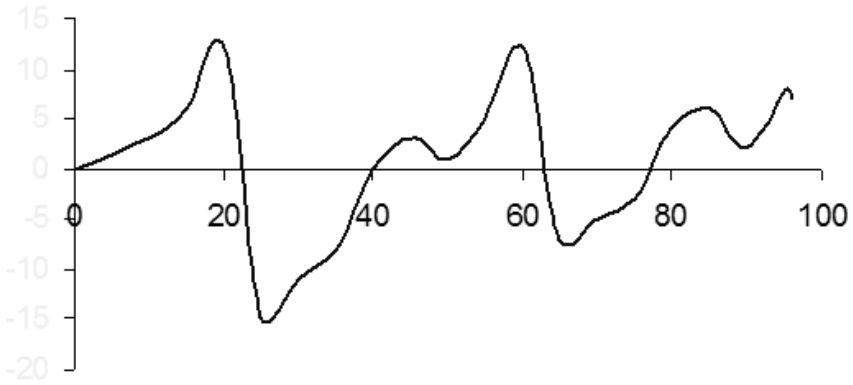

Figure 2: Representation of continuous data

This thesis mainly concentrates on the classification and clustering tasks of discrete non-temporal sequential data. Study of continuous temporal sequential data is out of scope of this thesis. However, wherever it is relevant we have given pointers the major references from the world of continuous sequential data.

1.3 Data Mining Tasks

In this section, we will describe some of the most important data mining tasks and kind of knowledge they mine:

• **Classification** (Supervised Learning) Classification is the task of learning a function that maps data objects to one or several classes in a predefined set of class labels. To learn this function, classification methods need labelled training set, containing data objects that are already mapped to the class as they belong to. After analyzing the training set, classification methods can predict class labels for unknown objects. A second purpose of classification is, deriving class models that explain how the objects are mapped to classes?

• **Clustering** (Unsupervised Learning) Clustering is the task of identifying a finite set of groups (or clusters) that describe the data. Thus, similar objects are assigned to the same category and dissimilar ones to different categories. Clustering is also called unsupervised learning as the mapping of objects to clusters is learned without external guidance in the form of labelled training data.

• **Association Rules** Finding association rules is the task of identifying rules that express co-occurrences within transaction databases. Given a database, association rules express which set of items appear together in the same transaction with a certain support or confidence. The most important example of transaction data for association rule generation is market basket data.

• **Data generalization** Data generalization derives compact representations for a sets of data objects.

• **Regression** The task of regression is to learn and approximation of the function which maps data objects to their values. To find a regression function, a training set of data objects and the corresponding function values is necessary. An additional goal of regression is to discover functional relationships between the feature values of the underlying training objects. Regression is closely related to classification, since both tasks learn functions from a labelled training set.

• **Dependency Modelling** Algorithms in this category are designed to find a model that describes significant dependencies among variables. For example, belief networks, hidden marked models.

• **Change and Deviation Detection** Algorithms in this category focus on discovering the most significant changes in the data from previously measured or normal values.

Recent research focus in data mining includes stream data mining, sequence data mining, web mining, text mining, visual mining, multimedia mining and multi-relational data mining [80]. In this thesis, we focus on issues related to sequence data mining.

One of the most important data mining problems is the discovery of patterns in sequences of events. Most of the research on sequence data concentrated on the discovery of frequently occurring patterns. The problem was first introduced by Agarwal and Srikant [7], and then generalized by Srikant and Agarwal [210] as a problem of mining frequently occurring sequential patterns in a set of sequences. Another approach was presented where various types of patterns called episodes were mined in a sequence of events [148, 149]. In many cases, a user might want to perform classification or clustering on sets of objects described by event sequences

associated with them. Lesh et al. [130] proposed an interesting approach to classification using features discovered from the sequential data. In this thesis, we aim to develop classification and clustering algorithms to sequential data to unearth the knowledge embedded in them.

1.4 Classification and Clustering tasks

In this section, we present more closely the classification and clustering tasks in detail and their relation to each other, as these are the main focus of the thesis.

• **Classification Task.** Classification is an important data mining task that has been studied extensively in the fields of statistics, pattern recognition, decision theory, machine learning, neural networks, etc. In classification task, we are given a set of example records called the training set, where each record consists of several fields or attributes. One of the attributes called the target attribute or class label indicates the class to which each record belongs.

There are several real world applications of classification algorithms. For example, in cellular telephone fraud detection, each record describes a telephone call with a class label of 0, if the call is legal or 1 if the call is illegal (that is, made from stolen cell) [55]. Another example involves computer intrusion detection where each trace of system calls describes a request for a computer network connection and the corresponding class label indicates whether the request results in an intrusion or not.

It is quite apparent that in these types of applications where training data consist of sequences which exhibit high correlation (because of serial dependence), the traditional supervised learning framework does not fit well. In order to perform classification of sequences, researchers or practitioners usually treat these sequences as independent events and traditional classification models with vector distance measure are applied.

• **Clustering Task.** Clustering has been studied in the field of machine learning and pattern recognition [51] and plays an important role in data mining applications such as scientific data exploration, information retrieval and text mining. It also plays a significant role in spatial database applications, web analysis, customer relationship management (CRM), marketing, computational biology and many other

related areas. Although finer details of data are lost due to representation of data into fewer groups, clustering results in simple and understandable groups.

Clustering algorithms have been classified using different taxonomies based on various important issues such as algorithmic structure, nature of clusters formed, use of feature sets, etc [101]. Broadly speaking, clustering algorithms can be divided into two types - *hierarchical* and *partitional*.

1.4.1 Relationship between classification and clustering tasks

Classification and clustering are strongly connected. Classification tries to learn the characteristics of a given set of classes, whereas clustering finds a set of classes within a given data set. An important feature of clustering is that it is not necessary to specify a set of example objects and their grouping. Therefore, clustering can be applied in applications where there is little or no prior knowledge about the groups or classes in a database. However, the usefulness of the results of clustering is often subject to individual interpretation and strongly depends on the selection of a suitable similarity measure. In applications for which the existence of a dedicated set of classes is already known, the use of *classification algorithms* is more advisable. In these cases providing example objects for each class is usually much easier than constructing a feature space in which the predefined classes are grouped into delimited clusters. Furthermore, the performance of a classifier can easily be measured by the percentage of correct class predictions it achieves.

In this thesis, we shall first establish that by incorporating sequential information in the traditional kNN classifier with various distance/similarity measures the efficiency of kNN classifier would be improved irrespective of the distance/similarity measure used. Our hypothesis is that better groupings would be achieved with PAM clustering algorithm when sequential information is embedded in the distance/similarity measures used with the PAM algorithm. Based on these insights, we plan to devise a new measure for sequential data which keeps into account both the order of occurrence information and the content information. We would evaluate the new measure in classification and clustering tasks.

1.5 Similarity Measure for sequences

In many data mining applications, both classification as well as clustering algorithms require a distance/similarity measure. In clustering, we are given unlabelled data and we have to group the data based on a similarity measure. These data may arise from diverse application domains. They may be music files, system calls, transaction records, web logs, genomic data and so on. In these data there are hidden relations that should be explored to find interesting information. For example, from web logs one can extract the information regarding the most frequent access path; from genomic data one can extract letter or motif (sequence of letters) frequencies; from music files one can discover harmonies etc. One can extract features from sequential data represent them as vectors and cluster the data using existing clustering techniques. Similar to clustering, in classification task also a similarity measure is required to determine the class membership of test data or sequence. The central problem in similarity based clustering/classification problems is to come up with an appropriate similarity metric for sequential data.

Researchers or practitioners, when dealing with sequences, usually first convert them into frequency vectors treating all the events within the sequences as independent of each other. Treating sequences in this manner results in a loss of the sequential information embedded in them and leads to inaccurate classification or clustering. This thesis first establishes the hypothesis that classification and clustering tasks, if carried out using sequential information of sequences give better results irrespective of the distance/similarity measures used. We feel that classification or clustering tasks would require a new similarity measure that takes into account both the order of occurrence information as well as the content information.

The main problem in calculating similarity between sequences is finding an algorithm that computes a common subsequence of two given sequences as efficiently as possible [203]. Intuitively, when two sequences are similar it is expected that the underlying subsequences are also similar, in particular, the longest common subsequence (LCS). In this thesis, we plan to incorporate LCS information into the new similarity measure.

1.6 Application areas

Classification and clustering of sequential data finds applications in various fields namely, computer security, speech recognition, automatic translation, web mining, hand-writing recognition, stock market prediction and many more. This thesis work considers applications in computer security (Intrusion detection) and web mining (User navigational pattern). Sequence classification is demonstrated in the computer security area whereas clustering task has been addressed in the web mining area. In this section, we briefly describe the two application areas.

Computer Security: The security of a computer system is compromised when an intrusion takes place. An intrusion can be defined as "any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource" [13]. Intrusion prevention techniques, such as user authentication (for example, using passwords or biometrics) and information protection (for example, encryption) have been used to protect computer systems as a first line of defense. Intrusion prevention alone is not sufficient because as systems become ever more complex, there are always exploitable weaknesses in the systems due to design and programming errors.

Intrusion detection is needed as a *wall* to protect computer systems. The elements central to intrusion detection are: resources to be protected in a target system, i.e., user accounts, file systems, system kernels, etc; models that characterize the "normal" or "legitimate" usage of these resources; techniques that compare the actual system activities with the established models, and identify those that are "abnormal" or "intrusive".

Many researchers have proposed and implemented different models which define different measures of system behavior, with an *adhoc* presumption that normal and misuse will be accurately manifested in the chosen set of system features that are modeled and measured. Intrusion detection techniques can be categorized into *misuse detection* which uses patterns of well-known attacks to identify intrusions; and *anomaly detection*, which tries to determine whether the deviation from the established normal usage patterns can be flagged as *intrusions*.

There are two types of Intrusion Detection Systems (IDS) namely:

- **Host-based Intrusion Detection System** (HIDS)
- **Network-based Intrusion Detection System** (NIDS)

All intrusion detection (HIDS and NIDS) is based on analyzing a set of discrete, time-sequenced events for patterns of misuse. All intrusion detection sources, network or host, are sequential records that directly reflect specific actions and indirectly reflect behavior. Host-based systems examine events like what files were accessed and what applications were executed. Network-based technologies examine events such as packets of information exchanged between computers (network traffic).

The recent development in the field of data mining has made available a wide variety of algorithms, taken from the fields of statistics, pattern recognition, machine learning and database.

This thesis is mainly concerned with designing an anomaly based host intrusion detection system using classification and sequence analysis task of data mining.

Web mining: When a user browses the web at different times, he or she could be accessing pages that pertain to different page categories. For example, a user might be looking for research papers at one time and airfare information for conference travel at another. That is, a user can exhibit different kinds of interests at different times, which provides different contexts underlying a user's behavior. However, different kinds of interests might be motivated by the same kind of interest at a higher abstraction level (computer science research, for example). That is, a user might possess interests at different abstraction levels where the higher-level interests are more general and the lower-level ones are more specific. During a browsing session, general interests are in the back of one's mind while specific interests are the current focus. Identifying the appropriate context underlying a user's behavior is important in more accurately pinpointing her/his interests.

The problem of web log mining consists of automated analysis of web access logs in order to discover trends and regularities (patterns) in users' behavior. The discovered patterns are usually utilized for improvement of web site organization and presentation. The term *adaptive web sites* has been proposed to denote such automatic transformation of web sites [180].

One of the most interesting web log mining methods is clustering of web users [232]. The problem of clustering of web users (or segmentation) is solved by using web access log files to partition a set of users into clusters such that users within a cluster are more similar to each other than users from different clusters. The discovered clusters can then help in on-the-fly transformation and presentation of the web site content. In particular, web pages can be automatically linked by artificial hyperlinks. The idea is to try to match an active user's access pattern with one or more of the clusters discovered from the web log files. Pages in the matched clusters that have not been yet explored by the user may be presented as navigational hints for the user to follow subsequently.

1.7 Problem Definition and Contribution

The primary problems addressed in this thesis are classification and clustering of sequential data. The following are related sub-issues and contributions of this thesis.

- Does incorporation of the order of occurrence (sequential aspects) information enhance the efficiency of classifier on sequential data ?
- Can a better grouping of data that preserves sequentiality be achieved by incorporating sequential information or not ?
- Devising a new similarity measure which takes into account the order as well as content information in clustering and classification tasks.
- Designing an efficient classifier for detecting anomalous sequences of system calls in the context of intrusion detection.
- Developing a new clustering algorithm to form groups that preserve sequential nature of data, for example, in the domain of web usage mining.
- Formulating a new hierarchical clustering algorithm for sequential data using the concept of rough set theory.

In this thesis, we first tested the hypothesis that the order of occurrence information does enhance classification and clustering efficiency. We came up with

a new similarity measure for sequential data which considers both the order of occurrence and the content information when computing similarity between two sequences. We propose a new partitional clustering algorithm which finds better grouping of sequences when sequential nature is taken into consideration for clustering. Upper approximation, a concept from rough set theory, was used to propose a new hierarchical clustering algorithm. This thesis also provides an overview of recent related work in the area of sequence clustering and classification. Recent work in the area of Intrusion detection system and web usage mining is also summarized.

We summarize the main contributions of the thesis below:

- **Sequential Information is important.** Researchers while dealing with sequential data usually, convert them into frequency vector treating all the events within the data sequence as independent of each other. Treating sequential data in this manner, ignores the embedded sequential information thus results in inaccurate classification or clustering. The thesis first establishes the hypothesis that classification and clustering tasks, if carried out using sequential information of sequential data, give better results irrespective of the distance/similarity measure used.

  Experiments were conducted on *DARPA'98 IDS* dataset using k-Nearest Neighbor classifier with Euclidean distance, Jaccard similarity function, Cosine similarity measure and Binary Weighted Cosine (BWC) similarity measure. Each distance/similarity metric was individually experimented with *kNN* classifier and sequence information is incorporated using different sliding window sizes.

  Receiver Operating Characteristics (ROC) curves for all the four measures were analyzed and also the area under the ROC curve (AUC) was calculated. It was observed from the results that as the sliding window size increases, high Detection Rate (DR), close to the ideal value of 1, is observed with all the distance/similarity measures.

  It was noted that as the length of the subsequence increases, better sum-of-squared error was observed irrespective of distance/similarity measure used

in the cluspering experiments. The experimental results obtained on various subsequence lengths for various *k*, the number of clusters, with the four distance/similarity measures gave an additional support to our hypothesis.

- **Content information is also important.** Apart from the sequential information, content information is also important for sequential data while performing classification and clustering tasks. It was observed that the content information further enhances the accuracy of classification and clustering algorithms.

  Superior classification and clustering results were observed using Jaccard measure, giving us a intuition that the content information is also important for processing sequential data.

  Thus, we believe that to carry out classification or clustering tasks, a new measure could be devised which considers both the order of occurrence information as well as the content information.

- **Devising a new similarity measure for sequential data.** A new similarity measure S^3M (Sequence and Set Similarity Metric) was devised for classification and clustering of sequential data. The designed metric qualifies as a similarity metric in the sense of obeying reflexivity, symmetry and normalization properties.

  S^3M similarity measure performed well for classification and clustering of sequential data.

- **Efficient classifier for classification of abnormal system calls.** *kNN* classification with proposed S^3M metric was used on DARPA'98 IDS dataset. The efficiency of *kNN* classifier with the S^3M metric was compared with and *cosine* as well as *BWC* metrics. Extensive experiments were performed with various values of *p* (p being the parameter controlling sequence similarity) in S^3M measure and designed an efficient classifier for Intrusion Detection.

- **Design of a new partitional clustering algorithm for sequential data.** We

conducted experiments and compared the results of *PAM*, a standard clustering algorithm, with two similarity measures: *Cosine* and S^3M. A new partitional clustering algorithm with modifications of *PAM* was designed and named as $SeqPAM$. $SeqPAM$ differs from PAM in two ways. Firstly, $SeqPAM$ uses a guided approach for mediod selection as opposed to random selection in PAM. Secondly, cost optimization function used is the maximization of pairwise similarity of data sequences within the cluster as compared to minimization of cost function in PAM. Additionally, $SeqPAM$ uses S^3M similarity measure thus capturing the embedded sequential information in the data sequences. The goodness of the clusters resulting from both the measures was computed using a cluster validation technique based on Levensthein Distance (*LD*).

The Average Levensthein Distance (*ALD*) values obtained on the *msnbc* dataset using *SeqPAM* clustering algorithm indicate that the groupings have preserved sequential information embedded in the *msnbc* dataset.

We also noted from the *LD* for each cluster that for *SeqPAM* the *LD* value was less than that for *PAM*. *LD* values are indicators of distance between cluster representatives. The *LD* measure indicates that in the *SeqPAM* clustering algorithm, the cost of converting a sequence to its cluster representative is less as compared to the *PAM* clustering algorithm. All the experiments for clustering algorithms were carried out on benchmark *msnbc* web navigational dataset.

- **Design of new hierarchical agglomerative clustering algorithm for sequential data.** Hirano and Tsumoto [86, 87] proposed indiscernibility based clustering method that can handle relative proximity. In this work, we designed a new indiscernibility-based rough agglomerative hierarchical clustering algorithm for sequential data. We used S^3M similarity measure for computing similarity between web data sequences. The proposed rough agglomerative clustering approach for sequential data uses the concept of constrained-similarity upper approximation. The proposed approach enables the merging of two or more clusters at each iteration and hence facilitates fast hierarchical clustering. We tested our clustering algorithm on *msnbc* web navigation dataset that is intrinsically sequential in nature. The proposed approach is

compared to the traditional complete linkage based hierarchical clustering algorithm.

1.8 Experimental validation

Experimental investigation was carried out in this thesis using *DARPA'98 IDS* dataset and *mscbc* datasets. *DARPA'98 IDS* dataset was used for classification task whereas msnbc dataset was used for clustering task. kNN classification technique was used. Results of classification technique were validated using the Receiver Operating Characteristic (ROC) curve and the Area Under Curve (AUC) measure of ROC. We compared the results of Intrusion detection classification with the techniques of Vemuri et al. [139] and Rawat et al. [195].

Clustering task was performed on web mining domain where web usage logs were clustered. A new partitioning clustering technique SeqPAM was proposed in this thesis. As these web logs are sequential in nature Levensthein distance based cluster validation technique was use to establish the superiority of the proposed technique over PAM (Partition Around Mediod) clustering algorithm.

Based on the upper approximation concept of Rough Set Theory, a new hierarchical agglomerative clustering algorithm is proposed called constrained-similarity upper approximation. Complete linkage based hierarchical clustering algorithm was used to cross-compare the results of the proposed technique.

Details of *DARPA'98 IDS* and *msnbc* web navigation datasets are presented in Appendix A and B respectively.

1.9 Dissertation structure and content

The thesis aims at designing new classification and clustering algorithms for sequential data. In order to perform classification and clustering of data a new sequence similarity measure has been designed. This thesis considers the classification and clustering tasks from computer security and web mining areas, respectively. This thesis is divided into eight chapters and four appendices.

Chapter 2: Literature Survey. This chapter describes the recent developments and

the research work carried out in the field of clustering and classification of sequential data.

Chapter 3: Classification and clustering using subsequence information. This chapter deals with the establishment of the hypothesis that the order of occurrence (sequence information) affects the performance of classification and clustering techniques. Results of experiments conducted using sliding window concept to extract partial subsequence information from the data sequences are presented in this chapter. Classification task was carried out using *kNN* classification algorithm and clustering task was carried out using *PAM* clustering algorithm. The efficiency of classifier was shown using ROC curve and AUC. Sum of squared error was used to measure the cluster quality of the clusters formed using *PAM* clustering technique.

Chapter 4: A new Similarity Metric for Sequential Data. Based on the the results of previous chapter, we came up with a new similarity measure S^3M (Sequence and Set Similarity Metric) which considers both the order of occurrence information and the content information while computing similarity between sequences. The chapter also presents theoretical justification for the choce of parameter controlling the sequence information (p).

Chapter 5: Application of sequence classification in Intrusion detection. In this chapter, we used the proposed similarity measure as a part of *kNN* classification algorithm for intrusion detection. Experiment were carried out on DARPA'98 IDS dataset. ROC curve analysis and area under ROC curve was used to test the efficiency of the classifier. This chapter also presents the recent related literature in the area of intrusion detection.

Chapter 6: Application of sequence clustering in web user navigation. In this chapter, we propose a new partitional clustering algorithm for sequential data, $SeqPAM$. We review the recent literature in the area of web usage mining. The clusters obtained were compared to traditional partitioning clustering algorithm, PAM. The validation of the quality of clusters is done using the Average Levensthein Distance and Levensthein Distance measures.

Chapter 7: Clustering of sequential data using rough sets. Upper approximation, a property of rough set theory, has been used to come with a new hierarchical

agglomerative clustering algorithm. In this chapter, the new algorithm has been compared with the complete linkage based agglomerative hierarchical clustering algorithm.

Chapter 8: Conclusion. We summarize the major contributions of the research work carried out in this thesis. We also highlight the future scope and further research directions.

Appendix A describes about the *DARPA'98 IDS* dataset. All the classification experiments reported in this thesis are performed on *DARPA'98 IDS* dataset.

Appendix B describes about the *msnbc* web navigation dataset. All the clustering experiments reported in this thesis are performed on *msnbc* web navigation dataset.

Appendix C describes about the basic folkfore algorithm used for computing longest common subsequence. The new proposed measure S^3M has longest common subsequence as one of the component.

Appendix D describes about the basic levensthein distance algorithm. Levensthein distance measure has been used in this thesis to compute the meaningfulness of clusters formed in chapters 6 and 7.

Overall, this thesis covers both the theoretical aspects as well as applied aspects of a sub-domain of data mining research called sequence mining. Theoretical characterization of the proposed similarity metric and the proposed algorithm have been done. The applied aspect of the research reported in the thesis is reflected in the exhaustive experimentation conducted on the benchmark datasets related to intrusion detection and web mining. The thesis touches on both traditional as well as soft computing techniques for pattern recognition tasks.

CHAPTER II

LITERATURE SURVEY

You don't destroy the mystery of a rainbow by understanding the light processes that form it. - *Anne McLaren*

Data mining aims at extracting previously unknown and potentially useful knowledge from large sets of data. A large amount of data is collected every day in the form of sequences. These event sequences can be analyzed not only to search for a particular value or event at a specific time, but also the frequency of certain events, or sets of events related by particular temporal or non temporal relationship. These types of analysis can be very useful for deriving implicit information from the data and for predicting the future behavior of the monitored processes.

In this chapter, we review the recent available literature on classification and clustering of sequential data. This chapter also presents some of the major application areas of sequential data.

2.1 Introduction

The types of structures data mining algorithms look for can be classified in many ways [80, 227]. It is useful to categorize outputs of data mining algorithms into models and patterns [81]. Models and patterns are structures that can be estimated from the data. These structures may be utilized to achieve various data mining tasks.

A model is defined as a global, high-level and abstract representation for the data. Generally models are specified by a collection of parameters estimated from the given data. It is possible to further classify models based on whether they are predictive or descriptive. Predictive models are used in forecast and classification applications while descriptive models are useful for data summarization. For example, autoregression analysis can be used to predict future values of a time series based on its historical data. Another popular class of predictive models is

Markov models. It has been extensively used in sequence classification applications. On the other hand, spectrograms (obtained through time-frequency analysis of time series) and clustering are good examples of descriptive modelling techniques. These are useful for data visualization and summarization.

A pattern is a structure that makes a specific interpretation about a few variables or data points. Spikes, for example, are patterns in a real-valued time series. Similarly, in symbolic sequences, regular expressions constitute a useful class of well-defined patterns. In biology, genes, regarded as the classical units of genetic information, are known to appear as local patterns interspersed between chunks of non-coding DNA. Matching and discovery of such patterns are very useful in many applications. Due to their readily interpretable structure, patterns play a particularly dominant role in data mining.

Although patterns and models distinguish themselves and are useful from the point of view of comparing and categorizing data mining algorithms, there are cases when such a distinction is not clear. This is due to the interdisciplinary nature of the field data mining [66].

2.2 Important application areas of sequential data

Supervised and unsupervised learning from sequential data has many possible applications. Some of the applications of sequence learning tasks are as follows:

Speech recognition [67, 193]: Speech recognition is often defined as the automatic transcription of human utterances as sequence of words. In the training phase, given a large number of utterances with their correct transcriptions, the task is to learn the mapping from the acoustic signal to the word sequence, so as to later be able to recognize new, previously unknown utterances.

Automatic (Machine) translation [49]: The task is to learn the mapping from a sequence of words in one language to a sequence of words in another language, which can be viewed as a sequence prediction problem with categorical variables in the input and output space.

Gesture Prediction [42, 230, 212]: In gesture (or human body motion) recognition, video sequences containing hand or head gestures are classified according to the actions they represent or the messages they seek to convey. The gestures

or body motions may represent, for example, one of a fixed set of messages like waving hello, goodbye, and so on. There could be the different strokes in a tennis video, or in other cases, they could belong to the dictionary of some sign language and so on.

Hand-writing recognition [35, 162]: Hand-writing recognition can today be done to some extent by many hand-held computers and PDAs. A large number of words are written by an appropriate number of people with different writing styles- the movement of the pen is recorded using a digitizer board. Based on the sequential data describing the relationship between pen movement and the corresponding words or letters, models are built that are used to recognize new words.

Stock market prediction [154]: The problem of predicting the value of stocks or currencies can be formulated as a sequence prediction problem. Historical data provides the example sequences. For example, the stock price index recorded over time is used to train a model for stock price forecasting.

All of the sequence predictions problems discussed are of different nature. Any scientific approach to solving them involves first a unification and abstraction of all the specific problems to one scientific core problem, which allows us to use established scientific methods and knowledge from other scientific areas. These areas are as follows:

Pattern recognition [18, 21, 51]: Pattern recognition is also referred to as learning from examples. Pattern recognition often involves the definition of stochastic models like neural networks or Hidden Markov Models, which are trained on training data and tested on unseen test data. These methods rely on the practical aspects of probability theory and are strongly related to coding and compression.

Information theory [76]: Information theory defines consistent ways of measuring the amount of information in data, called entropy. Many problems of pattern recognition which are formulated to maximize a probability, can often also be formulated to maximize the amount of information flow through some channel.

2.3 Classification

This section presents a brief introduction to classification which is the data mining task that is discussed in this thesis. General classification methods are discussed first and then followed by a survey of recent literature on sequential data classification including the classification methods. Some of the methods for the foundation for the novel techniques developed in this thesis for addressing efficient sequence classification are discussed.

2.3.1 Types of classification algorithms

The task of classification is to learn a function that maps data to the available set of classes. A classifier learns from a training set, containing a large number of already mapped objects for each class. The training objects are considered to be labelled with the name of the class they belong to. Classification is also called supervised learning because it is directed by these labelled objects.

Here, we briefly outline the basic classification algorithms. Though the methods to achieve classification vary based on the dataset and the model used but all the classifiers have something in common. The commonality is that they divide the given object space into disjunctive sections that are mapped to a given class. In the following, the most important approaches to classification are described.

Decision Trees.

Decision trees are powerful and popular tools for classification and prediction. Decision trees are more popular because in contrast to the neural network based approach they generate rules. These rules can easily be interpreted so that we can understand them. These rules can also be transformed into a database access language like SQL in order to retrieve quickly the records falling into a particular category.

A decision tree is a tree with the following characteristics:

- Each inner node corresponds to one attribute.
- Each leaf is associated with one of the classes.
- An edge represents a test on the attribute of its parent node.

For classification, the attribute values of a new object are tested beginning with the root. At each node the data object can pass only one of the tests that are associated with the departing edges. The tree is traversed along the path of successful tests until a leaf is reached.

Multiple algorithms have been proposed in the literature for constructing decision trees [23, 63, 192]. Generally, these algorithms split the training set recursively by selecting an attribute. The best splitting attribute is determined with the help of quality criteria. Examples of such quality criteria are information gain, gini index, shannon information theory and statistical significance tests.

The advantages of decision trees are that they are very robust against attributes that are not correlated to the classes because those attributes will not be selected for a split. Another more important feature is the induction of rules. Each path from the root to a leaf provides a rule that can be easily interpreted by a human user. Thus, decision trees are often used to explain the characteristics of classes.

The drawback of decision trees is that they are usually not capable of considering complex correlations between attributes because they only consider one attribute at a time. Thus, decision trees often model correlated data by complex rules which tend to overfit.

k-Nearest Neighbor Classification.

Nearest neighbor classifiers are based on the idea that an object should be predicted to belong to the same class as the objects in the training set with the biggest similarity. To use kNN classification, it is required to have a suitable similarity search system in which the training data is stored. Classification is done by analyzing the results of a kNN query. The simplest way to determine a classification result of a kNN classifier is the majority rule. The objects in the query result are counted for each class and the class having the majority count is predicted to be the class of the test object.

Another method is to consider the distances to the object to weigh the impact of each neighboring object. Thus, a close object contributes more to the decision than an object having a large distance.

kNN classifiers use the class borders of the objects within the training set and

thus do not need any training or model building apriori. As a result, kNN classifiers cannot be used to gain explicit class knowledge to analyze the structure of the classes. kNN classification is also known as a *lazy learning* algorithm. The parameter that determines the neighborhood size, k, is very important to the classification accuracy achieved by a kNN classifier. If k is chosen too small, classification tends to be very sensitive to noise and outliers. On the other hand, a too large value for k might extend the result set of the k nearest neighbor by objects that are too far away to be similar to the classified object.

Since kNN classification works on the training data, the classification time depends on the efficiency of the underlying similarity search system. In the case of large training sets linear search becomes very inefficient. Suitable index structures can offer a better solution to this problem [16, 37, 141]. Deleting unimportant objects from the training dataset also helps in speeding up the kNN classification algorithm [24]. kNN classification algorithms can be sped up by building the centroid for the objects of each class and using only the centroids and nearest neighbor classification [77]. This approach is rather simple but gives accurate classification for text data.

Support Vector Machines.

Support vector machines (SVMs) were first introduced in 1995 for classification of input feature vectors [40]. SVMs distinguish the objects of two classes by linear separation which is achieved by determining a separating hyperplane in the object space [25, 36]. The idea of SVMs is to find the hyperplane providing the maximum level of generalization and thus avoiding overfit as much as possible. The vectors in the training set that have minimal distance to the maximum margin hyperplane are called support vectors. The location of the maximum margin hyperplane does only depend on these support vectors and thus, the classifier was named *support vector machine*.

To determine the exact position of the maximum margin hyperplane and to find out the support vectors, a dual optimization problem is formulated which can be solved by algorithms like Sequential Minimal optimization [187].

The problem of linear separation is that there is not always a hyperplane that is able to separate all training instances. Therefore, two improvements for SVMs

have been introduced that enable SVMs to separate almost any kind of data.

The first improvement is the introduction of soft margins. The idea of soft margins is to penalize, but not prohibit classification errors while finding the maximum margin hyperplane. Thus, the maximum margin hyperplane does not necessarily separate all training instances of both classes. If the margin can be significantly increased, the better generalization can outweigh the penalty for a classification error on the training set.

The second improvement is the introduction of kernel functions. For many real-world applications, it is not possible to find a hyperplane that separates the objects of two classes with sufficient accuracy. To overcome this problem the feature vectors are mapped into a higher dimensional space by introducing additional features that are constructed out of the original ones. Since this mapping is not linear, hyperplanes in the so-called kernel spaces provide much more complicated separators in the original space. This way the data in the original space is separated in a non linear fashion. An import characteristic of the use of kernel functions is that the calculation of a maximum margin hyperplane in the kernel space is not much more expensive than in the original space. The reason for this effect is that it is not necessary to calculate the feature vectors in the kernel space explicitly. Since the optimization problem calculating the maximum margin hyperplane does only use a scalar product in the feature space, it is enough to replace this scalar product with a so-called kernel function to calculate the maximum margin hyperplane in the kernel space. SVMs have been extended to multi-class problems [186].

However, the training of SVMs tends to take large periods of time, especially for multi-class variants calculating many binary separators. The models built by SVMs do not provide any explicit knowledge that might help to understand the nature of the given classes.

Bayes Classifier

Baysian classifiers are based on the assumption that the objects of a class can be modelled by a statistical process [153]. Each data object has its origin in the process of a given class with a certain probability and each process of a class generates objects with a certain probability called prior probability. To decide which class is to be predicted for a given object, it is necessary to determine the probability called

the posterior probability of the object. It describes the probability that an object has its origin in that particular class. To determine the a posteriori probability, the rule of Bayes is used.

In bayesian classifier training is very fast. Also the model designed is simple and intuitive. In bayesian classifier error is minimized subject to the assumptions of independence of attributes and data satisfying distribution model. The major drawback of bayesian model is that it assumes a normal distribution of patterns. In addition, it can not model disjoint class boundaries automatically.

There are several additional approaches that are used for classification. For example, Neural networks are a very powerful direction of machine learning trying to rebuild the learning mechanism that can be found in the human brain. Among several other tasks, neural networks are applicable to classification. There are multiple variants of neural networks. An overview of neural networks can be found in many available publications [20]. Another direction that can be applied to classification is inductive logical programming (ILP). ILP is often used for data mining in multi-relational data and employs search algorithms that find logical clauses that are valid in a database of facts [159].

In the next section, we review the available literatures on classification techniques for sequential data.

2.3.2 Classification of sequential data

'Sequential data posses serial correlations and since traditional classification algorithms assume independence of attributes, traditional methods are not suitable for sequence classification tasks. The sequence classification task can be performed by ignoring the order of attributes and by aggregating the elements over the sequence [32]. Over the years, sequence classification applications have seen the use of both pattern based as well as model-based methods. In a typical pattern based method, prototype feature sequences are available for each class. Then the classifier searches over the space of all prototypes, for the one that is closest (or most similar) to the feature sequence of the new pattern. The prototypes and the given features vector sequences are of different lengths. Thus, in order to score each

prototype sequence against the given pattern, sequence aligning methods like Dynamic Time Warping are needed.

The literature on pattern-based sequence classification is not vast. Jain and Zongker [102] designed a dissimilarity measure, for a handwritten digit recognition problem, based on deformable templates. A multidimensional scaling approach was then used to project this dissimilarity space onto a low-dimensional space, where a 1-nearest-neighbor (1-NN) classifier was employed to classify new objects. Graepel et al. [70] investigate the problem of learning a classifier based on data represented in terms of their pairwise proximities, using an approach based on Vapniks structural risk minimization [217]. Jacobs, Weinshall and Gdalyahu [99] have studied distance-based classification with non-metric distance functions (the metric does not satisfy the triangle inequality). Basic features of feature-based methods were first addressed by Pekalska et al. [178]. They showed by experiments in three real handwritten datsets, that a Bayesian classifier (the RLNC - regularized linear normal density-based classifier) in the dissimilarity space outperforms the nearest neighbor rule [177].

Another popular class of sequence recognition techniques is a model based method called, Hidden Markov Models (HMMs). Here, one HMM is learnt from training examples for each pattern class and a new pattern is classified by analyzing which of these HMMs is most likely to generate it. In recent times, many other model-based methods have been explored for sequence classification.

A Naive Bayes sequence classifier is based on one of the simplest models of sequence data. It assumes that each element (letter) of the sequence is independent of other elements of the sequence for the given class. Training phase of Naive Bayes sequence classifier involves estimating the probabilities for each letter of the alphabet conditioned on the class. Naive Bayes classifiers, because of their simplicity and low computation cost, are often used in applications ranging from text classification [151] to biological sequence classification [12, 231] with varying degrees of success (as estimated by standard measures of classifier performance for example, classification accuracy). However, the independence assumption of Naive Bayes effectively ignores the sequential nature of the data. Hence, generative models of sequence data that relax the strong independence assumption of the

Naive Bayes model, the associated sequence classifiers, and efficient algorithms for learning accurate sequence classifiers from data are of significant interest.

The dependencies among the neighboring elements of a sequence can be modelled using k-grams [33]. An approach based on Naive Bayes classifier whose inputs consist of k-grams as opposed to single letters can be adopted for sequence classification. Since the successive k-grams in a sequence have $k-1$ elements in common, the use of k-grams as input features for a Naive Bayes classifier consistently and systematically violates the Naive Bayes assumption, that is, the inputs to the classifier are independent for the given class.

The well-known Hammersley-Clifford theorem [41] can be used to construct a generative Markov Model of order $k-1$, or Markov Model(k- 1) [179] or equivalently, the so-called Naive Bayes(k) that models the dependencies among k neighboring elements of a sequence. This model was shown to have consistent improvement in accuracy relative to Naive Bayes in sequence-based classification with the performance of the classifier improving with increasing values of k for large amount of training dataset [179].

Sliding window techniques are the most common techniques for sequence classification. In sliding window technique for a fixed parameter, called the window size L corresponding L-grams dimensions of elements are created. Thus, if the domain size of elements is m, the number of possible dimensions are mL. The sliding window approach has been applied to classify sequences of system calls as intrusions or not [93, 127]. The main shortcoming of the sliding window method is that it creates an extremely sparse space. A technique to address the shortcoming of sliding window technique is proposed by Leslie et al. [131].

Another commonly used technique for sequence classification is to consider the ordering information of the symbols in a sequence. For categorical elements, an example of such order-sensitive derived features is the number of symbol changes or the average length of segments with the same element. For real-valued elements, examples are properties like Fourier coefficients, Wavelet coefficients, and Autoregressive coefficients.

However, in all of the techniques mentioned above for sequence classification none of them considers both the order of occurrence information or the content

information simultaneously at a time. Some do consider the content information while some considers the order information while performing the sequence classification task. In this thesis we used kNN classification algorithm with different distance/similarity measures for the classification of sequential data. Commonly used sliding window technique was used for capturing the content information and performing sequence classification (Chapter 3). Later in chapter 4 we designed a new metric which considers both order of occurrence as well as the content information while computing similarity among two sequences. The designed measure is used with kNN classification algorithm in chapter 5 for performing sequence classification task for intrusion detection.

2.4 Clustering

The second task dealt with in this thesis is *clustering*. Clustering is an initial and fundamental step in data analysis. Although finer details of data are lost due to representation of data into fewer groups, clustering results in simple and understandable groups.

Given a set of sequences, during clustering the goal is to create groups such that similar sequences are in the same group and sequences in different groups are dissimilar. Like classification, clustering of sequences is also an extensively researched topic with several formulations and algorithms.

In this section, first we present the types of clustering algorithms followed by available literatures on clustering of sequential data.

2.4.1 Types of clustering algorithms

Partitioning method.

Partitioning clustering algorithms partition a database of n objects into k distinct clusters. Each cluster contains at least one object and each object belongs to exactly one cluster. In order to find a good clustering, partitioning methods divide the data set into k initial partitions and afterwards optimize the cluster quality in several iterations. In each iteration some objects are moved from one cluster to another one, improving the quality of the clustering. If it is not possible to augment the quality by relocating any object then the algorithm terminates. The partitioning clustering

algorithms are heuristic and do not usually guarantee maximal quality clustering. The most prominent examples of partitioning clustering algorithms are k-Means clustering and k-Medoid methods like PAM (Partitioning Around Medoids) or CLARANS (Clustering Algorithm based on Randomized Search) [80]. Partitioning clustering algorithms are well suited for small to medium sized databases and are useful to find k spherical clusters to describe a data set. However, partitioning clustering algorithms need a specification of number of clusters to be found and do not find arbitrary shaped clusters in large databases.

Hierarchical Methods.

Hierarchical methods create a hierarchical decomposition of the data set. Hierarchical clustering allows for smaller clusters to be part of bigger clusters that are more general. There are two approaches to hierarchical clustering, *agglomerative* and *divisive* clustering. *Agglomerative clustering* follows a bottom-up approach. Each object is initially a cluster of its own. In the next step, two clusters that are the closest are merged. The merging of clusters is repeated until either a complete tree of clusters is built or a termination condition is reached. This cluster tree is called dendrogram and contains clusters of varying size. Divisive methods follow a top-down approach and successively divide already found clusters into smaller clusters. The clustering terminates if each cluster consists of exactly one object or a termination condition is reached. In basic hierarchical clustering algorithms the decision that an object belongs to a cluster cannot be revised. Though this characteristic helps to find clusters efficiently, the quality of the clustering might suffer. The examples of hierarchical clustering algorithms include CURE (Clustering Using Representatives), Chameleon and BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [80].

Density based methods.

Density-based clustering algorithms define clusters according to the dense areas in the database. They are capable of finding an arbitrary number of arbitrary shaped clusters. Another advantage of density based clustering is its capability to consider noise which is not to be counted toward any cluster. In order to identify dense areas in the database, density-based algorithm employ ϵ-range queries to decide if the ϵ-neighborhood of a data object contains a sufficient number of other data

objects. Clusters are defined by connected areas where the data objects contain objects with dense ϵ-neighborhoods. Examples of density-based clustering algorithms includes DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points to Identify the Clustering Structure) [80].

Grid based methods.

Grid-based methods divide the object space into a finite number of cells that form a grid structure. All clustering operations are performed on this grid structure. The advantage of grid-based clustering is the fast computation. The drawback of these approaches is that the result is strongly dependent on the size and placement of the grid. Important examples of grid-based clustering algorithms are STING (STatistical INformation Grid) and CLIQUE (CLustering In QUEst) [80].

Model based methods.

This approach is based on the construction of a (statistical) model that explains the data distribution in a given data set as well as possible. The common way to construct such a model is using a mixture of several density functions, containing a density function for each Cluster. Often an additional uniform distributed density function is added to consider noise objects. A further extension determines the most likely number of clusters in the database. Examples of model based clustering algorithms are expectation maximization clustering(EM Clustering)and Self organizing maps [80]. Though model-based clustering algorithms create very expressive models, their computational costs are often very high. Thus, model-based clustering algorithms do not scale well for very large databases.

2.4.2 Clustering of sequential data

Sequential data often comprise sequences of variable length and with many other distinct characteristics, such as dynamic behaviors, time constraints, etc [75, 213]. Sequential data arise from various applications like DNA sequencing in molecular biology and speech processing. Other vital areas of application of sequences include, text mining, medical diagnosis, stock market, customer transactions, web mining and robot sensor analysis [53, 213]. Last decade has seen explosive growth in sequential data [229]. Clustering of sequences or time series involves grouping a collection of sequences (or time series data) based on similarity. Clustering is of

particular interest in sequence data mining since it provides an attractive mechanism to automatically find structure in large data sets that would be otherwise difficult to summarize (or visualize). There are many applications where sequence clustering is relevant. For example, in web activity logs, clusters can indicate navigation patterns of different user groups. In financial data, it would be of interest to group stocks that exhibit similar trends in price movements. Another example could be clustering of biological sequences like proteins or nucleic acids so that sequences within a group have similar functional properties [39, 170].

Several clustering algorithms for discrete sequential data have been proposed. A method for hypergraph-based clustering of transaction data in a high dimensional space was addressed by Han et al. [78]. The method used frequent itemsets when clustering items. Discovered clusters of items were then used to cluster customer transactions. A novel approach to cluster collections of sets and its application to the analysis and mining of sequences of categorical data is presented by Gibson et al. [65]. In the proposed algorithm a type of similarity measure arising from the co-occurrence of values in the data set is used for clustering.

The problem of clustering sequences of complex objects was addressed by Ketterlin [112]. The clustering method presented there used class hierarchies discovered for objects forming sequences in the process of clustering sequences seen as complex objects. The approach assumed applying some traditional clustering algorithm to discover classes of sub-objects, which makes it suitable for sequences of objects described by numerical values, for example trajectories of moving objects.

Sequential data clustering methods could be generally classified into three categories namely, *proximity-based methods*, *feature-based methods* and *model-based methods*.

In the **proximity-based approaches**, the main effort of the clustering process is in devising similarity or distance measure between sequences. With such measure, any proximity-based clustering algorithm, either hierarchical or partitional,is used to group sequences.

Over the years, a number of different approaches have been developed for computing similarity between two sequences [75]. In particular, in the context of comparing biological sequences, e.g. DNA or protein sequences, some of the most

widely used methods first compute an optimal alignment between two sequences (either global or local), and then use either PAM (Point Accepted Mutation) or BLOSUM (BLOcks SUbstitution Matrix) substitution matrices to compute the similarity score of the aligned positions.

The idea behind these approaches is that while comparing two sequences, if a sequence comparison is regarded as a process of transforming a given sequence to another with a task of substitution, insertion and deletion operations then the distance between the two sequences can be defined by the minimum number of required operations to perform it. The most commonly used technique for the process is sequence alignment and the defined distance is known as edit distance or Levenshtein distance [75, 197]. The edit distance may be weighted in order to punish or reward the transformation task based on some prior domain knowledge and the distance here is equivalent to the minimum cost to complete the transformation. Thus, the similarity or distance between two sequences can be redesigned as an optimal alignment problem which can be solved by dynamic programming approach.

The basic sequence alignment algorithm based on dynamic programming approach is the Needleman-Wunsch algorithm ţeDan:texbook. This is a dynamic backtracking algorithm which always produces an optimal alignment. The problem with sequence alignment strategies is that the amount of time and memory that they require tends to grow exponentially with the number of sequences. For two sequences of length n, sequence alignment algorithm would require n^2 units of memory. For three sequences it would require a cube, or 3 dimensional matrix, and the memory requirements would grow to n^3 memory units. Aligning m sequences would require n^m memory units, which is not feasible. Processing time grows at a similar rate.

For low sequence similarity, the alignment quality is not good in the case of basic alignment algorithms [88]. In these algorithms, once an error occurs in the alignment process, that error can never be corrected. Consequently new class of algorithms has been developed. These algorithms iteratively apply dynamic programming to partially aligned sequences to improve their alignment quality. The iteration corrects any errors that may have previously occurred in the alignment

process [75].

Significant amount of literature is available for sequence alignment based on heuristics. These include, genetic algorithms [167], simulated annealing [115] and alignment to a profile HMM [120, 53]. The most popular heuristic strategy involves tree-based progressive alignment [57] in which pairwise alignment steps for groups of sequences are assembled into a complete multiple alignment. As with any hierarchical approach, however, errors at early stages in the alignment not only propagate to the final alignment but also may increase the likelihood of misalignment due to incorrect conservation signals. Post-processing steps such as iterative refinement alleviate some of the errors made during progressive alignment [68].

In 1990 Gotoh first introduced consistency to identify anchor points for reducing the search space of a multiple alignment [69]. Later in 1991 Vingron and Argos [219] gave a mathematically elegant reformulation of consistency in terms of boolean matrix multiplication and implemented in the program MALI, which builds multiple alignments from dot matrices [218].

Another way of measuring similarity is to represent sequences using the vector-space model. In this model, each sequence is considered to be a vector in the 1-gram space. In its simplest form, each sequence is represented by the vector $S = (s_1, s_2, \ldots, s_N)$, where s_j is an indicator whether the j^{th} item is in the sequence. Given this representation, any vector based distance measure can be applied like, cosine, Euclidean and many more.

Although proximity based clustering of sequences is mainly applied in the field of biological sciences but it finds its application in the field of speech recognition [197] as well as in web navigational pattern [83].

The second class of sequence clustering algorithm is **feature-based approaches**. Feature based approaches extract a set of features from each individual data sequence that captures sequential (or temporal) information. All the sequences are then mapped into the transformed feature space, where classical vector space-based clustering algorithms can be used to form clusters. Thus, feature extraction becomes the essential factor that decides the effectiveness of these algorithms. Guralnik and Karypis [74] addressed the problem of potential dependency between

two sequential patterns. They came up with a suggestion that both the global and the local approaches to prune the initial feature sets in order to better represent sequences in the new feature space should be used. Morzy et al. [157] utilized sequential patterns as the basic element in the agglomerative hierarchical clustering and defined a co-occurrence measure as the standard of fusion of smaller clusters. These methods greatly reduce the computational complexities and can be applied to large-scale sequence databases. However, the process of feature selection inevitably causes loss of some information in the original sequences and needs extra attention.

Model-based approaches assume an analytical model for each cluster, and the aim of clustering is to find a set of such models that best fit the data. The first two approaches discussed earlier deal with sequential data composed of alphabets, while the model based approaches aim to construct statistical models that describe the dynamics of each group of sequences and can be applied to numerical or categorical sequences. The most important method is hidden Markov model (HMM) [168, 169], which first gained popularity in speech recognition [194]. It gets its name from two defining properties [64]. First, it assumes that the observation at time t was generated by some process whose state s_t is hidden from the observer. Second, it assumes that the state of this hidden process satisfies the Markov property: that is, given the value of s_{t-1}, the current state s_t is independent of all the states prior to $t-1$. Another assumption of first order HMM is that an observation depends only on its emitting state, not on any previously generated observations, which is known as state-conditional assumption of observation.

HMM are used as cluster prototypes for sequence clustering [125]. The clustering is obtained by employing the rival penalized competitive learning (RPCL) algorithm, a method developed for point clustering, together with a state merging strategy, aimed at finding smaller HMMs.

The approach not directly linked to speech was presented by Smyth [206]. This approach consists of two steps: first, it devises a pairwise distance between observed sequences, by computing a symmetrized similarity. This similarity is obtained by training an HMM for each sequence, so that the log-likelihood of each model, given each sequence, can be computed. This information is used to build an

log-likelihood matrix which is then used to cluster the sequences in to K groups, using a hierarchical algorithm. In the second step, one HMM is trained for each cluster; the resulting K models are then merged into a composite global HMM, where each HMM is used to design a disjoint part of this composite model. This initial estimate is then refined using the standard Baum-Welch procedure. As a result, a global HMM modelling all the data is obtained. The number of clusters is selected using a cross-validation method.

More contributions to the model-based HMM clustering methodology was made by Li and Biswas [136, 134, 137]. They attempted the clustering task by focusing on the model selection and the clustering structure issue. For model selection, they searched for the HMM topology that best represents the data. For clustering structure they focused on finding the most likely number of clusters [136]. Later they used the Bayesian information criterion [200] for model selection. As a clustering criterion they used the sequence-to-HMM likelihood measure to enforce the within-group similarity. The optimal number of clusters is then determined maximizing the partition mutual information (PMI), which is a measure of the inter-cluster distances. They used Bayesian model selection, using BIC [200], and the Cheesman-Stutz (CS) approximation [34] to address the same problem [136]. These clustering methodologies have been applied to specific domains such as physiology, ecology and social science, where the dynamic model structure is not readily available.

An application with a modified HMM model that considers the effect of context for clustering facial display sequences is illustrated by Hoey [92]. Initial problem by pre-grouping the sequences with the agglomerative hierarchical clustering, which operates on the proximity matrix determined by the dynamic time warping (DTW) technique was addressed by Dates et al. [168]. The area formed between one original sequence and a new sequence, generated by warping the time dimension of another original sequence, reflects the similarity of the two sequences. Several objective criterion functions based on posterior probability and information theory for structural selection of HMMs and cluster validity were addressed [135].

Other model-based sequence clustering includes mixtures of first-order Markov chain [207] and a linear model like autoregressive moving average (ARMA) model [228].

Smyth [207] and Cadez et al. [27]further generalize a universal probabilistic framework to model mixed data measurement, which includes both conventional static multivariate vectors and dynamic sequence data.

The paradigm models cluster directly from original data without additional process that may cause information loss. They provide more intuitive ways to capture the dynamics of data and more flexible means to deal with variable length sequences. However, determining the number of model components remains a complicated and uncertain process [206, 168]. Also, the model selected is required to have sufficient complexity, in order to interpret the characteristics of data.

People working with feature based or proximity based approaches generally make use of hierarchical or partitional clustering algorithms. In partitional clustering algorithms centriod based algorithm, $K - Means$ is commonly in use. $K - Means$ algorithm are suitable only for data which are spread in metric spaces (e.g. Euclidean space). If data is spread in the metric space it is possible to compute centroid for a given set of data. Because it is hard to compute centroids in the space of data-sequences, medoid-based approaches are better suited for clustering sequential data sets. Medoid-based methods work with similarity data, that is, data spread in arbitrary similarity space. These techniques try to find representative points (medoids) so as to minimize the sum of the distances of points from their closest medoid. Moreover, none of the clustering algorithm tries to group data sequences such that the sequential nature of the data is preserved while grouping them.

However, in all of the techniques used for sequence clustering none of them consider both the order of occurrence information or the content information simultaneously at a time. Some do consider the content information while some considers the order information while performing the sequence clustering task. In this thesis we used PAM Clustering algorithm with different distance/similarity measures for the classification of sequential data. Commonly used sliding window technique was used for capturing the content information and performing sequence clustering (Chapter 3). Later in chapter 4 we designed a new metric which considers both order of occurrence as well as the content information while computing similarity among two sequences. The designed measure is used with

PAM clustering algorithm in chapter 6 for performing sequence clustering task for web usage mining. Further, in Chapter 6, we presented a modification over PAM clustering algorithm in order to capture the sequential nature of data sequences while grouping them.

2.5 Chapter Summary

A large amount of data is collected every day in the form of sequences. These event sequences can be analyzed not only to search for a particular value or event at a specific time, but also the frequency of certain events, or sets of events related by particular temporal or non temporal relationship. These types of analysis can be very useful for deriving implicit information from the data and for predicting the future behavior of the monitored processes.

Supervised and unsupervised learning from sequential data has many possible applications. In this chapter, we presented a brief survey on the different techniques for sequential data. Classification and clustering task were emphasized as they forms the primary focus of the thesis.

In this thesis the classification task falls under the pattern based classification of sequential data. Clustering task reported in this thesis falls under both proximity based as well as feature based categories.

We have presented the literature survey related to intrusion detection and web usage mining in the chapters 5 and 6 respectively.

CHAPTER III

CLASSIFICATION AND CLUSTERING USING SUBSEQUENCE INFORMATION

We all have dreams. But in order to make dreams come into reality, it takes an awful lot of determination, dedication, self-discipline, and effort. - Jesse Owens

Classification algorithms help in predicting future trends as well as extracting a model of important data classes. Traditionally, sequential data are transformed into non-sequential variables in order to process them. This leads to a loss of sequential information of the data. Although traditional classification algorithms are robust and efficient for modeling non-sequential data, they fail to capture sequential information of the dataset.

The central theme of this chapter is to investigate whether information stored in subsequences (or sequences) plays any role in building a classifier. The same has been extended to clustering i.e., to investigate whether clustering using subsequence information leads to better natural grouping or not. In this chapter, we adopted the kNN classification algorithm for sequential data and explored PAM algorithm for clustering sequential data.

We extracted subsequence information from sequences and used this information for computing various distance/similarity measures. With the appropriate distance/similarity measure, a new sequence is classified using kNN classifier. In order to evaluate the efficiency and behavior of the classifier with the encoded vector measures, Receiver Operating Characteristic (ROC) curve is used. Area under ROC curve (AUC) is also used to understand the results. Experiments are conducted on *DARPA'98 IDS* dataset to show the viability of our model. Sliding window technique has been used for sequence classification [93, 127]. However, it has not been demonstrated that sequence information incorporated using the sliding window approach enables better classification in comparison to the case when no

subsequence information is utilized. So, in this chapter we set to demonstrate explicitly the usefulness of subsequence information for classification and clustering of sequences.

Similar to classification task, we extracted the subsequence information and encoded into frequency vector. The frequency encoded vector data was fed to PAM (Partitioning Around Medoids) clustering algorithm with the various distance/similarity measures. In order to evaluate the goodness of clusters formed, sum-of squared error cluster validation criterion is used. Experiments were conducted on *msnbc* web navigation data to show the viability of the approach.

This chapter is organized as follows. In section 3.1 a brief introduction to sequence pattern mining techniques is presented. In section 3.2, we describe the various distance/similarity measures used in this chapter. For classification and clustering tasks we used Euclidean distance, Jaccard similarity measure, Cosine similarity and Binary Weighted Cosine similarity measure. In section 3.3, we present the methodology adopted for vector encoding of sequences. In section 3.4, *kNN* classification algorithm is given and the techniques used for evaluation of classifier (*ROC* and *AUC*) have been described. In section 3.5, we present experimental results of kNN classification algorithm. In section 3.6, PAM clustering algorithm is given. Sum-of-squared-error measure is used for cluster validation. In the section 3.7, we present the experimental results obtained by considering partial subsequence information with PAM clustering algorithm.

3.1 Introduction

Sequential data are growing at a rapid pace. A pre-defined collection of historical data with their observed nature helps in determining the nature of newly arriving data sequence and hence will be useful in classification of the new data sequence. In data mining, classification algorithms are used for exploring the relationship among various object features at various conditions. Sequence data sets have an additional dimensions such as time and space [223].

Studies on sequential pattern mining mostly concentrate on symbolic patterns [4, 31]. As in symbolic patterns, numerical curve patterns usually belong to the scope

of trend analysis and prediction in statistical time series analysis. Many other parameters also influence the results of sequential pattern mining. These parameters include duration of time sequence (T), event folding window (w) and time interval between two events (int). If we assign w as the whole duration T, we get time independent frequent patterns. An example of such a rule is "In 2005, customers who bought PCs also bought digital cameras". If w is set to be 1, that is, no event sequence folding occurs, then all events are considered to be discrete time events. The rule of the type "Customers who bought hard disk and then memory chip are likely to buy CD-Writer later on" is example of such a case. If w were set to be something between 1 and T, events occurring between sliding windows of specified length would be considered. An example rule is "Sale of PC in the month of April 2005 is maximum". In this chapter, we used *event folding window* method to extract the subsequence patterns for analysis.

Sequence analysis can be categorized into two types, depending on the nature of the treatment. Either we can consider the whole sequence as one or subsequences of different sizes. Our hypothesis is that sequence or order information plays a role in sequence classification/clustering tasks. Hence, in this thesis we considered contiguous subsequences with different sizes.

3.2 Distance/ Similarity measures

Distance or similarity measure plays an important role in classifying or clustering observations in to homogeneous groups.

Below, we briefly discuss various measures such as the Euclidian distance measure, the Jaccard similarity measure, the Cosine similarity measure, and the Binary Weighted Cosine similarity measure (BWC).

3.2.1 Euclidean Distance measure

Euclidean distance is a widely used distance measure for vector spaces [191]. For two vectors X and Y in n- dimensional Euclidean space, it is defined as the square

root of the sum of difference of the corresponding dimensions of the vector. Mathematically, it is given as

$$EuclideanDistance(X,Y) = \left[\sum_{s=1}^{n}(X_s - Y_s)^2\right]^{1/2} \quad (1)$$

Euclidean distance is highly sensitive to outliers. Direct application of Euclidean distance measure is not possible across sets. Sets are first converted into n-dimensional vector space. Over these transformed vectors euclidean distance measure is applied to find the distance between the two. For two sets, X = $\langle$ M, N, P, Q, R, M, S, Q$\rangle$ and Y = $\langle$P, M, N, Q, M, P, P$\rangle$ the equivalent transformed frequency vector is $X_v =<$ 2,1,1,2,1,1$>$ and $Y_v =<$ 2,1,3,1,0,0 $>$ where the frequency of the components M, N, P, Q, R and S are written respectively. Here, the Euclidean distance between the transformed vectors X_s and Y_s is 2.64.

3.2.2 Jaccard Similarity measure

Jaccard similarity function is used for measuring similarity between binary values [66]. It is defined as the degree of commonality between two sets. It is measured as the ratio of number of common elements of two sets X and Y to the total number of elements contained in sets X or Y. Mathematically, it is given as follows. If X and Y are two distinct sets then the similarity between X and Y is

$$JaccardSim(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (2)$$

Consider two sets $X = \langle M, N, P, Q, R, M, S, Q\rangle$ and $Y = \langle P, M, N, Q, M, P, P\rangle$. $X \cap Y$ is given as $\langle M, N, P, Q\rangle$ and $X \cup Y$ is $\langle M, N, P, Q, R, S\rangle$. Thus, the similarity between X and Y is 0.66.

3.2.3 Cosine Similarity measure

Cosine similarity is a common vector based similarity measure. Cosine similarity measure is commonly used in text databases [191]. Cosine similarity measure calculates the angle of difference between the two vector directions, irrespective of their lengths. Cosine similarity between two vectors X and Y is given by

$$CosineSimilarity(X,Y) = \frac{X \bullet Y}{|X||Y|} \quad (3)$$

where, $X \bullet Y$ represents the dot product of vectors X and Y. $|X|$ and $|Y|$ represent the size of vectors X and Y, calculated as for example, $|X| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$.

Similar, to the Euclidean distance measure, application of cosine similarity measure on sets is not possible. So, sets are transformed into vectors before applying cosine measure. For two sets, X = $\langle$ M, N, P, Q, R, M, S, Q$\rangle$ and Y = $\langle$P, M, N, Q, M, P, P$\rangle$ the equivalent transformed frequency vectors are $X_v =<$ 2,1,1,2,1,1$>$ and $Y_v =<$ 2,1,3,1,0,0 $>$ respectively. The cosine similarity of the transformed vector is 0.745.

3.2.4 Binary Weighted Cosine (BWC) Similarity measure

Rawat et al. [195] proposed binary weighted cosine similarity measure for measuring similarity across sequences. They showed the effectiveness of the proposed measure in the Intrusion detection. They applied kNN classification algorithm with BWC measure to enhance the capability of the classifier. This similarity measure considers both the number of shared elements between two sets as well as frequencies of those elements. The similarity measure between two sequences X and Y is given by

$$BinaryWeightedCosine(X,Y) = \frac{|X_b \wedge Y_b|}{|X_b \vee Y_b|} \times \frac{X \bullet Y}{|X||Y|} \tag{4}$$

where, X_b, Y_b denotes binary cosine indicating the absence or presence of corresponding features. BWC measure captures the frequency and commonality of system calls to calculate similarity. Cosine similarity measure is a contributing component in a BWC similarity measure. Hence, BWC similarity measure is also a vector based similarity measure. The transformation step as carried out in cosine similarity measure or Euclidean measure for sets is performed here also. For two sets, X =$\langle$ M, N, P, Q, R, M, S, Q$\rangle$ and Y = $\langle$P, M, N, Q, M, P, P$\rangle$, the cosine similarity is given as 0.745 and Jaccard similarity as 0.66. Hence, the BWC similarity measure comes out to be 0.49.

3.3 Vector encoding of sequences

In data mining, objects of interest are frequently encoded in the form of feature/attribute vectors. For example, documents are given as sequences of words, computer sessions as a sequence of system calls, web sessions as a sequence of web page visits,

etc. Therefore, to be able to perform clustering on these sequences, an appropriate representation for sequences is needed. The most popular method is to represent sequences as vectors in multidimensional space. Each dimension is equivalent to a distinct subsequence from the sequence collection. Due to the size of alphabet from which these sequences are being constructed, the number of distinct subsequences can be extremely large. The unique subsequences also depends on the size of symbols being considered for the formation of subsequences. Computation in that high-dimensional space is prohibitively expensive and sometimes even impossible because of memory size restrictions.

In this section we illustrate the methodology of extracting sequential information from sets, thus making useful for vector based distance/similarity measures. We considered subsequences of fixed sizes: $1, 2, 3, \ldots$. This fixed size subsequence is called a *window*. This window is slided over the sequence to find unique subsequences of a fixed length over the whole sequence. Frequency count of each subsequence is recorded. Consider a sequence which consists of traces of system calls as shown in Figure 3.

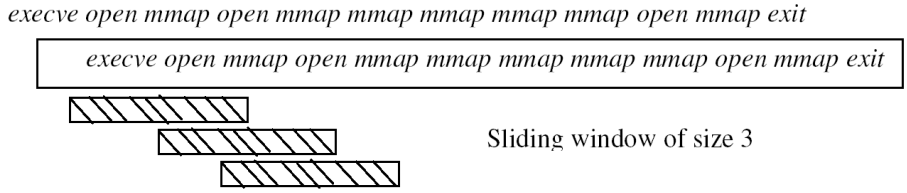

Figure 3: Sequence of system call trace

For a total length of sequence of 12 and with a sliding window size w of 3, we will have $(12 - 3 + 1 =)10$ subsequences of size 3. These 10 subsequences of size 3 are given in table 1. From among these 10, unique subsequences and their frequencies are shown in Table 2.

Table 1: Subsequence of system calls of size 3 of the example system call trace in Figure 3

execve open mmap	open mmap open	mmap open mmap	open mmap mmap
mmap mmap mmap	mmap mmap mmap	mmap mmap mmap	mmap mmap open
mmap open mmap	open mmap exit		

With these encoded frequencies for subsequences, we can apply any vector

Table 2: Sequence of system calls of size 3 with their frequencies

Sequence	frequency	Sequence	frequency
execve open mmap	1	mmap open mmap	2
open mmap open	1	mmap mmap open	1
open mmap mmap	1	open mmap exit	1
mmap mmap mmap	3		

based distance/similarity measure, thus incorporating partial sequential information with vector space representation.

3.4 kNN Classification

Many classification algorithms have been proposed by researchers in machine learning [153], expert systems [225], statistics [103] and neural networks [145]. Classification algorithms have been successfully applied to the problems, where the dependent variable (class variable) depends on independent (explanatory) variables. Typical classification techniques are Support Vector Machines, Decision Trees, Bayesian Classification, Neural Networks and k-Nearest Neighbor (kNN).

kNN classification algorithm does not build a classifier in advance. Whenever a new data record comes, kNN finds the k nearest neighbors to new data stream from training data set using some distance/similarity measure [43, 66]. kNN is the best choice for making a good classifier, when simplicity and accuracy are important issues [113].

kNN classifier is based on learning by analogy. It assumes that all instances correspond to points in an n-dimensional space. Nearest neighbors of an instance are described by a distance/similarity measure. When a new sample comes, kNN classifier searches the training dataset for the k closest samples to the new sample using distance/similarity measure for determining the nature of new sample. These k samples are known as the k nearest neighbors of the new sample. The new sample is assigned the most common class of its k nearest neighbors. k-Nearest neighbor algorithm is summarized as shown in algorithm 1.

In the nearest neighbor model, choice of a suitable distance (or similarity) function and the value of the members of k nearest neighbors are very crucial. Higher the value of k the more is the complexity of the nearest neighbor model. The model

Algorithm 1 Algorithm for k-Nearest Neighbor

Input:
D = Dataset of N instances
k = Number of nearest neighbors
$newinst$ = New sample (Class to be found)
S = Function for similarity measure
Output:
Class label of instance $newinst$
Begin
Training
Construct Training sample T from the given dataset D.
Classification
Given a new sample *newinst* to be classified
Compute the similarity of $newinst$ with all instances of training set Tusing function S
Let $I_1, \ldots, I_k$ denote the k instances from T that are nearest to new sample $newinst$
Find the majority class label from the class of k nearest neighbor samples.
Returned majority class label is the class label of new sample.
End

is less adaptive with higher *k* values [80].

In kNN algorithm all the k neighbors of the new data record may not contribute equally to decide the nature of new data record. The data which are similar to the new data record should be given relatively more importance. Hence, an improvement to kNN algorithm came that is to weight the contribution of each of the k neighbors according to their distance to the new data records, giving greater weight to closer neighbors and relatively lesser weight to farther neighbor. This modified algorithm is called *distance-weighted k-nearest neighbor* algorithm [52].

Since, different measures will yield different neighbors, the choice of distance measure is very important in kNN classification algorithm.

3.4.1 Receiver Operating Characteristic Curve

In a two class problem, each instance I, is mapped to either a positive class or a negative class. The classification problem can be stated as mapping of instances to the predicted class. For a given instance and the predicted class there are four possibilities.

- If the instance is positive and is classified as positive, then it is called **true**

positive (TP).

- If the instance is negative and is classified as negative, it is termed as **true negative** (TN).
- If the instance is positive and is classified as negative then it is called **false negative** (FN).
- If the instance is negative and is classified as positive, it is termed as **false positive** (FP).

In the context of intrusion detection, true positive rate is commonly referred to as Detection Rate (DR) that is, DR is the ratio of the number of intrusive sessions (abnormal) detected correctly to the total number of intrusive sessions. On the other hand, the FPR is defined as the number of normal sessions detected as intrusive, divided by the total number of normal sessions. Receiver Operating Characteristics (ROC) curve gives an idea of the trade off between FPR and DR achieved by a classifier. An ideal ROC curve would be parallel to FPR axis at DR equal to 1.

A ROC curve provides a graphical representation of the relationship between the true-positive and false-positive prediction rate of a model. The y-axis corresponds to the *sensitivity* of the model, i.e. how well the model is able to predict true positives, and the y-coordinates are calculated as:

$$y = \frac{TP}{TP + FN} \tag{5}$$

The x-axis corresponds to the *specificity*, i.e. the ability of the model to identify true negatives. An increase in specificity (i.e. a decrease along the X-axis) results in an increase in sensitivity. The x-coordinates are calculated as:

$$x = 1 - \frac{TN}{TN + FN} \tag{6}$$

The greater the sensitivity at high specificity values (in other words, high y-axis values at low X-axis values) the better the model.

A numerical measure of the accuracy of the model can be obtained from the area under the curve, where an area of 1.0 signifies near perfect accuracy, while an area of less than 0.5 indicates that the model is worse than just random. The quantitative-qualitative relationship between area and accuracy follows a fairly linear pattern.

3.4.2 Area under curve of ROC

To compare classifiers we may want to reduce ROC performance to a single scalar value representing expected performance. A common method is to calculate the area under the ROC curve, abbreviated as AUC [22, 82]. Since the AUC is a portion of the area of the unit square, its value will always be between 0 and 1.0. However, because random guessing produces the diagonal line between (0,0) and (1,1), which has an area of 0.5, no realistic classifier should have an AUC less than 0.5. Due to its simplicity it finds its application in various fields such as speech recognition, medical diagnosis, and so on. Figure 4 shows the sample Area under ROC curve for two hypothetical classifiers A and B. As the area enclosed by classifier B is more than the area enclosed by classifier A, the performance of classifier B is better than A.

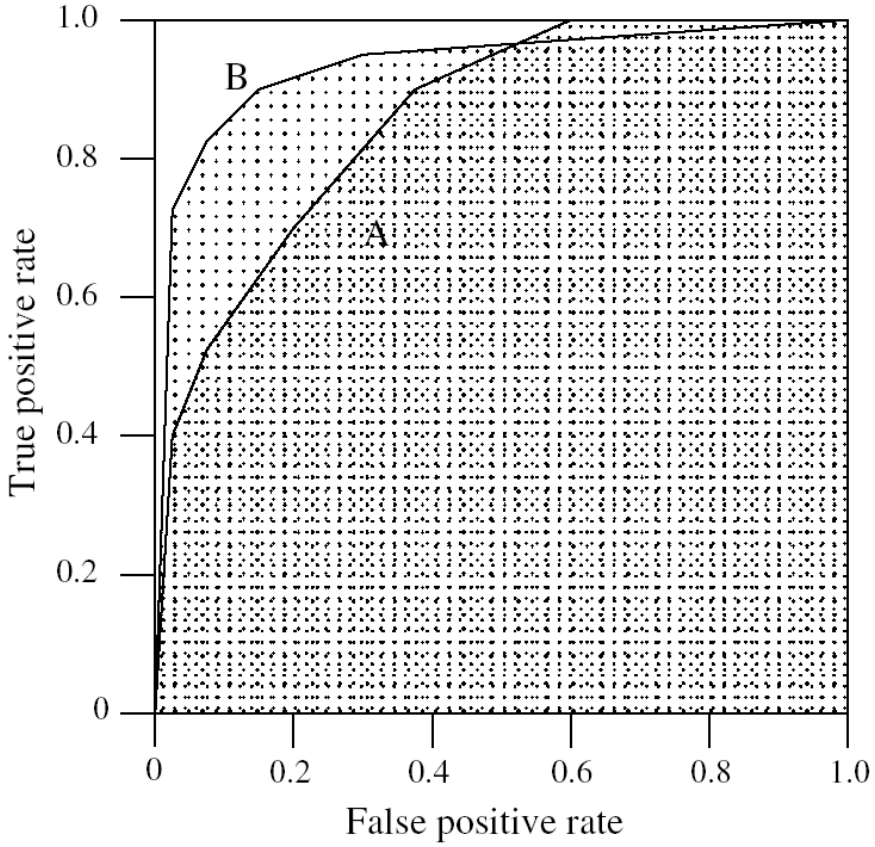

Figure 4: AUC for two classifiers A and B

The AUC has an important statistical property in that it is equivalent to the

probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

To find the AUC, calculate the trapezoidal area under each vertical column drawn due to unequally distributed points of ROC curve having a straight-line segment as its top; then sum all the individual areas.

All ROC curves are non decreasing. For smooth ROC curve the trapezoidal will underestimate the area under the ROC curve. However, if there are many threshold points in the ROC curve and they are spaced rather closely, the discrepancy will not be large. If the trapezoidal area is in fact too small, smoothing can be applied to the ROC curves, before the area covered by the curve is calculated. However, the simple nature of the trapezoidal AUC may be computationally efficient.

3.5 Classification using subsequence information

Detecting and using subsequence pattern information from a sequential dataset plays a significant role in many areas including intrusion detection, monitoring for malicious activities and molecular biology [122, 182, 183, 226]. An observed pattern may be significant or not depends on its likelihood of occurrence. In order to detect real intrusion as well as to avoid false alarms simultaneously, a proper choice of threshold and length of subsequence must be done. Keeping threshold value too low may lead for high false alarms and too high threshold values may allow many attacks undetected. A compromise in threshold value may allow an appropriate size for the sliding window.

The traditional kNN classification algorithm [43, 80] with suitable distance/similarity measure can be used to build an efficient classifier. An approach based on *kNN classifier* is proposed by Liao and Vemuri [139]. The proposed methodology consists of two phases namely training and testing phase. Dataset D consists of m sessions. Each session is of variable length. Initially in training phase, all the unique subsequences of size s are extracted from the whole dataset. Let n be the number of unique subsequences of size w, generated from the dataset D. A matrix C of size $m \times n$ is constructed where C_{ij} reflects the count of j^{th} unique subsequence in the i^{th} session. A distance/similarity measure is constructed by applying distance/similarity measure over the C matrix. The model is trained with the dataset

consisting of normal sessions.

In testing phase, whenever a new session P comes to the classifier, the algorithm looks first for the presence of any new subsequence of size s. If a new subsequence is found, the new session is marked as abnormal. When there is no new subsequence in new session P, the similarity of the new session is computed with respect to all the sessions. If similarity between any session in training set and new session is equal to 1, then it is marked as normal. Otherwise, k sessions that are of similar to the new session P in the training dataset are gathered. From these k maximum values, the average similarity for k-nearest neighbors is calculated. If the average similarity value is greater than user defined threshold value (τ) the new session P is marked as normal, else it is marked as abnormal. Algorithm 2 outlines the methodology in detail.

3.5.1 Experimental results

We implemented our approach in Java 1.4 and performed experiments on 2.4 GHz, 256 MB RAM, Pentium-IV machine running on Microsoft Windows XP 2002. Experiments were conducted using *k-Nearest Neighbor* classifier with Euclidean distance, Jaccard similarity, Cosine similarity and BWC similarity measures. Each distance/similarity measure was individually experimented with *kNN* classifier on *DARPA'98 IDS* dataset.

DARPA'98 IDS dataset consists of *TCPDUMP* and *BSM* audit data. The network traffic of an Air Force Local Area Network was simulated to collect *TCPDUMP* and *BSM* audit data. The audit logs contain seven weeks of training data and two weeks of testing data. There were 38 types of network-based attacks and several realistic intrusion scenarios conducted in the midst of normal background data. Detailed discussion of DARPA dataset is given at in Appendix A. For experimental purpose, 605 unique processes were used as a training dataset, which were free from any type of attack. Testing was conducted on 5285 normal processes. In order to test the detection capability of the proposed approach, we incorporated 55 intrusive sessions into our test data. For *kNN* classification experiments, $k = 5$ was considered. Experiments were carried out with various distance/similarity

Algorithm 2 k-Nearest Neighbor Algorithm for classification of system calls using subsequences

Input:
S = Set of sessions
k = Number of nearest neighbors
P = New session (Class to be determined)
L = Sliding window size
Sim = Function for similarity measure
τ = User specified similarity threshold
Output:
Class label of session P
Begin
Training
Construct Training sample T from S consisting of normal sessions.
Classification
Extract unique subsequences of window size L and store in database D
For session P
Extract all the unique subsequences from P of window size L
If P contains any subsequence $\notin D$
Mark as abnormal
Exit
Else
For session P
do
Calculate $Sim(P, T_j)$ /*T_j is the j^{th} session of test set T, $1 \leq j \leq |T|$.*/
Calculate average similarity, AvgSim, for k nearest neighbors
If (AvgSim $> \tau$)
Mark P as normal
Else
Mark P as abnormal
End For
End

measures as discussed in section 3.2 (Euclidean distance measure, Jaccard similarity measure, Cosine similarity measure and BWC similarity measure) at different subsequence lengths (sliding window size L) . Here, $L = 1$ means that no sequential information is captured whereas, for $L > 1$ some amount of order information across elements of the data is preserved. To analyze the efficiency of classifier, ROC curve is used.

Tables 3, 4, 5 and 6 show the FPR and DR for euclidean, Jaccard, cosine and BWC measures respectively. The FPR vs DR rate has been shown for various threshold values for different sliding window sizes varying from 1 to 5. For the

experimentation purpose the value for *k* in kNN classification algorithm was fixed at 5. Corresponding ROC curves for Euclidean distance, Jaccard similarity, Cosine similarity and BWC similarity measures are shown in Figures 5, 6, 7 and 8, respectively. It can be observed from Fig 5, 6, 7, and 8 that as the sliding window size increases from L =1 to L = 5, high DR (close to ideal value of 1) is observed with all the distance/similarity measures. This indicates the usefulness of incorporating subsequence information.

Rate of increase in false positive is less for Jaccard similarity measure (0.005-0.015) as compared to the other distance/similarity measures such as Cosine (0.1-0.4), Euclidian (0.05-0.15) and BWC (0.1-0.7) similarity measures. Thus, we have less false alarms generated in the case of Jaccard similarity measure in comparison to other distance/similarity measures under consideration in this thesis.

Table 7 depicts the factor (FPR or Threshold value) that was traded off in order to achieve high DR. For example, in the case of Jaccard similarity measure, there seems to be tradeoff between FPR and threshold value (highlighted in bold face) in order to achieve high DR.

Thus, our results support the hypothesis that classification accuracy of sequential data can be improved by incorporating the order information embedded in sequences. We also performed experiments with different k values for the nearest neighbor classifier with all the four measures.

We summarized data corresponding to the false positive rate at maximum attained detection rate for different subsequence lengths L = 1, 2, 3, 4, 5 across all the distance/similarity measures in Table 8 for k =5. It can be observed that, as per the trend, the FPR is increasing with increasing subsequence lengths for all the four measures. We also observed that as the subsequence length is increased a low false positive value at early threshold values were obtained irrespective of all the four distance/ similarity measure used. We conducted the experiments upto subsequence length of 11 and similar trend was observed. However, for higher subsequence length values ($L > 6$) the detection rate to the false positive rate was saturated.

Table 3: FPR vs DR for Euclidian Distance measure at K=5

Th	SL =1		SL =2		SL =3		SL=4		SL =5	
	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR
0.99	0.0052	0.7272	0.0087	0.9272	0.0089	0.98181	0.0107	0.9636	0.0121	0.9636
0.98	0.0026	0.3454	0.0066	0.9272	0.0073	0.9636	0.0094	0.9636	0.0111	0.9636
0.97	0.0026	0.3454	0.0056	0.9272	0.0064	0.9636	0.0085	0.9636	0.01021	0.9636
0.95	0.0017	0.3454	0.0056	0.9272	0.0064	0.9636	0.0085	0.9636	0.01021	0.9636
0.93	0.0015	0.3454	0.0055	0.9272	0.0064	0.9636	0.0083	0.9636	0.01021	0.9636
0.9	0.0015	0.3454	0.0055	0.9272	0.0062	0.96363	0.0083	0.9636	0.01002	0.9636
0.89	0.0015	0.3454	0.0055	0.9272	0.0062	0.96363	0.0083	0.9636	0.01002	0.9636
0.86	0.0015	0.3454	0.0055	0.9272	0.0062	0.96363	0.0083	0.9636	0.01002	0.9636
0.84	0.0002	0.3454	0.0042	0.9272	0.0049	0.96363	0.007	0.9636	0.00851	0.9636
0.8	0	0.3454	0.004	0.9272	0.0047	0.96363	0.0068	0.9636	0.00851	0.9636
0.78	0	0.3454	0.004	0.9272	0.0047	0.96363	0.0068	0.9636	0.00851	0.9636
0.75	0	0.3454	0.004	0.9272	0.0047	0.96363	0.0068	0.9636	0.00851	0.9636
0.7	0	0.3454	0.004	0.9272	0.0047	0.96363	0.0068	0.9636	0.00851	0.9636

Table 4: FPR vs DR for Jaccard similarity measure at K=5

Th	SL =1		SL =2		SL =3		SL=4		SL =5	
	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR
0.99	0.0083	0.9454	0.01324	0.9818	0.0132	1	0.0138	0.9818	0.0132	1
0.98	0.0083	0.9454	0.0132	0.9818	0.0132	1	0.0138	0.9818	0.0132	1
0.97	0.0083	0.9454	0.0111	0.9818	0.0132	1	0.0138	0.9818	0.0132	1
0.95	0.0083	0.9454	0.0100	0.9636	0.0111	1	0.0138	0.9818	0.0111	1
0.93	0.0083	0.9454	0.0100	0.9636	0.01	0.9818	0.0117	0.9818	0.01	1
0.9	0.0081	0.9454	0.0100	0.9636	0.01	0.9818	0.0102	0.9636	0.01	1
0.89	0.0081	0.9454	0.0100	0.9636	0.01	0.9818	0.0102	0.9636	0.01	1
0.88	0.0071	0.9454	0.0100	0.9636	0.01	0.9818	0.0102	0.9636	0.01	0.9818
0.87	0.006	0.9454	0.0100	0.9636	0.01	0.9818	0.0102	0.9636	0.01	0.9818
0.86	0.0058	0.9454	0.0100	0.9636	0.01	0.9818	0.0102	0.9636	0.01	0.9818
0.84	0.0056	0.9454	0.0096	0.9636	0.01	0.9818	0.0102	0.9636	0.01	0.9818
0.8	0.0045	0.6181	0.0090	0.9636	0.0098	0.9818	0.0100	0.9636	0.0098	0.9818
0.78	0.0041	0.5454	0.0083	0.9636	0.0098	0.9818	0.0100	0.9636	0.0098	0.9818
0.75	0.002	0.4909	0.0079	0.9636	0.0092	0.9818	0.0100	0.9636	0.0092	0.9818
0.7	0.0003	0.4181	0.0054	0.9636	0.0088	0.9818	0.0098	0.9636	0.0088	0.9818
0.65	0.0001	0.3636	0.00491	0.9454	0.0064	0.9818	0.0088	0.9636	0.0064	0.9818
0.6	0	0.3454	0.0047	0.9272	0.0049	0.9636	0.0079	0.9636	0.0049	0.9818
0.55	0	0.3454	0.0047	0.9272	0.0049	0.9636	0.0070	0.9636	0.0049	0.9818
0.52	0	0.3454	0.0041	0.9272	0.0049	0.9636	0.0068	0.9636	0.0049	0.9818

Table 5: FPR vs DR for cosine Similarity measure at K=5

Th	SL =1		SL =2		SL =3		SL=4		SL =5	
	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR
0.99	0.051	0.963	0.079	0.981	0.129	1	0.470	1	0.364	1
0.98	0.010	0.945	0.053	0.963	0.125	0.981	0.457	0.981	0.326	1
0.97	0.003	0.909	0.038	0.963	0.097	0.981	0.441	0.981	0.325	1
0.95	0.002	0.818	0.022	0.963	0.084	0.981	0.237	0.963	0.319	1
0.93	0.002	0.818	0.019	0.963	0.047	0.981	0.049	0.963	0.098	1
0.90	0.0003	0.763	0.007	0.963	0.037	0.981	0.032	0.963	0.085	1
0.89	0.0003	0.763	0.007	0.963	0.037	0.981	0.025	0.963	0.055	1
0.88	0.0003	0.763	0.006	0.963	0.030	0.981	0.018	0.963	0.055	1
0.87	0	0.763	0.006	0.963	0.029	0.981	0.017	0.963	0.054	1
0.86	0	0.745	0.005	0.963	0.004	0.963	0.015	0.963	0.054	1
0.84	0	0.363	0.004	0.963	0.004	0.963	0.014	0.963	0.052	1
0.80	0	0.363	0.004	0.963	0.004	0.963	0.014	0.963	0.051	1
0.78	0	0.363	0.003	0.963	0.004	0.963	0.007	0.963	0.038	1
0.75	0	0.345	0.003	0.945	0.004	0.963	0.007	0.963	0.037	1
0.7	0	0.345	0.003	0.945	0.004	0.963	0.007	0.963	0.014	0.981

Table 6: FPR vs DR for BWC Similarity measure at K=5

Th	SL =1		SL =2		SL =3		SL=4		SL =5	
	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR
0.99	0.3793	1	0.1659	1	0.2044	1	0.5298	1	0.4072	1
0.98	0.3258	1	0.1476	1	0.2023	1	0.5184	1	0.4026	1
0.97	0.3063	1	0.1429	1	0.1919	1	0.5171	1	0.4011	1
0.95	0.3033	1	0.1383	1	0.1875	1	0.5101	1	0.3752	1
0.93	0.2834	1	0.1353	1	0.1579	1	0.5052	1	0.3707	1
0.9	0.0886	1	0.0956	1	0.1549	1	0.5016	1	0.3665	1
0.89	0.0811	1	0.0948	1	0.6259	1	0.5016	1	0.6547	1
0.86	0.0741	0.9636	0.0816	1	0.5975	1	0.4772	1	0.6426	1
0.84	0.0498	0.9636	0.0689	0.9818	0.5639	1	0.4768	1	0.6412	1
0.8	0.0475	0.9636	0.0414	0.9818	0.3381	1	0.4418	1	0.5981	1
0.78	0.0469	0.9636	0.0394	0.9818	0.3307	1	0.2475	1	0.5412	1
0.75	0.0227	0.909	0.0369	0.9818	0.3226	1	0.2433	1	0.3448	1
0.7	0.01	0.8727	0.0333	0.9818	0.2857	1	0.2301	1	0.3383	1
0.65	0.0038	0.5272	0.0303	0.9818	0.1027	0.9818	0.0462	0.9818	0.302	1
0.6	0	0.3818	0.0233	0.9636	0.0675	0.9818	0.042	0.9818	0.109	0.9818

Table 7: Comparative analysis of different distance/similarity measures

	Jaccard similarity measure		Cosine similarity measure		Euclidian distance measure		BWC similarity measure	
	th value	**FPR**	**th value**	FPR	**th value**	FPR	th value	**FPR**
L =1	0.84	**0.0056**	**0.99**	0.051	**0.99**	0.0052	0.89	**0.0811**
L =2	0.97	**0.0111**	**0.99**	0.079	**0.52**	0.003	0.86	**0.0815**
L =3	0.95	**0.011**	**0.99**	0.129	**0.99**	0.0088	0.7	**0.285**
L =4	0.93	**0.0117**	**0.99**	0.470	**0.52**	0.006	0.7	**0.2300**
L =5	0.89	**0.0105**	**0.75**	0.037	**0.99**	0.06	0.65	**0.302**

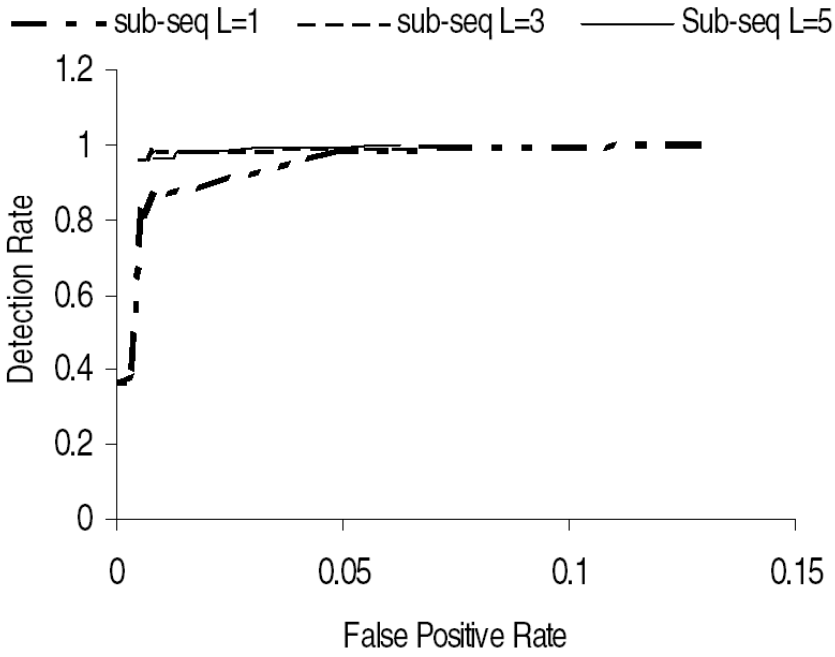

Figure 5: ROC curve for Euclidian distance measure using kNN classification for k =5

AUC is an another approach to evalute the performance of the classifier. Table 9 shows the Area under ROC curve for all the four distance/similarity measures used in this chapter. The higher the area the better is the classifier. As can be observed from the table the Jaccard measure resulted in the highest area among all the four measures for the same subsequence length. However, cosine measure performed better than both the Jaccard and BWC similarity measures for subsequence length 1. Subsequence length of 1 means that there is no sequential information

Table 8: False positive rate at maximum attained detection rate for different subsequence length for different distance/similarity measure at k =5

	Jaccard similarity measure	Cosine similarity measure	Euclidian distance measure	BWC similarity measure
	FPR	FPR	FPR	FPR
L =1	0.0083	0.051	0.0052	0.3793
L =2	0.013245033	0.0794	0.0087	0.1659
L =3	0.0132	0.129	0.00889	0.20435
L =4	0.013812677	0.47076	0.0107	0.5298
L =5	0.0132	0.3648	0.0121	0.4071

Table 9: AUC for different Similarity/Distance measure for different subsequence length for K=5

Distance/ Similarity Measure	**SL =1**	**SL =2**	**SL =3**	**SL=4**	**SL =5**
Cosine Similarity Measure	0.987684	0.963095	0.995943	0.983943	0.992649
Jaccard Similarity Measure	0.970601	0.988575	0.9971377	0.988917	0.996393
Euclidean Distance Measure	0.861402	0.961429	0.968407	0.978329	0.97948
BWC Similarity Measure	0.97447408	0.89056376	0.96438996	0.97417882	0.94179448

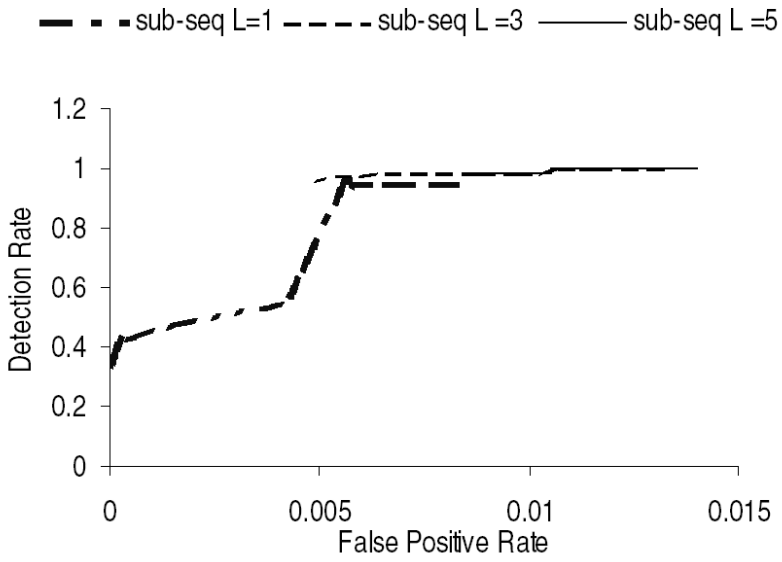

Figure 6: ROC curve for Jaccard similarity measure using kNN classification for k =5

captured. Capturing no sequential information has attracted people when making kNN classifier for intrusion detection to use cosine similarity measure. However, as the subsequence length is increased the classifier with Jaccard measure performed better than others.

The idea of looking at short sequences of behavior is quite general and might be applicable to several other security problems. For example, people have suggested applying the technique to several other computational systems [93]. Our approach is similar to other approaches. Teng et al. [215] considered sequencing information. Their approach differ from our approach that they look at the domain of user behavior, and use a probabilitistic approach for detecting anomalies. Because our results are sufficiently promising, the added complexity of using probabilities seems unnecessary. It is possible that our simple deterministic approach is successful because our data is well-structured. If this is the case, it may well be that probabilities are necessary in less structured domains, such as user behavior. Hofmeyr et al. [93], commented that the window length size of at least 6 is required to detect the intrusion. Later Tan and Maxion [214], argued and gave explanation on why the length of window size be considered as 6 to detect the anomalous behavior. The experiments in this chapter adhere to the results of Tan and Maxion [214] work for *DARPA'98 IDS* dataset. In our experiments it was observed that

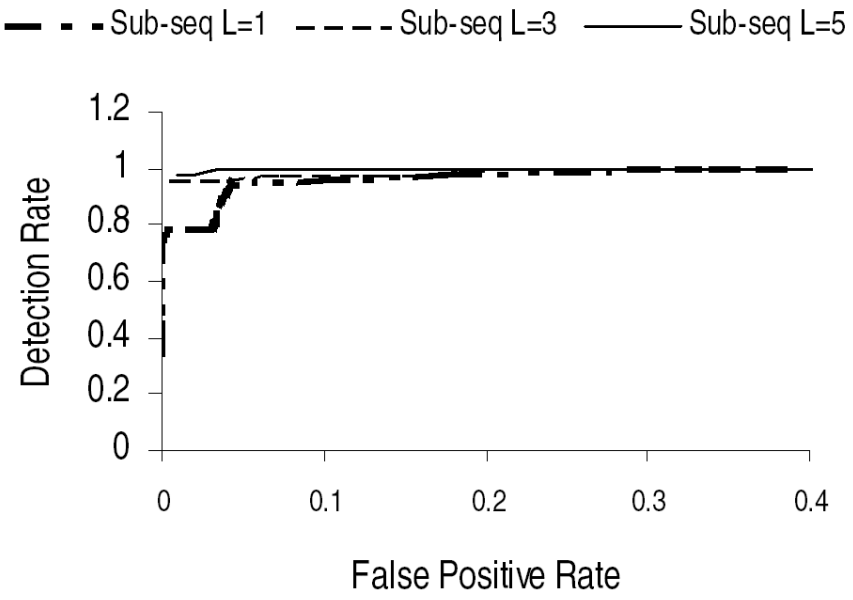

Figure 7: ROC curve for Cosine similarity measure using kNN classification for k =5

the window length of 5 is still enough to detect the intrusive sessions.

With the above observed results for classification task in the field of intrusion detection we have an intuition that the same promising results may be achieved when performing clustering task. In order to test our hypothesis, we conducted experiments using PAM clustering algorithm with varying subsequence lengths. Benchmarking $msnbc$ web log dataset is used for the experimentation. The same encoding scheme described in Section 3.3 for converting sequences to vectors was adopted for the clustering task. Sum of squared error was used to find the validity of clusters.

3.6 PAM Clustering

Clustering is an established and widely known technique for grouping data. It has been recognized and found successful applications in various areas like data mining, statistics and information retrieval.

Let D = $\{d_1, d_2, \ldots, d_n\}$ be a set of objects, and Sim(d_i, d_j) denote a similarity measure between objects d_i and d_j . Clustering then can be defined as a task of finding the decomposition of D into K clusters $C = \{c_1, c_2, \ldots, c_k\}$ so that each object is assigned to a cluster and the ones belonging to the same cluster are similar to

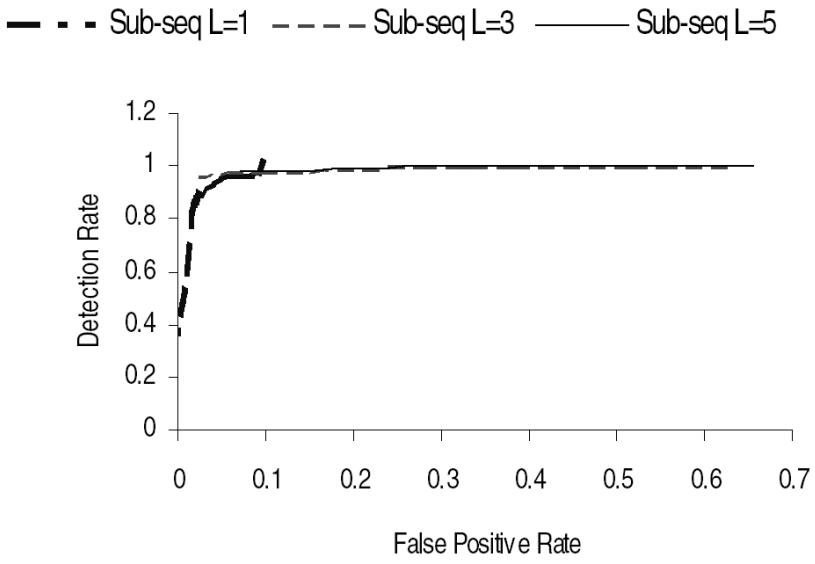

Figure 8: ROC curve for BWC similarity measure using kNN classification for k =5

each other (with respect to the similarity measure) and are as dissimilar as possible to objects from the other clusters. There are numerous clustering algorithms ranging from vector-space based, model-based (mixture resolving) to graph-theoretic approaches.

Partitioning Around Medoid (PAM) is a variation of k-means based clustering algorithm. It falls under the category of partitional algorithm. It works differently than the well known k-Means partitional clustering algorithm [80]. PAM uses the principle of center of gravity. By the use of centre of gravity, the affects of large outliers is minimized that exist in k-Means algorithm. In PAM, we take the most centrally located object in a cluster as the center. This is called the medoid. The method works similar to k-Means except that the data points are placed in a cluster based on how similar or close they are to the medoid of a particular cluster. Every time a data object is added to a cluster, a new medoid is selected for the new cluster based upon four cost functions (described below) which ensure that the total quality of the resulting cluster is continuously improved.

To determine whether a non-medoid object, $O_{nonmediod}$ is a good replacement for the currently selected medoid, o_j, the following four cases are examined for each of the non-medoid objects, p,

- **Case 1**: p currently belongs to o_j: If o_j is replaced by $O_{nonmediod}$ as a medoid and p is closer to O_i, where $i \neq j$ then p is assigned to o_i.
- **Case 2**: p currently belongs to o_j: If o_j is replaced by $O_{nonmediod}$ as a medoid and p is closer to $O_{nonmediod}$, then p is assigned to $O_{nonmediod}$.
- **Case 3**: p currently belongs to medoid o_i where $i \neq j$: If o_j is replaced by $O_{nonmediod}$ as a medoid and p is still closer to the O_i, then medoid is not changed.
- **Case 4**: p currently belongs to medoid o_i, where $i \neq j$: If o_j is replaced by $O_{nonmediod}$ as a medoid and p is closer to $O_{nonmediod}$, then p is assigned to the $O_{nonmediod}$.

The medoid approach makes the clustering algorithm more adaptable to noise and outliers. At the same time it is more expensive computationally than k-Means. Similar to k-Means, in the PAM the user has to specify the value of k, even though user may not necessarily know how many clusters should be formed.

PAM clustering algorithm places more emphasis on quality of clusters. Algorithm 3 shows the PAM clustering algorithm.

Algorithm 3 PAM Clustering Algorithm

Input:
D = Dataset of N objects
k = Number of Clusters
Sim = Function for similarity/Distance measure
Output:
A set of k clusters
Begin
Compute the similarity matrix for database D using given similarity/distance using function Sim
Randomly choose the k objects as initial set of medoids.

repeat
Assign each non-selected object to the cluster with the nearest medoid.
Randomly select a non medoid object, mark it $O_{nonmedoid}$.
Calculate the total cost, C, for swapping o_j with $O_{nonmedoid}$, where O_j belongs to the cluster representatives of the current medoid.
If $C < 0$, then swap o_j with $o_{nonmedoid}$ to form a new set of medoid.
until (there is no change in cost C)
End

3.6.1 Sum of squared error

There are two main issues in clustering techniques. Firstly, finding the optimal number of clusters in a given dataset and secondly, given two sets of clusters, computing a relative measure of goodness between them. For both these purposes, a criterion function or a validation function is usually applied. The simplest and most widely used cluster optimization function is the sum of squared error [51]. Studies on the sum of squared error clustering were focused on the well-known k-Means algorithm [58, 105] and its variants [101]. The sum of squared error (SSE) is given by the following formula, where $\overset{\wedge}{t_j}$ is the cluster center of j^{th} cluster, t_{j_s} is the s^{th} member of j^{th} cluster, $|c_j|$ is the size of j^{th} cluster and k is the total number of clusters.

$$SSE = \sum_{j=1}^{k} \sum_{s=1}^{|c_j|} |\overset{\wedge}{t_j} - t_{j_s}| \tag{7}$$

3.7 Clustering using subsequence Information

Clustering of sequences is concerned with grouping a collection of sequences based on their similarity. Clustering is of particular interest in sequence data mining since it provides an attractive mechanism to automatically find some structure in large data sets that would be otherwise difficult to summarize (or visualize). There are many applications where a sequence clustering activity is relevant. For example in web activity logs, clusters can indicate navigation patterns of different user groups. In financial data, it would be of interest to group stocks that exhibit similar trends in price movements.

Sequential data clustering methods could be generally classified into three categories namely, proximity-based methods, feature-based methods and model-based methods. Both proximity based and feature based method require a distance or similarity measure whereas model based methods do not. Feature based clustering has been presented in the current chapter.

The sequences are first transformed to n-dimensional vector space. In order to transform the sequences into vector space the unique set of subsequences of fixed sliding window is extracted from the dataset. These unique set of subsequences form the feature space. Now, the count of occurrence of each unique subsequence

for all the sequences is recorded by sliding the fixed size window across the sequences. Once the whole sequence is encoded into frequency vector, the traditional PAM algorithm with various distance/similarity measures is applied.

The objective of the clustering methods is to discover significant groups present in a data set. In general, they should search for clusters whose members are close to each other (in other words, enforcing a high degree of similarity) and well separated. In order to evaluate the grouping, we used sum-of-squared error criterion in this chapter.

3.7.1 Experimental results

We implemented our approach in Java 1.4 and performed experiments on 2.4 GHz, 256 MB RAM, Pentium-IV machine running on Microsoft Windows XP 2002. Experiments were conducted using *PAM* clustering algorithm with Euclidean distance, Jaccard similarity, Cosine similarity and BWC similarity measure. Each distance/similarity measure was individually experimented with *PAM* clustering algorithm on *msnbc* web log dataset.

We collected $msnbc$ web log data from the UCI dataset repository [*http://kdd.ics.uci.edu/*] that consists of Internet Information Server (IIS) logs for *msnbc.com* and news-related portions of *msn.com* for the entire day of September 28, 1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are not recorded at the finest level of detail but at the level of page categories as determined by the site administrator. There are 17 page categories. The details of preprocessing and description of the dataset is provided in Appendix B. Table 31 presented in the appendix shows the characteristics of the dataset. As the average length of the user session is 5.7, we used sequences of length 6 in our experimentation.

We randomly selected 5000 sequences from the preprocessed dataset for our experimentation. Experiments were conducted with PAM clustering algorithm using various subsequence lengths ranging from L=1 to L=5. For varying subsequence lengths, the number of cluster values (that is k) was also varied and the results were observed.

Table 10: Clustering results using different distance/similrarity measures

Clustering result using Euclidean measure										
	L=1		L=2		L=3		L=4		L=5	
Number of clusters	Number of Iterations	Sum of Squared Error	Number of Iterations	Sum of Squared Error	Number of Iterations	Sum of Squared Error	Number of Iterations	Sum of Squared Error	Number of Iterations	Sum of Squared Error
2	41	4091.5	44	4123.2	56	4254.6	63	4192.6	74	4019.5
3	67	3891.2	56	3902.7	72	4113.5	82	4157.4	91	4215.9
4	72	4139.2	81	4254.6	83	4380.1	92	4412.5	98	4519.2
5	83	4252,1	65	4290.1	87	4379.6	82	4471.3	91	4567.2
Clustering result using Jaccard measure										
2	12	3741.2	13	3701.8	13	3653.1	15	3534.7	19	3519.5
3	12	3471.3	12	3319.6	14	3302.3	15	3219.8	18	3118.4
4	14	3856.1	15	3709.3	15	3593.7	18	3498.6	18	3209.4
5	14	4034.3	15	3917.5	19	3834.1	22	3723.5	26	3694.2
Clustering result using Cosine measure										
2	23	3986.72	24	3342.6	25	3251.2	24	3113.47	33	2913.52
3	25	3607.4	25	3513.1	27	3416.9	24	3398.2	41	3107.4
4	25	3903.6	29	3811.4	35	3612.1	40	3514.7	44	3486.2
5	31	4117.53	43	4012.3	40	3916.4	43	3812.3	49	3772.4
Clustering result using BWC measure										
2	15	3813.1	15	3801.4	16	3767.3	17	3712.9	18	3602.1
3	16	3541.2	16	3490.4	18	3411.3	18	3397.6	19	3291.1
4	16	3912.6	17	3848.2	17	3769.8	18	3689.4	20	3529.8
5	18	4092.1	17	3991.8	18	3906.1	19	3812.5	21	3791.6

As can be observed from the Tables 10 that as the length of the subsequence increases a lower sum-of-squared error was observed irrespective of the distance/similarity measure used. The better the sum-of-squared error the better the quality of clustering. Results demonstrate that for the same k, i.e. number of clusters higher subsequence lengths result in better grouping. These results indicate the importance of sequential information in grouping sequential data.

In the case of Jaccard similarity measure, lowest sum-of-squared error was recorded with respect to other distance/similarity measures for corresponding values of k (number of cluster value) and subsequence length. As Jaccard measure captures the content information the result indicates that the content information is also important. Also, Jaccard similarity measure requires less number of iterations to find the best combination of medoids in the PAM clustering algorithm. The results of Jaccard measure give us an intuition that the content information is also important while considering clustering of sequences.

These results show that in web log mining task where web logs exhibit sequentiality, ignoring the sequential information may lead to incorrect grouping.

3.8 Chapter Summary

Using Intrusion Detection as an example domain, we demonstrated in this chapter the usefulness of utilizing subsequence information for kNN classification of sequential data. We presented results on DARPA'98 IDS dataset wherein we systematically varied the length of the sliding window and used various distance /similarity measures such as Euclidian distance, Jaccard similarity, Cosine similarity and BWC similarity measures. As the subsequence information is increased, high DR is achieved with all the four measures. Also, better AUC was observed for higher subsequence lengths. Our results show that if order information is made available, a traditional classifier such as kNN can be adapted for sequence classification problem.

Clustering of sequences by preserving partial sequential information leads to better groupings with respect to sum-of-squared error criterion in the case of clustering web logs. The web log sequences were converted into frequency vectors using a sliding window scheme. Unique subsequences of fixed sliding window

size L formed the feature vector. Similar to the classification task, in the clustering task also the size of the sliding window was varied. This way the correlation among continuous symbols forming the sequences was maintained. The quality of the clusters formed was evaluated using sum-of-squared-error function. It was observed that as the subsequence size increases the clusters formed were good in respect of inter-intra cluster distances and the number of iterations used to find an optimal combination of medoids was less in case of Jaccard measure.

Finally to conclude, in this chapter, we established the hypothesis that incorporating sequential information plays a key role in classification and clustering of sequences. Both in the case of classification and clustering tasks, we observed that as the sequence information is fed to the kNN classifier or PAM clustering algorithm, the efficiency of the classifier increases as well as the goodness of clusters is better. It was also observed that the Jaccard measure performed well over all the distance/similarity measures under study for classification and clustering tasks. These results indicate that there is a need for a new similarity measure which considers both order of occurrence as well as content information while performing classification and clustering tasks for sequences.

CHAPTER IV

A NEW SIMILARITY METRIC FOR SEQUENTIAL DATA

Technology makes it possible for people to gain control over everything, except over technology. - *John Tudor*

In chapter 3, we demonstrated the usefulness of using subsequence information along with *kNN* for classification of sequential data. We presented results on *DARPA' 98 IDS* dataset where we systematically varied the length of the subsequence and tested in conjunction with various distance /similarity measures such as Euclidian distance, Jaccard similarity, Cosine similarity and BWC (Binary Weigthed Cosine) similarity measure. As expected, classification accuracy improved with increasing availability of subsequence information. Thus, results demonstrated that if order information is made available to traditional classifiers such as kNN, they can also be used for sequence classification problems. Further, Jaccard measure which captures the content information yielded the lowest rate of increase in false positives compared to other measures. Encouraging results were also observed for PAM clustering algorithm on *msnbc* web data.

Therefore, there is a need for a new measure which captures both order information as well as content information among sequences while performing classification/clustering of sequential data. Based on these encouraging results we designed a new similarity measure that captures both order as well as content information.

In this chapter, we propose a similarity preserving function called Sequence and Set Similarity Measure *(S^3M)* that captures both the order of occurrence of items in a sequences as well as the content of sequences.

This chapter is organized as follows. In section 4.1, we present a brief introduction to sequences. In section 4.2, we discuss general characteristics on similarity measures. In section 4.3, we describe about the sequence similarity. A study of longest common subsequence is presented in the section 4.4. We propose a new similarity measure S^3M measure in section 4.5. Finally, we conclude in section 4.6.

4.1 Introduction

In the last ten years, an increasing number of data mining techniques has emerged for which efficient and effective similarity measures are required. In general, the importance of similarity search grows in application areas such as multimedia, medical imaging, molecular biology, computer aided engineering and marketing. [4, 5, 17, 54, 100, 152]. Particularly, the task of finding similarity between sequences becomes more and more important. Examples of applications that require data mining of similar sequences includes databases for molecular biology, medical imaging, web mining and system and computer security domains.

A sequence is made of set of items that happen in time, or happen one after another, that is, in position but not necessarily in relation with time. We can say that a sequence is an ordered set of items. A sequence is denoted as follows:
$S = <a_1, a_2, ..., a_n>$, where $a_1, a_2, \ldots, a_n$ are the item sets in sequence S. Sequence S contains n elements or ordered item sets. Sequence length is defined as the count of number of item sets contained in the sequence. It is denoted as $|S|$ and here, $|S| = n$. A number of metrics have been proposed for sequences, many of them not really qualifying as metrics, because they do not satisfy one or more of the requirements of a metric, for failing on one or more of the requirements. Similarity has both a quantitative and a qualitative aspects. In this chapter, we introduce a new similarity measure that considers both sequence as well as set similarity among sequences. We tested the performance of our proposed similarity measure on both classification and clustering tasks. Standard methods like kNN classification and PAM clustering algorithms were used along with the cosine measure as well as the proposed similarity measure for comparative experimental analysis in chapter 5 and 6, respectively. The effectiveness of the proposed measure is studied in both the intrusion detection (classification task) as well as in the web usage mining (clustering task).

4.2 Similarity

In many data mining applications, we are given unlabelled data and we have to group them based on a similarity measure. These data may arise from diverse application domains. They may be music files, system calls, transaction records,

web logs, genomic data and so on. In these data there are hidden relations that should be explored to find interesting information. For example, from web logs one can extract the information regarding the most frequent access paths; from genomic data one can extract letter or block frequencies; from music files one can extract various numerical features related to rhythm, harmony, etc. One can extract feature vectors from sequential data to quantify parameters expressing similarity. The resulting vectors corresponding to the data are then clustered using existing clustering techniques. The central problem in similarity based sequence clustering is to come up with an appropriate similarity metric.

Formally, *similarity* is a nonnegative real valued function S, defined on the Cartesian product $X \times X$ of a set X. It is called a *metric* on X if for every $x, y, z \in X$, the following properties are satisfied by S.

1. Non-negativity: $S(x, y) \geq 0$
2. Symmetry: $S(x, y) = S(y, x)$
3. Normalization: $S(x, y) \leq 1$

A set X along with a metric is called a *metric space*.

In the following sections, we closely investigate issues related to sequence similarity and propose a new sequence similarity measure, S^3M similarity measure, which forms a core idea of this thesis.

4.3 Sequence Similarity

Sequence comparison finds important and interesting applications in the field of computer science both from the theoretical as well as practical point of view. The vide variety of the applications of sequence similarity finds place in various inter-related disciplines such as computer science, molecular biology, speech and pattern recognition, mathematics etc. Sankoff and Kruskal presents the application of sequence comparison and various methodologies adopted [197]. In computer science, sequence comparison finds its application in various fields such as string matching, text and web classification and clustering.

Sequence mining algorithms make use of either distance functions [51] or similarity functions [19] for comparing pairs of sequences. In this section, we investigate measures for computing sequence similarity.

Given two sequences of equal length we can define a measure of similarity by considering distances between corresponding elements of the two sequences. The individual elements of the sequences may be vectors of real numbers (e. g. in applications involving speech or audio signals) or they may be symbolic data (e. g. in applications involving gene sequences or web data). Symbolic data may be encoded into feature vectors in many ways such as the absence or presence of feature (boolean vector), frequency count of the feature and probabilistic way of encoding. When the sequence elements are feature vectors (with real components) standard metrics such as Euclidean distance, Cosine distance may be used for measuring similarity between two data elements. However, the vector based distance measure is unable to capture subjective similarities effectively. For example, in speech or audio signals, similar sounding patterns may give feature vectors that have large Euclidean distances and vice versa. When the sequences consist of symbolic data we have to define dissimilarity between every pair of symbols which in general is determined by the application. Below we investigate some of the commonly used distance measures for sequences.

Feature distance is a simple and effective distance measure [117]. A feature is a short subsequence, usually referred to as N-gram, where N being the length of the subsequence. Feature distance is defined as the number of subsequences by which two sequences differ. This measure cannot qualify as a distance metric as two distinct sequences can have zero distance. For example, consider the sequences PQPQPP and PPQPQP. These sequences contain the same bi-grams (PQ, QP, and PP) and hence the feature distance will be zero with $N = 2$.

Another common distance measure for sequences is the *Levenshtein distance* (LD) [133]. LD is also known as edit distance. It is good for sequences of different lengths. LD measures the minimum cost associated with transforming one sequence into another using basic edit operations, namely, replacement, insertion and deletion of a subsequence. Each of these operations has a cost assigned to it. Consider two sequences s_1 = "test" and s_2 = "test". As no transformation operation

is required to convert s_1 into s_2, the LD between s_1 and s_2, is denoted as LD(s_1, s_2) = 0. If s_3 = "test" and s_4 = "tent", then LD(s_3, s_4) = 1, as one edit operation is required to convert sequence s_3 into sequence s_4. The greater the LD, the more dissimilar the sequences are. Although LD can be computed directly for any two sequences, in cases where there are already devised scoring schemes as in computational molecular biology [158], it is desirable to compute a distance that is consistent with the similarity score of the sequences.

However, none of these methods capture both the subjective as well as the order information embedded in the sequential data. Some of the metrics like N-gram captures both the subjective as well as order information but the order information is limited to the size of N. Rawat et al. [195] proposed a similarity measure which computes the cosine similarity considering the shared system calls. *BWC similarity measure* was introduced in the area of intrusion detection. This measure does not captures the order information. Based on this insight we devised a new measure S^3M which captures both the content and order information. In general, content information is captured using the well known Jaccard similarity measure. The order information can be captured using the sliding window concept, where local order information in continuous subsequence is captured or by using the Longest common subsequence which captures the approximate global order information. The main problem in calculating similarity between sequences is finding an algorithm that computes a common subsequence between two given sequences as efficiently as possible [203]. Intuitively, when two sequences are similar it is expected that the underlying sequences are also similar, in particular, then it is possible to find the longest common subsequence (LCS). In this chapter, we devise a new similarity measure which takes into account both the order of occurrence information (with the help of LCS measure) and the content information (using Jaccard similarity measure) while comparing two data sequences. Firstly, we make a closer look to the LCS measure and then we devise the new measure.

4.4 Length of Longest common subsequence

The longest common subsequence (LCS) problem can be defined as follows:
Let A_1, A_2 be two sequences of length n_1, n_2, correspondingly, over alphabet $\sum =$

$\{\sigma_1, \sigma_2, \ldots, \sigma_s\}$. A subsequence of A_1 can be obtained by removing zero or more symbols from A_1. Similarly, a subsequence of A_2 can be obtained by removing zero or more symbols from A_2. The longest common subsequence problem for two sequences A_1, A_2 is to find a sequence B such that B is a subsequence of both A_1 and A_2 and it has the largest length. For example, $B =$ "ce" is the LCS for $A_1 =$ "$computer$" and $A_2 =$ "$science$". Note, that for a given set of sequences there can be more than one LCS.

LCS algorithms are implemented using dynamic programming approach. It is not hard to see that the straight-forward implementation of dynamic programming method for constructing the similarity score matrix would lead to $O(mn)$ time and space complexities where m and n are the lengths of two sequences in consideration. Lot of research has been done in implementing LCS algorithm using dynamic programming approach. However, for the general case it is still an open question. We have compiled various algorithms available in the literature with time complexities for computing LCS in table 11. In the table, m and n are the lengths of two sequences such that $n \geq m$. S is the size of the alphabet from which two sequences are made. K denotes the number of minimal mismatches in the two sequences and p is the length of LCS of two sequences.

Table 11: Compiled results for the LCS problem

Year	Author(s)	Time Complexity	Reference
1974	Wagner, Fischer	O(mn)	[222]
1975	Hirschberg	O(mn)	[89]
1977	Hunt and Szymanski	O((n + S) log n)	[96]
1977	Hirschberg	O(mn)	[89]
1977	Hirschberg	O(mn)	[89]
1980	Masek, Paterson	$O(\frac{n^2}{\log n})$	[150]
1984	Hsu, Du	$O(pnlog\frac{n}{p} + pn)$	[94]
1986	Myers	O (n(n-p))	[161]
1982	Nakatsu et al.	$O(n(m-p))$	[164]
1992	Apostolico et al.	O(n(n - p))	[11]
1994	Rick	O(nS + min(KS,pn))	[196]

The degree of similarity between two sequences can be measured by extracting the maximal number of identical symbols existing in both sequences in the same order and is known as (one of) the longest common subsequence(s). The length of

this subsequence describes the similarity between the sequences.

Consider two sequences X[1. . . m] and Y[1. . . n] made up from D different symbols belonging to the alphabet $\sum$. A subsequence S[1. . . s] ($0 \leq s \leq m$) of X can be obtained by deleting arbitrarily (m - s) symbols from X. Further, if S is also a subsequence of Y, then S is a common subsequence of X and Y, denoted by CS(X,Y). The longest common subsequence of X and Y, abbreviated by LCS(X,Y) is the CS(X,Y) having maximal length, whose length is denoted by LLCS. Between any pair of sequences the longest common subsequence need not be unique. That means there may be several subsequences satisfying the Longest Common Subsequence criterion. It is also possible that the same longest common subsequence occurs at multiple positions of the sequences.

In some applications, the length of the LCS is required, whereas in some applications the longest common subsequence itself is required. All algorithms calculating the length of longest common subsequence can be modified to obtain the longest common subsequence. Once the LLCS is known, the LCS can be constructed by backtracking the selections made during the computation of LLCS. Backtracking leads to time and space overhead, which means that the behavior of the LLCS of a LCS algorithm may considerably differ from its LCS-variant. This overhead can be diminished by having knowledge on the lower and the upper bound of LLCS.

A large number of variants of LCS algorithms have been proposed in the literature [19].

4.5 S^{3}M –Similarity measure for sequences

In this section, we propose a new similarity measure for computing similarity between sequences. In order to find patterns in sequences, it is necessary to not only look at the items contained in sequences but also the order of their occurrence. In order to account for both kinds of information, a new measure, called Sequence and Set Similarity Measure (S^3M), is proposed. S^3M consists of two parts: one that quantifies the composition of the sequence (set similarity) and the other that quantifies the sequential nature (sequence similarity). Sequence similarity is defined as

the order of occurrence of item sets within two sequences. Length of longest common subsequence (*LLCS*) with respect to the length of the longest sequence determines the sequence similarity aspect across two sequences. For two sequences A and B, sequence similarity is given by

$$SeqSim(A, B) = \frac{LLCS(A, B)}{\max(|A|, |B|)}$$

Set similarity (Jaccard similarity measure) is defined as ratio of the number of common item sets and the number of unique item sets in two sequences. Thus, for two sequences A and B, set similarity is given by

$$SetSim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Let us consider two sequences A and B, where A = <*a, b, c, d*> and B = <*d, c, b, a*>. Now, the set similarity measure for these two sequences is *1*, indicating that their composition is alike. But we can see that they are not at all similar when considering the order of occurrence of item sets. This aspect is quantified by the sequence similarity component. For instance, sequence similarity component is *0.25* for these sequences. For two sequences, C = <*a, b, c, d*> and D = <*b, a, k, c, t, p, d*>, $LLCS(C, D)$ works out to be *3* and after normalization, the sequence similarity component turns out to be *0.43*. The set similarity for these two sequences is *0.57*. The above two examples illustrate the need for combining set similarity and sequence similarity components into one function in order to take care of both the content as well as position based similarity aspects. Thus, S^3M is constituted as a function of both set similarity and sequence similarity and is defined as below.

$$S^3M(A, B) = p * \frac{LLCS(A, B)}{\max(|A|, |B|)} + q * \frac{|A \cap B|}{|A \cup B|}$$

where $p + q = 1$ and $p, q \geq 0$.

Here, p and q determine the relative weights to be given for order of occurrence (sequence similarity) and to content (set similarity), respectively. In practical applications, user could specify these parameters.

4.5.1 Characteristics of S^3M Similarity Measure

Given a set of finite sequences s_i, $s_j \in$ S, a function $F(s_i, s_j) = Sim_{ij}$ from $S \times S \rightarrow \Re$ (set of real numbers) is called an index of similarity, if it satisfies the following properties.

1. Non-negativity: Sim$_{ij} \geq 0$, $\forall$ s$_i$, s$_j \in$ S
2. Symmetry: Sim$_{ij}$= Sim$_{ji}$, $\forall$ s$_i$, s$_j \in$ S
3. Normalization: Sim$_{ij} \leq 1$, $\forall$ s$_i$, s$_j \in$ S

Proposed similarity function has six parameters namely *p*, *q*, *|LLCS(A,B)|*, *Max(|A|, |B|)*, *|A∩B|* and *|A∪B|*. By the two conditions *p+q = 1* and *p,q* $\geq$ *0*, we can infer that *p* and *q* can never be negative. Rest of the four parameters, being absolute values, cannot attain negative values. Hence, the parameters cannot be negative. Finally, the sum and division operations on non-negative values will always result in positive values. Thus, it is straightforward to see that the first condition of similarity holds. Since all the operations used in computing the similarity score are symmetric, it is easy to see that the proposed similarity function also obeys the symmetry property. Further, the proposed measure is a convex combination of two parameters *p* and *q*. Also, the values of *p* and *q* lie between 0 and 1, hence the convex combination of *p* and *q* will lie between *0* and *1*. Thus, the third property of normalization also holds.

4.5.2 Theoretical justification for choice of 'p' parameter

When the length of longest common subsequence is used as a measure of similarity between pair of sequences then it becomes important to have an idea of expected length of longest common subsequence between them. Equation 8 gives the expected length $EL_n^{(k)}$ of a longest common subsequence over an alphabet of size *k* over all the pair of sequences of length *n* [174].

$$EL_n^{(k)} = \frac{1}{k^{2n}} \sum_{|u|=|v|=n} LLCS(u, v) \tag{8}$$

A bound γ_k on the expected length of the Longest Common Subsequence has

been derived [174]. For every $k \geq 2$ there is some γ_k such that

$$\gamma_k = \lim_{n \to \infty} \frac{EL_n^{(k)}}{n} \tag{9}$$

Exact values of γ_k are not known. Upper bound on γ_k for $k = 1, 2, \ldots, 15$ as calculated by Paterson and Danc'ik [174] is 0.47620 for alphabet size 14 and 0.46462 for alphabet size of 15. The upper bound of γ_k can be used to specify the value of p (and hence q, as $q = 1 - p$) for S^3M similarity measure.

Table 12: The Bounds on γ_k [174]

Alphabet size	Lower Bound	Upper bound	Alphabet size	Lower Bound	Upper bound
2	0.77391	0.83763	9	0.40321	0.55394
3	0.61538	0.76581	10	0.38656	0.53486
4	0.54545	0.70824	11	0.37196	0.51785
5	0.50615	0.66443	12	0.35899	0.50260
6	0.47169	0.62932	13	0.34737	0.48880
7	0.44502	0.60019	14	0.33687	0.47620
8	0.42237	0.57541	15	0.32732	0.46462

In this thesis, we used $p = 0.5$ in the S^3M for the experiments conducted with $msnbc$ web log dataset in Chapter 6. The alphabet size for the $msnbc$ web log dataset is 17 and from table 12 the value of $p = 0.5$ is justified, as for aplhabet size 15 the upper bound on γ_k is 0.46462.

4.6 Chapter Summary

In many data mining applications, both classification as well as clustering algorithms require a distance/similarity measure. The data to be clustered or classified may arise from diverse application domains. They may be music files, system calls, transaction records, web logs, genomic data and so on. In these data there are hidden relations that should be explored to find interesting information. One can extract features from sequential data to quantify parameters expressing similarity. The resulting vectors are then clustered using existing clustering techniques. Similar to clustering, in classification task also a similarity measure is required to determine the class membership of test data or sequence. The central problem in similarity based clustering/classification problem is to come up with an appropriate similarity metric for sequential data.

Researchers or practitioners when dealing with sequences use to first convert them into frequency vector treating all the events within the sequences as independent to each other. Treating sequences in this manner, losses its sequential information embedded in them thus resulting in inaccurate classification or clustering. In chapter 3, we established the hypothesis that the classification and clustering tasks if carried out by using sequential information of sequences leads to a better result irrespective of distance/similarity measure used. It was also observed that the content information of two sequences further enhances the capability of classification and clustering tasks. Thus, we feel that to carry out classification or clustering task, a new measure should be devised which keeps into account both the order of occurrences information as well as the content information.

When two sequences are similar it is expected that the underlying sequences are similar in particular, to the longest common subsequence. In this chapter, we devised a new similarity measure which keeps into account both the order of occurrences information and the content information while comparing two sequences. The proposed measure is a linear weighted combination of Jaccard and LCS measures. The weights are assigned by the user based on experience or knowledge of the domain of application. However, we described an approach for choosing these parameters based on the theory of expected length of the longest common subsequence. We showed that the proposed measure qualifies to be a metric by satisfying properties like reflexive, symmetry and normalization.

CHAPTER V

APPLICATION OF SEQUENCE CLASSIFICATION IN INTRUSION DETECTION

The real problem is not whether machines think but whether men do.

- B. F. Skinner

There has been an enormous growth in commercial and scientific data that are sequential in nature. These data are being generated on a daily basis. An efficient classification method is needed for classification of new data stream based on the pattern in the historical data. $k - NearestNeighbor(kNN)$ has been used and proved to be an efficient classification technique for the two-class problems as shown in chapter 3. The proposed similarity measure, S^3M, which considers both sequence as well as set similarity between sequences is considered in this chapter to calculate the similarity between sequences of system calls. Considering both the order of occurrence as well as the content in sequential data enhances the capabilities of kNN classifier for the two-class problems significantly, especially in the context of intrusion detection. The objective of this work is to construct concise and accurate classifier to detect intrusions based on sequence as well as set similarity to the normal set of system call sequences.

Intrusion detection is the process of monitoring and analyzing the events occurring in a computer system in order to detect signs of security problems. Computer security can be achieved by maintaining and analyzing audit data. Cryptographic techniques, authentication means and firewalls have gained importance with the advent of new technologies. With the ever-increasing size of audit data logs it becomes crucial for network administrators and security analysts to use efficient intrusion detection systems in order to reduce the monitoring activity. Data mining techniques are useful in providing several important contributions to the field of intrusion detection.

The data used in this work is *DARPA'98 IDS* dataset originated from MITs Lincoln Lab. It was developed for intrusion detection system evaluations by DARPA and is considered as a benchmark dataset [111]. We present the description of the dataset in Appendix A.

The chapter is organized as follows. Firstly in section 5.1, we present an introduction of intrusion detection and the basic steps involved in its development. The relevant recent literatures in the area of intrusion detection has been reviewed in section 5.2. Section 5.3 deals with the classification of sequential data in the context of intrusion detection. In section 5.4, we present the experimental results and discussion is presented in section 5.5. Finally we conclude in section 5.6.

5.1 Introduction to Intrusion Detection

Intrusion detection is the process of identifying and responding to suspicious activities targeted at computing and communication resources [199]. An intrusion detection program or system (hereafter abbreviated as IDS) is designed to monitor all the network activities including both within and outside the network and identify any suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromising the system security. IDS is considered to be a passive-monitoring system, since the main function of an IDS product is to give a warning about suspicious activity and not preventing it. IDS covers a large variety of products, whose aim is detecting intrusions. An IDS solution can be in the form of cheaper shareware program or freely distributed open source programs or an expensive and secure vendor software solution.

ID Systems can be classified in various ways. Based on the approach that the system takes, IDS can be classified as *misuse* based or *anomaly* based. Misuse based is also called *signature* based. Misuse based method uses specifically known patterns of unauthorized behavior to predict and detect similar attempts. These specific patterns are called *signatures*. The occurrence of a signature might not signify an actual attempted unauthorized access (for example, it can be an honest mistake), but it is a good idea to take each alert seriously. Depending on the robustness and seriousness of a signature that is triggered, some alarm, response or notification should be sent to the proper authorities. An example of intrusive signature is

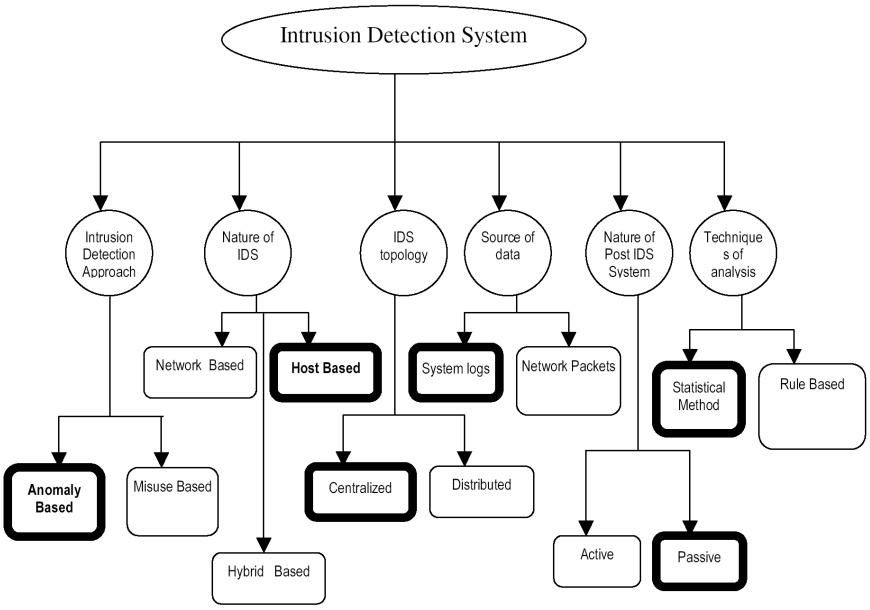

Figure 9: Classification of IDS

"continuous five time login failure".

On the other hand, anomaly based intrusion detection systems are designed to uncover abnormal patterns of behavior. In this method, IDS establishes a baseline of normal usage patterns and anything that widely deviates from it gets flagged as a possible intrusion. What is considered to be an anomaly can vary, but normally, any incident that occurs within a threshold value greater than or less than two standard deviations from the statistical norm raises an alarm. An example of deviation from normal behavior is "a user logs on and off of a machine 20 times a day instead of the normal 4 or 5 times of the day". In this chapter, we designed a program for *anomaly based intrusion detection system*.

IDS can be classified on the basis of data under consideration and its source. If data is held on individual computers that serve as hosts then it is said to be *host* based IDS (HIDS) where as if data is exchanged between computers in a network it said to be *network based IDS* (NIDS). All intrusion detection (HIDS and NIDS) is based on analyzing a set of discrete, time-sequenced events for patterns of misuse. All intrusion detection sources, network or host, make use of sequential pattern of a user that directly reflect specific actions and indirectly reflect behavior. Host-based systems examine events like what files were accessed and what

applications were executed. Network-based technologies examine events such as packets of information exchanged between computers (network traffic). With the growth in technology, the third category of IDS in this classification $hybrid - based\, intrusion\, detection\, system$ has emerged. Hybrid intrusion detection systems offer management alert notification from both network and host-based intrusion detection devices. Hybrid solutions provide the logical complement to NIDS and HIDS - central intrusion detection management. The data under consideration may be from system logs or network packets thus requiring different analysis and treatment.

In this work, we collected benchmark data related to system logs of a computer from a victim Solaris machine, to analyze the efficiency of our intrusion detection program and so our effort falls under the category of HIDS. However, it can be extended further for NIDS also.

IDS can be classified on the basis of mode of operation, namely, *centralized* or *distributed*. If the data are collected and analyzed at different places in the network or system it is called distributed intrusion detection system. While the other is called centralized intrusion detection system.

Based on the analysis of these data, ID Systems can be classified as *statistical* or *rule based* IDS. IDS can take post action in an *active* or *passive* mode. In our work intrusion detection program is *passive* and based on *statistical* aspects.

Figure 9 shows all the possible classification dimensions of IDS discussed above. The boxes shown in dark lines show the characteristics of the intrusion detection problem that this thesis deals with.

5.1.1 Steps in IDS

The various steps involved in the implementation of a typical intrusion detection program or system can be summarized as follows:

Data Collection: For host-based intrusion detection, data collected includes process activity, disk usage, memory usage and system calls. The log of these data are maintained in a file for later analysis. Several tools including UNIX commands such as *netstat*, *ps* and *strace* are used. Whereas for network-based intrusion detection, collection of network traffic i.e. *tcpdump* is carried out using software like

sniffer. *Sniffer* software has filters, which allow the analyst to select the types of packets, hosts and protocols in which he/she is interested. The data can be collected from the network interface in real-time and written to a file.

However, in this thesis no effort has been taken for data collection as we used benchmarking *DARPA'98 IDS* dataset. The details regarding how *DARPA'98 IDS* dataset was collected is presented in Appendix A.

Feature Selection: The raw data collected is usually substantially large, hence it is desired that we only select a subset of this data by creating feature vectors that represent most of the information needed from the data. For host-based intrusion detection, feature vectors include user name, login time and date, duration of the session and number of opened files. Whereas in network-based intrusion detection, typically the Internet Protocol (IP) packet header information is selected as the basis of the feature vector which includes source and destination IP addresses, packet length, layer four protocol type and other flags. In this thesis, the features selected for analysis are system calls of user session.

Analysis: The collected data is analyzed to determine if there is an attack signature or whether the data is anomalous as compared to normal traffic. This is the main research area where a large number of methods have been used and analyzed for detecting intrusions.

In this work, a new session is compared to the training set consisting of normal sessions. If the features of a new session deviate from the model of a normal session obtained from the training set by a user defined threshold value, it is marked as abnormal.

Action: The IDS alerts the system administrator for a possible attack and provides information to the administrator as to the nature of the attacks using several methods including e-mail, alarm icons and visualization techniques.

Based on the analysis step, an alarm can be raised to the system administrator indicating that the current session under analysis is abnormal and that appropriate measures are taken.

5.2 Literature survey

The goal of intrusion detection is to monitor network assets to detect anomalous behavior and misuse. Recently the area of computer and network security has attracted many researchers. The notion of intrusion detection was born in the early 1980s by James Anderson [10]. Since then, several pivotal events in IDS technology have advanced intrusion detection to its current state.

There are two basic approaches for detecting intrusion. The first approach, *anomaly detection*, attempts to define and characterize the correct static form of data and/or acceptable dynamic behavior. In effect, it searches for an anomaly in either stored data or in the system activity. IDS utilizing anomaly detection include Tripwire [114], Self-Nonself [59], NIDES (Next-Generation Intrusion Detection Expert System) [9], ADAM (Audit Data Analysis and Mining) [139], IDDM (Intrusion Detection using Data Mining) [2] and eBayes [216].

The second approach is called *misuse detection*, which involves characterizing known ways to penetrate a system in the form of a pattern. Rules are defined to monitor system activity essentially looking for the pattern. The pattern may be a static bit string or may describe a suspect set or sequence of events. The rules may be engineered to recognize an unfolding or partial pattern. ID Systems utilizing misuse detection include MIDAS (Multics Intrusion Detection and Alerting System) [201], STAT (State Transition Analysis Tool for Intrusion Detection) [98], JAM (Java Agents for Meta learning) [138, 127, 128] MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection) [129] and Automated Discovery of Concise Predictive Rules for Intrusion Detection [126].

In some cases, they are combined in a complementary way in a single intrusion detector. There is a consensus in the community that both approaches continue to have value. Systems taking the combination of both the approaches include NADIR (Network Anomaly Detection and Intrusion Reporter) [91], NSTAT (Network State Transition Analysis Tool) [110], GrIDS (Grid Intrusion Detection) [211] and EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) [189].

Commonly intrusion detection systems take rule based approach [98, 146]. They

are also referred as signature based intrusion detection systems. Rule based intrusion detection systems use an attack "signature" to identify an attack and subsequently alert the system administrator. Hence, initially a list of signatures are developed for making an effective IDS. The major drawback of the rule based approach is the development of signature list. Unfortunately, creating the list of signatures or rules is not trivial.

To overcome the limitations of rule based intrusion detection systems statistical based methods were evolved. Statistical analysis involves statistical comparison of current events to a predetermined set of baseline criteria. The technique is most often employed in the detection of deviations from typical behavior and determination of the similar events to those which are indicative of an attack [84, 160]. Clustering algorithms are used to find the intrusions [132, 190]. Clustering algorithms work by finding set of clusters in the given dataset. In the work by Portnoy [190] a clustering method for automatically detecting both known and new intrusions has been demonstrated using a single-linkage clustering algorithm. Yu Guan [73] proposed a Y-means clustering algorithm for intrusion detection using K-means clustering algorithm.

An increasing amount of research in the last few years has investigated the application of neural networks to intrusion detection. IDS can be implemented using neural networks to learn behavior of attacks [233]. If properly designed and implemented, neural networks have the potential to address many of the problems encountered by rule-based approaches. Neural networks were specifically proposed to learn the typical characteristics of system users and identify statistically significant variations from their established behavior [48, 61].

Other techniques for implementing ID Systems includes variations of the back-propagation algorithm, support-vector machines and radial basis functions. Unsupervised learning methods such as Kohonen's Self-Organizing Maps [30] and Principal Component Analysis [123] have also been applied for IDS problem. These algorithms helps in dimensionality reduction, thereby providing the system administrator with a summary of the multidimensional data on a two dimensional scale. From the field of artificial intelligence, rule-based algorithms are used especially in the misuse model, where the attacks are modelled as a set of rules and

the data is checked against the models. From statistics, Chi-square and hypothesis testing are used along with other methods [8].

There are many ways in which system call data could be used to characterize normal behavior of programs, each of which involves building or training a model using traces of normal session. An approach described by Fink et al. [116], employs security relevant behavior of privileged programs using a program specification language, in which the allowed operations (system calls and their parameters) of a process are formally specified. The methods for normal behavior in terms of short sequences of system calls in a running process is defined by many researchers [59, 93]. This normal behavior ignores many aspects of process behavior, such as the parameter values passed to system calls, timing information, and instruction sequences between system calls. In their work, they demonstrated the usefulness of monitoring sequences of system calls for detecting anomalies induced by processes. These works depend only on enumerating sequences that occur empirically in traces of normal behavior and subsequently monitoring for unknown patterns. Forrest et al. [59] defined normal behavior as a short-range correlations in a process system calls. Using this definition they showed that their approach were able to clearly distinguish between different kinds of processes and that it is perturbed by several different classes of anomalous or foreign, behavior, allowing these anomalies to be detected. Hofmeyr et al. [93] reported that contiguous sequences of some fixed length gave better discrimination. Somayaji and Forrest [208] developed a method that monitors individual processes at the system-call level and automatically responds to anomalous behaviors by either slowing down or aborting system calls. A similar approach is followed by Lee et al. [127] also but they make use of a rule learner RIPPER to form the rules for classification.

Cabrera et al. [26] introduced the concept of anomaly dictionaries which are sets of anomalous sequences for each type of anomaly. The sequences in the anomalous dictionary enable a description of 'Self' for the anomalies. There was no statistical analysis of these sequences in the earlier work. Artificial neural networks have also been used for anomaly detection due to their ability to learn behavior and generalize from this learning [95]. In this approach, they use the Leaky Bucket Algorithm

to capture temporal locality. A scheme based on the kNN Classifier has been proposed by Liao and Vemuri [139], in which each process is treated as a document and each system call as a word in that document. Processes are converted into vectors and cosine similarity measurement is used to calculate similarity among them. A similar approach was followed by Rawat et al. [195], in which they introduced a new similarity measure, termed Binary Weighted Cosine (BWC) metric. This new similarity measure considers both the number of shared system calls between two processes as well as frequencies of those calls.

We extended the work of Liao and Vemuri [139] and Rawat et al. [195] by adopting the same text based classification scheme for system calls with a new similarity measure called, S^3M.

5.3 Classification of Sequential data

Computer security can be achieved by maintaining audit data. Cryptographic techniques, authentication means and firewalls have gained importance with the advent of new technologies. With the ever-increasing size of audit data logs it becomes crucial for network administrators and security analysts to use some efficient intrusion detection program or system in order to reduce the monitoring activity. Data mining techniques are useful in providing several important contributions to the field of intrusion detection.

Formally, the intrusion detection problem on system calls or command sequences can be defined as follows:
Let $\sum = \{s_1, s_2, s_3, \ldots, s_m\}$ be the set of system calls where m denotes the total number of system calls that occur in sequences. A training set D is defined as containing several sequences of normal behavior. The goal in this case, is to devise a binary classification scheme that maps the incoming sequence to the class of normal session or to the class of intrusive session. In general, dealing with sequences is difficult hence each sequence is coded with vector notation. The coding could be binary indicating the presence or absence of a system call in the session or it may be the frequency of each system call in sequences. The other way of vector encoding could be in terms of time stamp associated with the system calls. However, all these techniques does not consider the order information embedded within the

session. In this work, we used the classical kNN classification algorithm with S^3M. Algorithm 4 presents the algorithm for classification of system calls.

Algorithm 4 k-Nearest Neighbor Algorithm for classification of system calls

Input:
S = Set of sessions
k = Number of nearest neighbors
P = New session (Class to be determined)
Sim = Function for similarity measure
τ = User specified similarity threshold
Output:
Class label of session P
Begin
Training
Construct Training sample T from S consisting of normal sessions.
Classification
For session P
do
Calculate $Sim(P, T_j)$ /*T_j is the j^{th} session of test set T, $1 \leq j \leq |T|$.*/
Calculate average similarity, AvgSim, for k nearest neighbors
If (AvgSim $> \tau$)
Mark P as normal
Else
Mark P as abnormal
End For
End

The algorithm consists of two phases namely training and testing. Dataset D consists of m sessions. Each session is of variable length. Initially training set is build by extracting all the normal sequences from the dataset D. The model is trained with the dataset consisting of normal sessions. For a unlabelled session, the similarity metric is constructed using S^3M with all the sessions of the training set. If similarity between any session in training set and new session is equal to 1, mark it as normal. In other case, pick the *k* highest values of similarity between new sample *P* and training dataset. From this *k* maximum values, calculate the average similarity. If the average similarity value is greater than user defined threshold value (τ) mark the new session *P* as normal, else mark *P* as abnormal.

5.4 Experimental results

We implemented the approach in Java 1.4 and performed experiments on 2.4 GHz, 256 MB RAM, Pentium-IV machine running on Microsoft Windows XP 2002. Experiments were conducted using *k-Nearest Neighbor* classifier with S^3M. *DARPA'98 IDS* dataset was used for the experimental purpose.

DARPA'98 IDS dataset consists of *TCPDUMP* and *BSM* audit data. The network traffic of an Air Force Local Area Network was simulated to collect *TCPDUMP* and *BSM* audit data. The audit logs contain seven weeks of training data and two weeks of testing data. There were 38 types of attacks and several realistic intrusion scenarios conducted in the midst of normal background data. Detailed discussion of *DARPA'98 IDS* dataset is given in Appendix A. For experimental purpose, 605 unique processes were used as a training dataset, which were free from all types of attacks. Testing was conducted on 5285 normal processes. In order to test the detection capability of proposed approach, we incorporate 55 intrusive sessions into the test data. To evaluate the efficiency and behavior of the classifier with different similarity measures Receiver Operating Characteristics (ROC) curve is used. ROC curve depicts the relationship between false positive rate (FPR) and detection rate (DR). ROC curve gives an idea of the trade off between FPR and DR achieved by a classifier. Detection rate is the ratio of number of intrusive sessions (abnormal) detected correctly to the total number of intrusive sessions. False positive rate is defined as the number of normal sessions detected as abnormal divided by the total number of normal sessions. The threshold values determine how close the given session is to the training dataset containing all normal sessions. If the average similarity score obtained for the new session is below the threshold value, it will be classified as abnormal. We performed both the parametric and comparative analysis with the new measure called, S^3M.

5.4.1 Parametric analysis

The parametric analysis was performed for two parameters namely p, sequence controlling parameter in S^3M and k the nearest neighbor of kNN classification algorithm. We performed the experiments with $k = 5$, and $k = 10$, that is, the number of nearest neighbors of kNN classification algorithm.

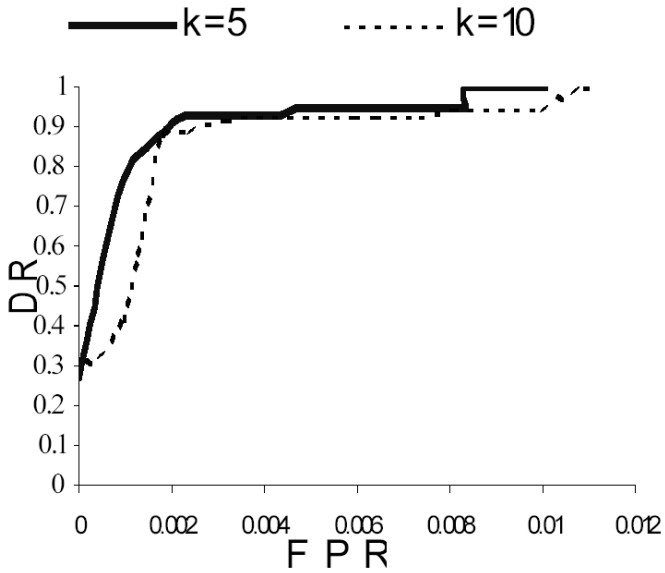

Figure 10: ROC Curves for k= 5 and k=10 using S^3M

Table 13: FPR vs DR for k =5 and k=10 at p = 0.5

	k=5		**k=10**	
th	FPR	DR	FPR	DR
0.89	0.010028	1	0.010974	1
0.88	0.010028	1	0.010974	1
0.87	0.008325	1	0.010974	1
0.86	0.008325	0.94545	0.010785	1
0.84	0.00756	0.94545	0.009839	0.94545
0.8	0.00491	0.94545	0.008136	0.94545
0.78	0.004162	0.92727	0.00719	0.92727
0.75	0.00245	0.92727	0.00359	0.92727
0.7	0.00189	0.8909	0.00227	0.8909
0.65	0.000946	0.76363	0.0017	0.85454
0.6	0.000189	0.36363	0.00089	0.4
0.55	0	0.27272	0	0.2909

Figure 10 shows the ROC curve of kNN classification with S^{3}M measure for $k = 5$ and $k = 10$. Table 13 shows the FPR and DR values for threshold values for the figure 10. For both the k values the p value of S^3M was fixed to 0.5. A high DR of 1 at low FPR of 0.008 was recorded with kNN classifier for $k = 5$ which is comparatively far better than k =10, in which case DR value equal to 1 was recorded at FPR value of 0.013. Thus, a better ROC curve was observed for $k = 5$ than $k = 10$ with S^3M. A low k value means less comparison thus reducing time complexity. As a result, further experiments were carried out using $k = 5$. Table 14 shows the results for $k = 5$ for the different combinations of p and q values. Higher the p value more the sequential information is considered. The low p value infers that the more emphasis is given on the content information than the sequential information. From the table 14 it can be observed that the DR of 1 was recorded for low FPR value of 0.0053 for p value 1. p value 1 means that only weightage is assigned to the sequential similarity and hence no weightage is given for the content information. However, the experiments in chapter 3 shows that apart from the sequential information content information also plays vital role in classifying the intrusive sessions. The second lowest FPR value of 0.007 was recorded for $p = 0.5$ and $p = 0.7$. This shows that the p value for *DARPA'98 IDS* dataset should be chosen between 0.5 and 0.7. These values will contain both sequential as well as content aspect of the dataset at the same it will provide the better classification accuracy. The effect of varying sequence weightage (p) on DR at various threshold values was also studied. With the increase in p value the DR value tends to reach to the ideal value of 1 faster. That means the order of occurrences of system calls plays significant role in determining the nature of a session, whether it is normal or abnormal.

Table 14: FPR vs DR for various combination of p and q values (k=5)

	p =0.0, q =1		**p =0.2, q =0.8**		**p =0.4, q =0.6**		**p =0.5, q =0.5**		**p =0.7, q =0.3**		**p =0.8, q =0.2**		**p =1, q =0**	
th	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR
0.89	0.0072	0.9455	0.0081	0.9455	0.01	1	0.0095	1	0.0108	1	0.01	1	0.0087	1
0.88	0.0072	0.9455	0.0081	0.9455	0.01	1	0.0095	1	0.0098	1	0.01	1	0.0072	1
0.87	0.0061	0.9455	0.0074	0.9455	0.0085	1	0.0089	1	0.0093	1	0.0083	1	0.0072	1
0.86	0.0059	0.9455	0.0074	0.9455	0.0083	1	0.0079	1	0.0089	1	0.0083	0.9455	0.0068	1
0.84	0.0057	0.9455	0.0064	0.9455	0.0076	0.9636	0.007	1	0.007	1	0.0076	0.9455	0.0053	1
0.8	0.0045	0.6182	0.0053	0.9455	0.0055	0.9455	0.0042	0.9455	0.0042	0.9273	0.0049	0.9455	0.0049	0.9455
0.78	0.0042	0.5455	0.0045	0.9455	0.0032	0.9273	0.0042	0.9273	0.004	0.9273	0.0042	0.9273	0.0042	0.9273
0.75	0.0021	0.4909	0.004	0.9091	0.0025	0.9091	0.0038	0.9091	0.0032	0.9091	0.0025	0.9273	0.004	0.8909
0.7	0.0004	0.4182	0.0011	0.4727	0.0019	0.8909	0.0021	0.8727	0.0021	0.8909	0.0019	0.8909	0.0015	0.8545
0.65	0.0002	0.2909	0.0002	0.4	0.0009	0.7818	0.0009	0.8	0.0011	0.8182	0.0009	0.7636	0.0008	0.7636
0.6	0	0.0545	0	0.2909	0.0002	0.4545	0.0002	0.6545	0.0002	0.6545	0.0002	0.3636	0.0002	0.6909
0.55	0	0.0182	0	0.0364	0	0.2545	0.0002	0.2545	0.0002	0.2545	0	0.2727	0.0002	0.1818

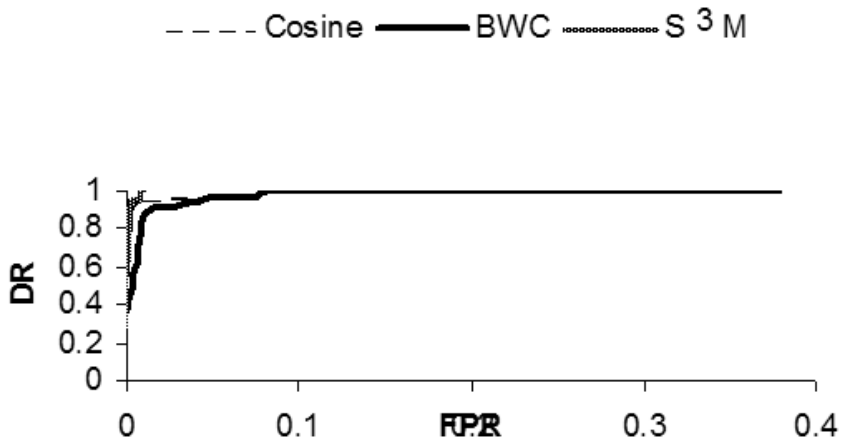

Figure 11: Comparative ROC curves for Cosine, BWC and S^3M measures for k =5

5.4.2 Comparative analysis

The comparative analysis of the S^3M measure was done with respect to *cosine* and *BWC* similarity measures. Figure 11 shows the comparative ROC curves for *DARPA'98 IDS* dataset with the three measures. The experiments were conducted for $k = 5$. k is the nearest neighbor parameter of the kNN classification algorithm. The sequence controlling parameter, p, for the S^3M measure is taken as 0.5. As can be seen from the figure 11 ROC curve due to S^3M measure has high DR than cosine and BWC similarity measures at low FPR. As can be observed from the figure that the detection rate of 1 was achieved at FPR of 0.01 in the case of S^3M measure where as in the case of cosine measure it is achieved at FPR of 0.12. With the BWC similarity measure the DR of 1 was achieved at 0.08 FPR value. These values indicate that with S^3M the *kNN* classification algorithm performed well over cosine and BWC similarity measures. Both the cosine as well as the BWC similarity measures use the vector encoding of the sequences before feding into the kNN classifier, thus losing the sequential information embedded in them. Whereas with the S^3M the sequences are not encoded into the vector rather the similarity is computed directly across two sequences keeping both the sequential as well as content information into account.

Further as quoted in the paper [139, 195] they achieved the DR =1 after removing two attacks from 55 attacks where as DR of 1 was achieved with S^3M

without removing any attacks. There were two attacks included in the testing data namely, $4.5it162228loadmodule$ and $5.5itfdformatchmod$, which could not be detected by Rawat et al. [195] whereas Liao and Vemuri [139] scheme detected them at a lower threshold value. Rawat et al. [195] quoted that though the attack $4.5it162228loadmodule$ was launched, it failed to compromise the system thus making the process to seem normal. In case of the second one, $5.5itfdformatchmod$, the attack was not launched in early stage. Hence, they argued that the attack has not manifested and thus may not be detected by their scheme as they are normal processes. Rawat et al. reported results on experiments after removing these two attacks from the testing data set. To compare with the Rawat et al. [195] results we also removed these two attacks from our test set and experiments were conducted. Figure 12 shows the comparative ROC curves with the full and reduced dataset of abnormal attacks. The curve shows that with the reduced abnormal test set S^3M performed more better hence showing its effectiveness. The decrease in the performance on non-reduced dataset is minimal and of course on the reduced dataset the performance is superior.

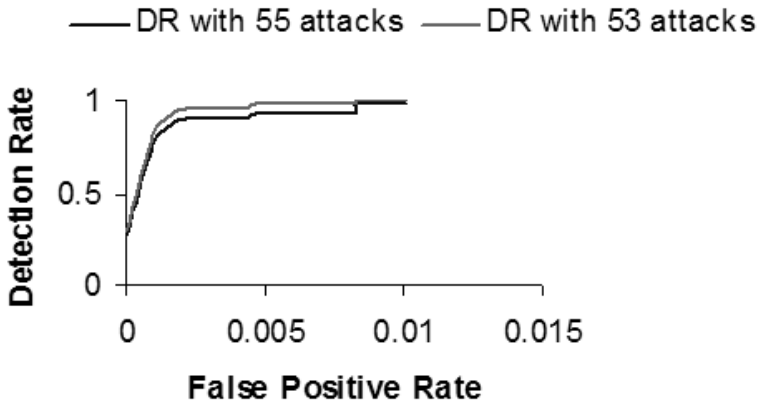

Figure 12: Comparative ROC graph with reduced and without reduced attacks

For *DARPA' 98 IDS* dataset we found that the order of occurrence of system calls plays a key role in determining the nature of the session. From the results presented in this section, we can conclude that the proposed similarity measure could form an integral part of an intrusion detection system that incorporates sequence classification. Experiments with $DARPA'98\ IDS$ dataset for $k = 10$ were

also conducted. A similar trend and results were recorded.

5.5 Discussion

In the previous section we have presented evidence that combining the sequential and content information of system calls results in good discrimination between normal and abnormal sessions. These results are interesting for mainly two reasons. Firstly, the technique shows the importance of considering sequential information in classifying the intrusion. Secondly, the techniques might be applicable to security problems in other computational settings.

Although the results presented in previous section are suggestive, much more testing needs to be completed to validate the approach. In particular, extensive testing on a wider variety of programs being subjected to large numbers of different kinds of intrusions is desirable. For each of these programs, ideally results should be in both controlled environment (in which we could run large numbers of intrusions) and in live user environments. As the *DARPA'98 IDS* dataset is collected from solaries operating system, the testing of the approach should also be done in other operating systems such as Windows NT.

However, there are some problems associated with collecting data in live user environments. Most operating systems do not have robust tracing facilities, hence data may be collected in standardized environments for testing purpose combining with the datasets collected from real time system.

Assuming that more detailed experiments confirm our results, there are a still many problems that need to be addressed before an IDS based on these principles could be implemented. These problems include firstly, what combination of synthetic and actual behavior should be collected to define a normal database? In many user environments, certain normal system calls might be seldom used, and hence a database generated from live user traces might contain false positives, whereas constructing a synthetic database appropriately could prevent these false positives. Secondly, which programs should be monitored, and how (and when) should databases be switched when different processes are started? Finally, an IDS should be a real-time, on-line system that could potentially discover and interrupt intrusions before they are successful. The feasibility of this is highly dependent on

efficient design and implementation of both the tracing facility and the algorithms that detect mismatches. Our emphasis has been on determining mismatches of the intrusive session to the normal.

An important question in the context of an IDS is what response is most appropriate once a possible intrusion has been detected. This is a deep topic and largely beyond the scope of this thesis. Most ID Systems respond by sending an alarm to a human operator. In the long run, however, we believe that the response side will have to be largely automated if IDS technology is going to be widely deployed.

5.6 Chapter Summary

Intrusion detection is the process of monitoring and analyzing the events occurring in a computer system in order to detect signs of security problems. With the ever-increasing size of audit data logs it becomes crucial for network administrators and security analysts to use efficient Intrusion detection systems (IDS) in order to reduce the monitoring activity. Data mining techniques are useful in providing several important contributions to the field of intrusion detection. IDS implementations make use of many techniques including, statistical methods. Statistical Analysis involves statistical comparison of current events to a predetermined set of baseline criteria.

In this chapter, we used kNN classification algorithm for classifying intrusive session. S^3M was used with kNN classification algorithm. Results shows the effectiveness of the proposed measure S^3M. The comparative analysis of the proposed measure with the $cosine$ and BWC measures has been demonstrated where the S^3M performs well over these measures. Parametric analysis of the proposed measure over two quantities p (sequence controlling parameter) and k (nearest neighbor) has been performed.

For *DARPA' 98 IDS* dataset we found that the order of occurrence of system calls plays a key role in determining the nature of the session. From the results presented in this chapter, we can conclude that the proposed similarity measure could form an integral part of an intrusion detection system that incorporates sequence classification.

CHAPTER VI

APPLICATION OF SEQUENCE CLUSTERING IN WEB USER NAVIGATION

Every scientific truth goes through three states: first, people say it conflicts with the Bible; next, they say it has been discovered before; lastly, they say they always believed it.

- Louis Agassiz

With the growth in the number of web users and necessity for making information available on the web, the problem of web usage mining has become very critical and popular. Developers are trying to customize a web site to the needs of specific users with the help of knowledge acquired from user navigational behavior. Since user page visits are intrinsically sequential in nature, efficient clustering algorithms for sequential data are needed. In this chapter, we utilized S^3M proposed in Chapter 4. We conducted pilot experiments comparing the results of *PAM*, a standard clustering algorithm, with two similarity measures: *Cosine* and S^3M. The goodness of the clusters resulting from both the measures was computed using cluster validation technique based on average levensthein distance. Results on pilot dataset established the effectiveness of S^3M for sequential data. Based on these results, we proposed a new clustering algorithm, *SeqPAM* for clustering sequential data. We tested the new algorithm on *msnbc* web navigational dataset.

This chapter opens with an introduction to web usage mining in section 6.1. In the section 6.2, we present the recent related available literature in the area of web usage mining. Section 6.3, presents a methodology for clustering sequential data, in context to user navigational behavior. The proposed clustering algorithm for sequential data is described in section 6.4. Section 6.5 presents the comparative experimental results of *SeqPAM* and *PAM* clustering algorithms on *msnbc* dataset. Finally, summary of the chapter is presented in section 6.6.

6.1 Introduction to web usage mining

With the wide spread of growing information on the internet, web technology is evolving in a unordered and unpredictable fashion. It is estimated that the information available on the internet has expanded by about 2000 % since its existence and is doubling in size every six months. In recent years, the advances in computer and web technologies and the decrease in their cost have expanded the means available to collect and store data. As an intermediate consequence, the amount of information (meaningful data) stored has increased at a very fast pace. Traditional information analysis techniques are useful to create informative reports from data and to confirm predefined hypothesis about the data. However, huge volume of data being collected create new challenges for available techniques as organizations look for ways to make use of the stored information to gain an edge over competitors. It is reasonable to believe that data collected over an extended period contains hidden knowledge about the business or patterns characterizing customer profile and behavior.

Web mining has been and is the focus of many research papers. Web mining is classified into three categories namely,

1. **Web content mining** which is the process of discovering information from various resources available on WWW.

2. **Web structure mining** which is the process of discovering knowledge from the interconnections of hypertext documents.

3. **Web usage mining** which is the process of pattern discovery and analysis.

Our work concentrates in the area of web usage mining. Mining web usage data enables capturing users navigational patterns and identifying users' intentions. Once the user navigational behaviors are effectively characterized, it provides benefits for further web applications such as facilitation and improvement of web service quality for both web-based organizations and for end- users. As a result, web usage mining recently has become active topic for the researcher from the fields of database management, artificial intelligence and information systems [15, 38, 106, 140, 155, 166, 181]. Meanwhile, with the benefits of great

progress in data mining research, many data mining techniques such as clustering [79, 156, 181], association rule mining [3, 6] and sequential pattern mining [7] are adopted widely to improve the usability and scalability of web mining techniques. The clustering techniques for web pages is the prime focus of this work.

Server-side data and client-side data constitute the main sources of data for web usage mining. Different data sources and the technological aspects of the techniques used for gathering web data has been discussed in detail by Pierrakos et al. [184]. Web server access logs constitute the most widely used data source for performing web usage mining. That is why, the term *web log mining* is used interchangeably with *web usage mining*.

Web log mining concerns the analysis of server logs by applying data mining techniques, while web log analysis usually refers to the traffic analysis by applying statistical techniques. A web server log is a comprehensive summary of the access of a specific server from various clients with the time stamp attached to it. Server logs are stored in a variety of formats depending on the server technology used, such as Common Log Format (CLF), Extended Log Format (ELF), Internet Information Server (IIS) format, etc.

In general a log entry will include the URL requested, the IP address from which the request originated, a timestamp, as well as additional fields depending on the log format. Table 15 shows the sample log format. The Table 15 is interpreted as follows:

A user with login name *pradeepkumar* surfing the web site having a client IP address of *172.16.0.144* and *WWW* service downloaded a *GET* operation from the web page */publications/index.htm* of *idrbt* server on 20^{th} *day of May, 2006* at *09 hrs 12 mts 33 seconds*. The IP address of the server is *213.12.0.200*. The request of the user is processed in *1256* milliseconds with *0* errors. The bytes transmitted is *562*.

Advantages of user access data over other types of user data (demographic) can be summarized as follows.

- they are automatically collected
- they are of great volume providing a rich data source
- they cannot be modified or deleted by the users

Table 15: Sample web server log format

Field	Value
Client's IP Address	172.16.0.144
Client's user name	pradeepkumar
Date	20-05-2006
Time	09:12:33
Service	WWW
Server Name	idrbt
IP address of the server	213.12.0.200
Processing time	1256
Bytes Received	312
Bytes Sent	562
Service status code	200
Windows NT status code	0
Name of the operation	GET
Target of the operation	/publications/index.htm

- they protect users privacy, as long as the user identity is not provided and cannot be determined.

Although the web access logs have many advantages and applications still it has its own limitations.

- Due to caching mechanism, it leads to many missing web page references
- the misinterpretation of the IP addresses due to the use of proxy servers assigning the same IP to all users
- the difficulty to update the user model using only access information

6.2 Literature Survey

Web usage mining aims at processing the automatically collected user access data to build suitable user models. Cluster analysis and pattern discovery, association rule mining and sequential pattern mining are the most frequently used techniques covering the needs of web usage mining. The aim of this study is to unearth the available related literature in the area of web usage mining mainly in the context of clustering techniques.

Clustering is an unsupervised classification technique widely used for web usage mining with main objective to group a given collection of unlabelled objects

into meaningful clusters [101]. In web domain, clustering can be performed on either the users or the page views. Clustering analysis in web usage mining intends to find the cluster of user, page or sessions from web log file, where each cluster represents a group of objects with common interesting characteristic. User clustering is designed to find user groups that have common interests based on their behaviors and it is critical for user community.

Page clustering is the process of clustering pages according to the users' access over them. Such knowledge is especially useful for inferring user demographic in order to perform market segmentation in e-Commerce applications or provide personalized web content to the users. On the other hand, clustering of pages will discover groups of pages having related content. This information is useful for the internet search engines and web assistance providers. In both applications, permanent or dynamic HTML pages can be created that suggest related hyperlinks to the user according to the user's query or past history of information needs. The intuition is that if the probability of visiting page is high, then they can be grouped into one cluster. For user session clustering, all the user sessions are processed to find some interesting user session clusters.

Due to the non-numerical nature of web data the development of the new web systems for clustering is required. The factors that influence the performance of clustering algorithms include the viewing time of web documents and the order of the visited documents in a single user transaction. Also, the selection of the appropriate algorithm depends both on the type of data and on the particular purpose for the clustering, since it has great impact on the scalability of the particular clustering approach. The shape of the resulting clusters depends on the choice of the similarity function that measures the degree of similarity among web objects. For example, algorithms based on euclidean or manhattan measures tend to create spherical clusters of similar size. However the formation of clusters with arbitrary shape and size is an important requirement for web usage mining.

Abhraham et al. [1] proposed an ant-clustering algorithm to discover web usage patterns and a linear genetic programming approach to analyze the visitor trends. They proposed hybrid framework, which uses an ant colony optimization algorithm to cluster web usage patterns. The raw data from the log files are cleaned

and preprocessed and the *ACLUSTER* algorithm is used to identify the usage patterns. The developed clusters of data are fed to a linear genetic programming model to analyze the usage trends.

A methodology for automatic classification of the users of a web site based on their access patterns has been proposed by Yan et al. [232]. User visits are represented as n-dimensional vectors of document references with proper weights assigned to them. Based on the weights taken for the representation of web documents the view time of web documents may or may not be taken into consideration. In offline mode, clustering of users' visits is done using Sequential Leader Clustering (SLC), which makes use of the euclidean metric for measuring similarity among user visits. In online mode, the recommendation engine by dynamically generating hyperlinks is provided. Each visit is assigned to only one cluster, thus this approach forms hard clusters. In this approach, the order of visits is not taken into consideration.

The WebCANVAS (Web Clustering Analysis and Visualization of Sequences) presented a new methodology for exploring and analyzing navigation pattern on a web site [29]. The patterns that can be analyzed consist of sequences of URL categories traversed by users. In WebCANVAS algorithm, first site users are partitioned into clusters such that users with similar navigation paths through site are placed into the same cluster. The clustering approach used was model-based and partitioned users according to the order in which user request web pages. The advantage of using model-based clustering is that learning time scales linearly with data size in contrast to the agglomerative distance-based methods which scale quadratically with data size.

A user visit is not just a collection of documents visited in a given period of time rather it offers information about the order of documents visited and possibly the time spent for each document. In general, the length of viewing time cannot always be considered a reliable measure of users preferences, however it may be used as an indication. After clusters of visits have been formed, they can be used for automatic generation of hyperlinks that guide users based on their browsing intentions.

Shahabi et al. [202] proposed an approach to cluster user visits using k-Means

algorithm, a partitioning algorithm that groups the data set into a set of k disjoint clusters. The motivation behind their work is that the purpose of knowledge discovery from users profile is to find clusters of similar interests among the users. If the site is well designed, there will be strong correlation among the similarity of the navigation paths and similarity among the users interest. Therefore, clustering of the former could be used to cluster the latter. Shahabi et al. [202] represented user visits as sequences of pairs of documents along with their corresponding view time. The definition of the similarity is application dependent. They provide an overview on a powerful path clustering method called *path mining*. This approach is suitable for knowledge discovery in databases with partial ordering in their data. In this method, first a general path feature space is characterized. Then a similarity, measure among the paths over the feature space is introduced. Finally this similarity measure is used in the clustering purpose. Path-mining algorithm finds a scalar number as the similarity among the paths. These similarity numbers could be given to standard data-mining algorithms to cluster the user interests.

Fu et al. [62] proposed a clustering approach for modelling of web sites. They said that when the number of pages in web sites is very large then finding groups with similar access patterns does not necessarily correspond to page level. Hence, they came up with a tree like representation, where a leaf node represents a page corresponding to a file in the server and a non-leaf node represents a page corresponding to a directory. They used BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [234] clustering algorithm . BIRCH clustering algorithm requires a clustering feature tree as input, as well as a branching factor and a diameter threshold as input parameters. Euclidean distance is used for measuring similarity among different user visits. Their approach resulted in non-overlapping clusters. They mainly focused on the web usage mining process. The output from clustering is fed to the web master.

Perkowitz and Etzioni [180] proposed *PageGather* algorithm to cluster web document references. It improves the organization and presentation of web sites by learning from processed server log access patterns. Later, they proposed a refinement over their previous algorithm, where they addressed the problem of index

page synthesis [181]. Index page is a collection of hyperlinks that cover a particular area of a web site under consideration. Index page synthesis is to automatically generate index pages in order to facilitate efficient navigation within the site and to provide a novel view of the site. They clustered the references of web documents where each references were represented by a distinct name. Clusters obtained were overlapping in nature. They did not considered the view time on each page references or order of visit in their work. The frequency of co-existence of two web document references in user visits forms the similarity measure for the clustering algorithms. A similarity matrix having the similarity between two web document references forms the basis for the formation of a similarity graph. Then the graph algorithm is applied to identify the connected components or the maximal cliques and finally to find clusters of web document references. Graph algorithm resulted in low computational performance of *PageGather* algorithm. *PageGather* algorithm modifies a web hypermedia system semi-automatically. Web site modification is manually accomplished by the web master, who takes the final decision about what to incorporate in the web site. The clusters are generated automatically in the offline mode.

Mobasher et al. [155] attempts to design a adaptive navigation support system. In their work, they used usage patterns and captured common usage patterns among them. In their work, they have usage profiles comprising of usage patterns obtained after clustering the references of web documents. The references are represented by unique URLs, for the corresponding web documents. Similar to Perkowitz and Etzioni [180] they did not considered the view time and order of visits. They have two modes of operation namely, *offline* and *online* modes. In the offline mode, the algorithm for the generation of frequent item sets is applied to form clusters of document references when two or more documents appear together frequently in user visits. A frequent item set is a set of items occurring at least as frequently as a pre-defined minimum support threshold [3]. Frequent item sets are then used to form a hypergraph, which is partitioned into a set of clusters using Association Rule Hypergraph Partitioning (ARHP). The online component, dynamically provides navigational pointers to users based on their current browsing activity, acting as a real-time recommendation engine.

Paliouras et al. [173] applied clustering techniques on web references by assigning the users of a web site into communities with common behaviors. Each web document reference is referenced by distinct name. They used three algorithms and presented a comparative study on them. Algorithms used in their study are Autoclass algorithm [34], Self-Organizing Maps (SOM) [118] and a variation of PageGather algorithm [181]. Autoclass algorithm produces overlapping clusters of high quality at high computational complexity. SOM algorithms provides visualization of high dimensional data but requires a lot of input parameters. The variation of PageGather has the main advantages and drawbacks of the original version of the algorithm producing overlapping clusters. In all of the three approaches they did not considered the view time or order of visit.

While most of the approaches construct hard clusters, the approach described by Nasraoui et al. [163] in their study on clustering user visits attempted to extract user session profiles by generating overlapping clusters. The approach uses a fuzzy mechanism. However, overlapping clusters of user visits could easily become a prerequisite, since any user may belong to several clusters based on her/his interests and needs. In their work, the representation of visits comprise viewing time for each page visited and the order of visits. The clustering algorithm used here is based on the Relational Fuzzy C-Maximal Density Estimator (RFC-MDE). They proposed a new similarity measure for capturing both individual documents in a cluster as well as the structure of the site.

Banerjee and Ghosh [14] clustered user visits on clicks. User visits were grouped into hard clusters using *Metis* algorithm [108]. User visits were represented as sequences of ordered pairs of web pages visits and corresponding viewing times. They also proposed a new similarity measure consisting of two components, similarity and importance. The Metis algorithm takes as input a similarity graph, in order to generate the desired clusters. In their approach they considered both the visit time on web page reference as well as the order of visit.

Wang and Zaiane [224] used TURN clustering algorithm to cluster user visits. TURN clustering algorithm was proposed in 2001 by Foss et al. [60]. Wang and Zaiane [224] represented users visits as sequences of URLs and the similarity between

sequences is computed by using a new alignment measure based on dynamic programming borrowed from the field of bio-informatics. A similarity matrix is then created and is used as input to TURN clustering algorithm. This approach is proposed in the field of e-learning to determine learning behavior of students. Again in this approach they did not consider the visit time. However they considered the order of visit. The resulting clusters were hard in nature.

Pitkow et al. [185] explored predictive modeling techniques by introducing Longest Repeating Subsequence model which can be used for modelling and predicting user surfing paths. Spiliopoulou et al. [209] built a mining system, WUM (Web Utilization Miner), for discovering of interesting navigation patterns. In their system, interestingness criteria for navigation patterns are dynamically specified by the human expert using WUM's mining language MINT. Mannila and Meek [147] presented a method for finding partial orders that describe the ordering relationship between the events in a collection of sequences. Their method can be applied to the discovery of partial orders in the data set of session sequences.

None of the approaches described above considers both the the order of page visits as well as the content visits simultaneously. Some approaches considers only the order of page visits or content information while computing similarity between user sessions. The sequential nature of web logs makes it necessary to devise an appropriate similarity metric for clustering which should consider both the order as well as content information while computing similarity. The main problem in calculating similarity between sequences is finding an algorithm that computes a common subsequence of two given sequences as efficiently as possible [203]. Intuitively, when two sequences are similar it is expected that the underlying subsequences are also similar, in particular, the longest common subsequence (LCS).

In this work, we used S^3M, proposed in chapter 4, which combines information of both the content as well as their order of occurrence while computing similarity between sequences. S^3M is used with PAM clustering algorithm. The results of S^3M is compared to the cosine similarity measure with PAM clustering algorithm. The meaningfulness of the clusters obtained using both the measures was demonstrated using average levensthein distance and levensthein distance. Further in section 6.4, we designed a new clustering algorithm, $SeqPAM$. $SeqPAM$ differs

from PAM in two aspects: mediod selection and optimization function.

6.3 Clustering of web data

In this section, we used *PAM*, a standard clustering algorithm that represents data in a vectorial form and partitions the space into groups of items that are close to each other based on *cosine* similarity measure. *PAM* is also used here with S^3M, our proposed similarity measure with a representation scheme that preserves sequence information within a session.

In the case of web transactions, each cluster represents a group of transactions that are similar, based on co-occurrence patterns of page categories. Let $\Sigma = \{p_1, p_2, p_3, \ldots, p_m\}$ be the set of page categories, t be a user session and $t \in \Sigma^*$, where Σ^* represents the set of all sessions made up of sequences of page categories. Let D be the training dataset consisting of N user sessions, i.e., $D = \{t_1, t_2, t_3, \ldots, t_N\}$. Each user session can be represented in two ways. In the vectorial representation, $t_i = < f(p_1), f(p_2), f(p_3), \ldots, f(p_m) >$, where each $f(p_j)$ can be formulated in three different ways. $f(p_j) \in \{0, 1\}$ indicating the presence or absence of j^{th} page category in the i^{th} user session, t_i. Boolean representation has been used in the literature [202, 232]. If $f(p_j)$ could represent the duration of time user spends in the j^{th} page category in the i^{th} user session, then user session can be vectorially formulated with respect to the time spent. It has been commented that time spent is not a good indicator of interest [119]. $f(p_j)$ can be used to represent the frequencies of page categories for a user session. In this chapter, for experiments with PAM using cosine measure, we used the third approach.

In the sequence representation scheme, the user session consisting of page categories, i.e., $t \in \Sigma^*$ is considered directly. We have used this formulation in all the experiments where S^3M similarity measure was considered.

Cosine similarity is a common vector based similarity measure. This metric calculates the angle of difference in the direction of two vectors, irrespective of their lengths. Cosine similarity between two vectors, V_1 and V_2 is given by

$$S(V_1, V_2) = \frac{V_1 \bullet V_2}{|V_1||V_2|} \tag{10}$$

6.3.1 Pilot Experiments with PAM

We took 200 randomly chosen web transactions from the *msnbc* dataset and performed the pilot experiments. For the experiments value of k, number of clusters, is fixed to 4 since for PAM with cosine measure the optimal number of clusters achieved was 4. The sum of squared error is used to find the optimal number of clusters.

As these clusters are composed of sessions that are sequential in nature, the cost associated with converting the sequences within a cluster to the cluster representative must be minimum. At the same time, the cost of converting the sequences from two different clusters must be high. We computed a well known measure of the conversion cost of sequences, namely, the levensthein distance for each cluster. The average levensthein distance reflects the goodness of the clusters. Average levenshtein distance (ALD) is expressed as,

$$ALD = \frac{1}{k}\sum_{j=1}^{k}\frac{\sum_{j_s=1}^{|c_j|} LD(\hat{t_j}, t_{j_s})}{|c_j|} \tag{11}$$

where, k is the number of clusters
$|c_j|$ is the size of j^{th} cluster
$\hat{t_j}$ is the medoid of the j^{th} cluster
t_{j_s} is the s^{th} element of the j^{th} cluster
$LD(\hat{t_j}, t_{j_s})$ is the Levensthein distance between the s^{th} member of the j^{th} cluster to its cluster centre.

Table 16: Comparison of clusters formed with the two similarity measures

	LD with cosine measure	LD with S^3M measure
C1	4.514	4.448
C2	4.938	4.62
C3	5.593	3.7
C4	4.92	3.908
ALD	4.9905	4.169

As can be seen from table 16, the ALD for the clusters formed with the S^3M

measure is less than that computed for clusters formed with the cosine measure. So, the user sessions within the clusters formed based on S^{3}M have retained more sequential information than those obtained by the cosine measure.

So far, we have looked at the intra-cluster LD measure where S^3M seems to perform better. The quality of a cluster is also measured by how well various clusters differ from each other and it is usually denoted as inter-cluster distance. Table 17 and 18 show the LD measure across cluster representatives formed using the two similarity measures. Since we considered only user sessions of length 6, the theoretical maximum and minimum inter-cluster LD would be 6 and 0, respectively. In table 17, we find that a minimum cost of 2 is required to convert a cluster representative to another cluster representative (shown in bold face in table 17). Whereas, the minimum cost needed for conversion across clusters is 5 with the S^3M measure (see table 18).

These results clearly point out the advantage of using a measure such as S^{3}M that preserves order information for the pilot dataset.

Table 17: LD between cluster representatives obtained using cosine measure

	C1	C2	C3	C4
C1	0	5	3	6
C2	5	0	5	**2**
C3	3	5	0	6
C4	6	**2**	6	0

Table 18: LD between cluster representatives obtained using S^3M measure

	C1	C2	C3	C4
C1	0	6	5	6
C2	6	0	5	5
C3	5	5	0	5
C4	6	5	5	0

6.4 SeqPAM: *Partition around medoid for sequential data*

From the pilot results presented in the previous section, it has been observed that the PAM partitioning algorithm performed better for web usage data when sequential information was considered. With this inspiration, we modified the standard PAM algorithm and named it *SeqPAM*, partition around medoid algorithm for sequential data.

6.4.1 Modifications in PAM

In SeqPAM, we mainly modify two things over PAM clustering algorithm to make it an efficient algorithm for sequences.

- **Selection of initial medoid**

  Arbitrarily, selecting the initial medoid in traditional PAM clustering algorithm, introduces extra execution time. In order to reduce the execution time required in medoid selection we propose a guided approach for medoid selection.

- **Cost computation**

  Vector based distance measure does not hold good for clustering sequences. Hence an efficient sequence similarity measure is required.

  Calculating the cost associated with medoid reselection incur extra overhead and time consuming as the current approach requires more number of iterations.

6.4.2 Description of SeqPAM algorithm

Consider a dataset D with N item sets, $D = \{t_1, t_2, \ldots, t_N\}$ where each $t_i = < p_1, p_2, \ldots, p_m >$ where p_2 follows p_1, p_3 follows p_2, and so on. Our objective is to cluster these item sets into k distinct clusters. The sequence clustering problem is to identify the underlying clusters based on a given sample of sequences. Because of the variability of the lengths of sequences and lack of good distance metrics for comparing sequences, the sequence clustering problem is non-trivial.

SeqPAM differs from PAM in two aspects. The first difference is in the medoid selection process and the second aspect is in the formulation of objective function.

The optimization function used in SeqPAM accounts for sequence information. The algorithm constructs similarity matrix using S^3M function.

The initial medoid selection process starts by randomly selecting the first medoid. The first medoid guides the process of selecting the remaining $k-1$ medoids. The remaining medoids are selected such that the similarity value between any two adjacent medoids is approximately equal to $1/(k-1)$. The process of selection from guidance of the first medoid ensures that the difference between the similarity values of the first and the k^{th} medoid is maximal (close to one). These k medoids form the initial representatives of k clusters. Each item set from the dataset D is now assigned to the cluster around its nearest medoid. Thus, the entire dataset is partitioned around k medoids.

The objective of SeqPAM is to find clusters such that they have maximal intra-cluster similarity and minimal inter-cluster similarity. To ensure this property, an optimization function called *Total Benefit* (TB) is devised. Since the aim of SeqPAM is to cluster sequential data, total benefit is formulated using our S^3M similarity metric. Total benefit reflects the average intra-cluster similarity with respect to the k medoids. Total benefit is given by

$$TB = \frac{1}{k}\sum_{j=1}^{k}\frac{1}{|c_j|}\sum_{s=1}^{|c_j|}S^3M(\overset{\wedge}{t_j}, t_{j_s}) \tag{12}$$

where, k is the number of clusters
$|c_j|$ is the size of j^{th} cluster
$\overset{\wedge}{t_j}$ is the medoid of the j^{th} cluster
t_{j_s} is the s^{th} element of the j^{th} cluster
$S^3M(\overset{\wedge}{t_j}, t_{j_s})$ is the S^3M between the s^{th} member of the j^{th} cluster to its cluster centre.

A new set of cluster medoids is selected based on the process of maximizing pairwise similarity within the respective clusters. For each cluster, pairwise similarity values among the members of the cluster are computed and a new cluster representative, $\overset{\wedge'}{t_j}$, is chosen that has the maximal average pairwise similarity as shown below.

$$\hat{t}_j' = \arg\max_{t_{j_l}} \left\{ \frac{1}{|c_j|} \sum_{s=1}^{|c_j|} S^3M(\hat{t}_{j_l}, t_{j_s}) \right\} \tag{13}$$

where, $j_l = 1, 2, \ldots, |c_j|$.

All the item sets in the dataset D are re-partitioned around the new set of medoids. The total benefit with respect to new set of medoids is computed. The process of selection of new medoids and re-partitioning is continued till the difference in the successive total benefit values is within the user specified tolerance value, τ. An outline of SeqPAM algorithm is given in Algorithm 5.

Algorithm 5 Algorithm for SeqPAM

Input:
D = Dataset of N item sets
k = number of desired clusters
τ = Tolerance on total benefit (stopping criterion)
Output:
Set of *k* clusters
Begin
Construct the similarity matrix for the dataset D using S^3M.
Select initial medoids.

repeat
Partition the dataset D around k medoids.
Compute the total benefit.

for all Clusters **do**
Compute the new cluster representatives.
end for
until the change in total benefit is within τ
End

6.4.3 Pilot experiments with SeqPAM

Experiments with SeqPAM were conducted on the pilot dataset consisting of 200 web transactions and the results are shown in tables 19 and 20. For the sake of comparison, results of PAM with cosine and S^3M measures are replicated in table 19 from table 16. ALD values are also shown for all the experiments in table 19.

The ALD obtained with SeqPAM (3.685) is better compared to that obtained with cosine (4.9905) and S^3M (4.169) measures used with the PAM clustering algorithm.

Minimum value for LD based inter-cluster similarity using SeqPAM is 5 (see table 20) whereas, for PAM with cosine measure is 2. Hence, a better inter-cluster as well as intra-cluster similarity is achieved using SeqPAM over PAM with cosine measure. Also, SeqPAM results in better intra-cluster similarity than PAM with S^3M measure.

Thus, from the pilot experiments we can conclude that the new sequence clustering algorithm *SeqPAM* is more suitable for sequential data such as web usage data. In the next section, we report results on the larger *msnbc* dataset.

Table 19: Comparative results of PAM and SeqPAM on pilot dataset

	LD using cosine	LD using S^3M	LD using SeqPAM
C1	4.514	4.448	3.427
C2	938	4.62	3.493
C3	5.593	3.7	4.04
C4	4.92	3.908	3.783
ALD	4.9905	4.169	3.685

Table 20: LD between cluster representatives obtained from SeqPAM

	C1	C2	C3	C4
C1	0	6	5	5
C2	6	0	5	6
C3	5	5	0	5
C4	5	6	5	0

6.5 Experimental Results and discussion

All the experiments quoted in this chapter were implemented in Java 1.4 running on 2.4 GHz, 256 MB RAM, Pentium-IV machine on Microsoft Windows XP 2002. Experiments were conducted on *msnbc* web log dataset.

We collected $msnbc$ web log data from the UCI dataset repository [http://kdd.ics.uci.edu/ that consists of Internet Information Server (IIS) logs for *msnbc.com* and news-related portions of *msn.com* for the entire day of September 28, 1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user

during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are not recorded at the finest level of detail but at the level of page categories as determined by the site administrator. There are 17 page categories. The detail preprocessing step and description of the dataset is provided in Appendix B. Table 31 present in the appendix, shows the characteristics of the dataset. As the average length of user session is 5.7 we used the sequences of length 6 in our experimentation.

Preprocessed *msnbc* dataset with 44,062 user sessions was given as input to SeqPAM. The results are summarized in Tables 20 and 21. For this larger datset, k was fixed at 12.

Table 21: LD using SeqPAM and PAM on msnbc dataset

	LD using SeqPAM	LD using PAM
C1	3.265	3.291
C2	3.532	3.919
C3	4.952	4.672
C4	3.905	4.012
C5	3.997	4.619
C6	4.238	4.735
C7	4.713	4.962
C8	4.538	4.829
C9	4.681	4.124
C10	5.293	4.892
C11	3.991	4.726
C12	4.117	4.721
ALD	4.2685	4.4585

The ALD obtained on the larger msnbc dataset using SeqPAM clustering algorithm is 4.2685 as shown in table 21 and the inter-cluster levensthein distances among the 12 clusters ranged from 4 to 6 (see table 22). Both the values indicate that the preprocessed as well as pilot dataset groupings have preserved sequential information embedded in the *msnbc* dataset.

As can be seen from the Table 21, the ALD obtained with SeqPAM is less than PAM ($4.2685 < 4.4585$) thus indicating that the intra-cluster distance for SeqPAM is minimum. We also recorded the LD for each cluster and observed that for SeqPAM the LD value was less than that for PAM (except for the 3 clusters out of the 12 clusters). This figure indicates that in the SeqPAM clustering algorithm the cost

Table 22: Inter-cluster LD with SeqPAM

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
C1	0	5	5	4	6	4	4	5	6	5	5	6
C2	5	0	6	6	4	4	4	5	6	6	4	5
C3	5	6	0	5	6	6	6	4	5	5	6	6
C4	4	6	5	0	5	4	6	4	4	5	6	6
C5	6	4	6	5	0	4	4	5	6	6	5	5
C6	4	4	6	4	4	0	5	5	6	6	6	5
C7	4	4	6	6	4	5	0	5	4	4	5	6
C8	5	5	4	4	5	5	5	0	6	5	5	6
C9	6	6	5	4	6	6	4	6	0	6	6	5
C10	5	6	5	5	6	6	4	5	6	0	5	6
C11	5	4	6	6	5	6	5	5	6	5	0	4
C12	6	5	6	6	5	5	6	6	5	6	4	0

Table 23: Inter-cluster LD with PAM

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
C1	0	4	2	3	4	5	2	6	4	3	5	2
C2	4	0	6	5	5	5	2	4	5	4	5	5
C3	2	6	0	4	5	3	4	6	4	2	5	5
C4	3	5	4	0	4	5	2	3	5	2	4	6
C5	4	5	5	4	0	3	6	5	2	4	5	3
C6	5	5	3	5	3	0	4	6	5	2	3	6
C7	2	2	4	2	6	4	0	4	5	3	5	3
C8	6	4	6	3	5	6	4	0	4	4	5	3
C9	4	5	4	5	2	5	5	4	0	4	6	3
C10	3	4	2	2	4	2	3	4	4	0	4	5
C11	5	5	5	4	5	3	5	5	6	4	0	6
C12	2	5	5	6	3	6	3	3	3	5	6	0

of converting a sequence to its cluster representative is less as compared to PAM. Table 22 and 23 show the LD (inter-cluster distance) using SeqPAM and PAM, respectively. These values are indicators of distance between cluster representatives. The high values obtained for LD indicate better clustering result as compared to PAM.

Since the length of sequences being considered for experimentation from *m*snbc dataset is 6, the maximum value for inter-cluster distance (LD) can be 6. It can be clearly observed from Tables 22 and 23 that for SeqPAM we obtained the value of 6 for 26 pairs of clusters whereas for PAM it was observed only among 9 pairs. Thus these results indicate that the cost of converting a sequence taken from two different clusters formed in SeqPAM is higher than those from the clusters of PAM.

Above result show that the clusters formed in SeqPAM have high inter-cluster distance and low intra-cluster distance than those formed in PAM.

6.6 Chapter Summary

Clustering is an important task in web personalization. User sessions comprising web pages exhibit intrinsic sequential nature. We introduced a similarity measure for sequential data called S^{3}M. We compared the performance of the standard clustering algorithm PAM with cosine as well as S^3M measures over a pilot dataset consisting of 200 web transactions. Results of the pilot experiments established that using sequence sensitive similarity measure such as S^3M results in better clusters. Cluster quality is measured using a cluster validation index called, average levensthein distance (ALD). Based on the pilot experiments, we devised SeqPAM, a modified PAM algorithm for clustering sequential data. SeqPAM differs from PAM in medoid selection process and optimization criterion. Results on *msnbc* dataset were demonstrated to establish the validity of SeqPAM clustering algorithm.

Although the number of clusters (k) has been fixed in this work, optimal number of clusters can be automatically determined based on the cluster validity index utilized in this work. The web transaction data considered for all our experiments had a fixed session length of 6 and so results need to be replicated for various fixed session lengths.

CHAPTER VII

CLUSTERING OF SEQUENTIAL DATA USING ROUGH SETS

An inventor is a person who makes an ingenious arrangement of wheels, levers and springs, and believes it civilization. *-Bierce*

In the last chapter, we clustered user sessions with PAM clustering algorithm as well as with a variation of PAM clustering algorithm, $SeqPAM$. The clusters formed using both the PAM as well as the $SeqPAM$ were hard in nature, in the sense that a user session belongs to one and only one cluster. However, as discussed later in the chapter a user session may belong to multiple groups based on the current user interests. Thus, there is a need for a clustering approach that deals with ambiguity present in the data. Rough set theory is a mathematical tool developed for dealing with these types of ambiguities.

This chapter presents a new indiscernibility-based rough agglomerative hierarchical clustering algorithm for sequential data. In this approach, the indiscernibility relation has been extended to a tolerance relation with the transitivity property being relaxed. Initial clusters are formed using a similarity upper approximation. Subsequent clusters are formed using the concept of constrained-similarity upper approximation wherein a condition of relative similarity is used as a merging criterion. We have compared the results of the proposed approach with that of the traditional hierarchical clustering algorithm using vector coding of sequences. The results establish the viability of the proposed approach. The rough clusters resulting from the proposed algorithm provide interpretations of different navigation orientations of users present in the sessions without having to fit each object into only one group. Such descriptions can help web miners to identify potential and meaningful groups of users.

The rest of the chapter is organized as follows: In section 7.1, we present a introduction to soft clusters and its need in the field of web mining. Overview of rough set theory is presented in section 7.2. Section 7.4 presents the clustering

approach that uses constrained-similarity upper approximation. A step-by-step illustration on a small example into a set of 10 web navigation transactions has been provided to give insight into the working of the proposed algorithm. We present the experimental results on *msnbc* web data in section 7.5. We compare results of the proposed algorithm with those from the traditional hierarchical clustering algorithm using vector coding of sequences. We discuss the results and conclude in section 7.6.

7.1 Introduction

Clusters can be *hard* or *soft* in nature. In conventional clustering, objects that are similar are allocated to the same cluster while objects that differ significantly are put in different clusters. These clusters are disjoint and are called *hard* clusters. In *soft* clustering, an object may be a member of two or more clusters. Soft clusters may have *fuzzy* or *rough* boundaries [142]. In fuzzy clustering, each object is characterized by the partial membership whereas in rough clustering objects are characterized using the concept of a boundary region. A rough cluster is defined in a similar manner to a rough set. The lower approximation of a rough cluster contains objects that only belong to that cluster. The upper approximation of a rough cluster contains objects in the cluster which are also members of other clusters [221]. The advantage of using rough sets is that, unlike other techniques, rough set theory does not require any prior information about the data such as apriori probability in statistics and membership function in fuzzy set theory.

Joshi and Krishnapuram [107] argued that the clustering operation in many applications involves modelling an unknown number of overlapping sets, that is, the clusters do not necessarily have crisp boundaries. Web mining is one such area where overlapping clusters are required. Since web users are highly mobile, it is important to clearly understand and combine users as per their needs and characteristics. The clusters of web users' sessions are different from traditional clusters, where a user may belong to more than one group. A clear distinction among users' visiting behavior may not be present and hence these clusters tend to have vague or imprecise boundaries. Thus, the clustering algorithm should be able to put a user into two or more groups based on his/her needs or interests. This

suggests the use of soft clustering approaches that allow a user session to appear in multiple clusters. Rough clustering can help researchers to discover multiple needs and interests in a session by looking at the multiple clusters that a session belongs to.

Hirano and Tsumoto [86, 87] proposed indiscernibility based clustering method that can handle relative proximity. The automatic personalization of a web site from user transactions is presented using fuzzy proximity relations [45]. Recently, rough set theory has been used for clustering [46, 47, 85, 86, 87, 142, 144]. Lingras proposed two different methodologies based on properties of rough sets for developing interval representations of data [142, 144]. These models are very useful for dealing with soft boundaries. A clustering algorithm using rough approximation to cluster web transactions from web access logs has been attempted [47]. Fuzzy sets and rough sets have been integrated in soft computing framework to develop stronger models of uncertainty [47, 171, 172, 198].

Reliable clustering of web user sessions can be achieved if both the content as well as the order of page visits is considered. In this way, both the actual user's page visit as well as their preferences and requirements are captured. However, most of the approaches in web mining do not utilize the sequential nature of user sessions. Usually sessions are modelled in an n dimensional vector space of web pages. These n-dimensional vectors may be binary, indicating whether a particular web page has been visited or not within a session. These vectors may carry the information regarding the frequency count of web page visits within a session. Thus, depending on the nature of the values associated with these n dimensions, different kinds of limited user behavior analysis are being performed.

In this chapter, we present a hierarchical clustering algorithm that uses similarity upper approximation derived from a tolerance (similarity) relation. The presented approach results in rough clusters wherein an object is a member of more than one cluster.

This chapter presents a new clustering technique for sequences using the concept of constrained-similarity upper approximation. The key idea of our approach is to find a set of features that captures the sequential information of the data sequences as well as the content information. These feature sets are projected into

an upper approximation space. Constrained-similarity upper approximation technique is applied to obtain upper approximation of rough clusters wherein one element may belong to more than one cluster. This chapter uses web mining as an example to demonstrate the usefulness of the proposed rough clustering algorithm.

7.2 Basics of rough set theory

Rough set theory introduced by Z. Pawlak [175] deals with uncertainty and vagueness. Rough set theory became popular among scientists around the world due to its fundamental importance in the field of artificial intelligence and cognitive sciences. The building block of rough set theory is an assumption that with every set of the universe of discourse, we associate some information in the form of data and knowledge. Objects clustered by the same information are similar with respect to the available information about them. The similarity generated in this way forms the basis for rough set theory.

In rough set theory, a set can be defined using a pair of crisp sets, *lower* and *upper* approximations of a set. Incorporating classic axioms with rough set theory, we have rough set based logics and new features that are very useful in intelligent decision-making. Based on uncertain and inconsistent data, rough set logic allows correct reasoning and discovery of hidden-associations.

Let *U* be the universe and let $R \subseteq U \times U$ be an equivalence relation on U, called an indiscernibility relation. The pair *A = (U, R)* is called an approximation space. The lower and upper approximation of set X with respect to R can be written as

$$\underline{R}X = \{x \in U : [x]_R \subseteq X\}$$

$$\overline{R}X = \{x \in U : [x]_R \cap X \neq \phi\}$$

where $[x]_R = \{y \in U \mid xRy\}$ is the equivalence class of x. If $[x]_R \subseteq X$, then it is certain that $x \in X$. If $[x]_R \subseteq U - \overline{R}X$ then it is clear that $x \nsubseteq X$. $[x]_R \subseteq X$ is called rough with respect to *R* iff $\underline{R}X \neq \overline{R}X$. Otherwise *X* is R-discernible. Rough set of *X* is defined by a pair of lower and upper approximations as $(\underline{R}X, \overline{R}X)$. The set $BNR(X) = \overline{R}X - \underline{R}X$ is called rough-boundary (BNR) of *X* and the set $U - \overline{R}X$

is called negative region of X. Figure 13 shows all the approximation spaces of the rough set.

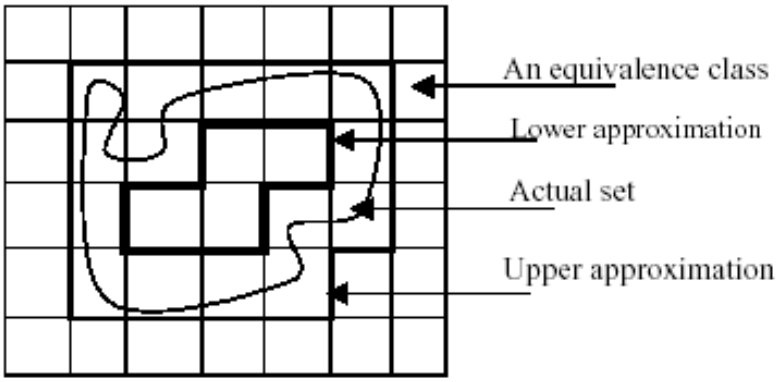

Figure 13: Rough Approximation Space

A basic granule of knowledge is called an *elementary*. It consists of well defined collections of similar objects. A crisp boundary is defined as the union of some elementary objects. The inability to group elementary objects into distinct partitions leads to rough boundaries.

A rough cluster is defined in a similar manner to a rough set, that is with a lower and upper approximation. The lower approximation of a rough cluster contains objects that only belong to that cluster. The upper approximation of a rough cluster contains objects in the cluster which are also members of other clusters. In order to carry out rough clustering, two additional requirements, namely, an ordered value set (V_n) of each attribute N and a distance measure for clustering need to be specified. Ordering of value set (V_n) of attributes enables measurement of distance between objects. A distance measure used in rough clustering should be defined such that the strict requirements of indiscernibility relation used in canonical rough set theory is relaxed. Thus rough clustering allows for grouping of objects based on a notion of similarity relation rather than based on equivalence relation. An important distinction between rough clustering and traditional clustering approaches is that, with rough clustering, an object can belong to more than one cluster.

7.3 Rough set based clustering

When the objects cannot be partitioned using strict indiscernibility relation, it is required to extend this to a binary relation like tolerance relation. For an incomplete information system, reduction of knowledge has been accomplished using tolerance relation [121]. Tolerance based rough set model has been developed for document clustering and information retrieval [90, 109]. Later Ngo and Nguyen [165] extended the work for clustering web search results. Attempts have been made to define rough sets on similarity relation [204, 205]. In general, similarity relations do not give the same kind of partitions of the universe as the indiscernibility relation. Similarity classes of each object x present in the universe U provide the similarity information for the object. An object from one similarity class may be similar to objects from other similarity classes. Therefore the basic granule of knowledge is intermixed. Extending indiscernibility to similarity relation requires weakening of some of the properties of binary relations in terms of reflexivity, symmetry and transitivity [175, 176].

The reflexivity property cannot be relaxed, as an object is trivially similar to itself. Researchers dealing with similarity relations do impose the symmetry property but some researchers relax it [204]. However, most researchers extending indiscernibility relations to similarity relations do relax the transitivity property [205].

Let $X \subseteq U$ then a relation $\tau \subseteq X \times U$ is a tolerance relation on U, if

1. τ is reflexive, that is for any $x \in U$, $x \ \tau \ x$ holds.
2. τ is symmetric, that is for any pair of $x, y \in U$, $x \ \tau \ y = y \ \tau \ x$.

Definitions of lower and upper approximations of set can now be easily formulated using tolerance classes. In order to do this, we substitute tolerance classes for indiscernibility classes in the basic definition of lower and upper approximations of set. Thus, the tolerance approximations of a given subset X of the universe U is defined as in definition 1.

Definition 1 [47].

Let $X \subset U$ and a binary tolerance relation R is defined on U. The lower approximation of X, denoted by $\underline{R}(X)$ and the upper approximation of X, denoted by $\overline{R}(X)$

are respectively defined as follows:

$\underline{R}(X) = \{x \in X, R(x) \subseteq X\}$ and

$\overline{R}(X) = \cup_{x \in X} R(x)$ ■

In this work, we propose an agglomerative clustering algorithm using rough sets for clustering web user transactions. Let $x_i \in U$ be a user transaction consisting of sequence of web page visits. For clustering user transactions, initially each transaction is taken as a single cluster. Let the i^{th} cluster be $C_i = \{x_i\}$. Clearly, C_i is a subset of U. The upper approximation of C_i, denoted as $\overline{R}(C_i)$, is a set of transactions similar to x_i, that is, a user visiting the web pages in x_i may also visit other web pages present in the transactions belonging to $\overline{R}(C_i)$.

For any non-negative threshold value $\delta \in (0, 1]$ and for any two objects $x, y \in U$, a binary relation τ on *U* denoted as $x \ \tau \ y$ is defined by $x \ \tau \ y$ iff $Sim(x, y) \ \geq \ \delta$. This relation *R* is a *tolerance* relation and *R* is both reflexive and symmetric but transitivity may not always hold.

The first upper approximation $\overline{R}(\mathrm{x}_i)$ is a set of objects that are most similar to x_i. Thus, first upper approximation of an object x_i can be defined as follows:

Definition 2.

For a given non-negative threshold value $\delta \in (0, 1]$ and a set $X = \{x_1, x_2, \ldots, x_n\}$, $X \subseteq U$ the first upper approximation is

$$\overline{R}(\{x_i\}) = \{x_j | sim(x_i, x_j) \geq \delta\}$$ ■

Let $\mathrm{t}_i \in \mathrm{T}$ be user web log. The upper approximation $\overline{R}\ \overline{R}(\mathrm{t}_i)$ is a set of transactions similar to t_i, that is, a user, who is visiting the hyperlinks in t_i, may also visit the hyperlinks present in other transactions in $\overline{R}(\mathrm{t}_i)$. Similarly, $\overline{R}\ \overline{R}(\mathrm{t}_i)$ is a set of transactions that are possibly similar to $\overline{R}(\mathrm{t}_i)$, and this process continues until two consecutive upper approximations for $\mathrm{t}_i$are same. The process of finding the two equal consecutive upper approximations is known as *Similarity Upper Approximation* and denoted by S_i.

Initially, each web log transaction has been considered as individual cluster. The similarity upper approximation for each web log transaction is calculated for a given web log transaction data set. In each iteration of agglomerative clustering,

the clusters are agglomerates based on the similarity upper approximation. The process of computing similarity upper approximation is repeated for each transaction, until the two consecutive upper approximations are same.

Let $S_1, S_2, S_3, \ldots, S_n$ be similarity upper approximation for transaction $t_1, t_2, t_3, \ldots, t_n$ respectively. Now, if $S_i = S_j$ (i $\neq$ j) allocate t_i and t_j in the same cluster. Performing this way, we get a distribution of m disjoint clusters. Let these m clusters be $C_j (j = 1, 2, \ldots, m)$. Here, C_j's are all distinct and $\cup C_j = T$. These C_j's represent the subgroups of the transactions representing the transaction cluster.

To illustrate our approach, we considered *10* web data transactions whose similarity table is computed using S^3M similarity metric with *p=0.5* (Table 24). The first similarity upper approximation at threshold value δ = 0.2 is given by $\overline{R}$ (Ti) for i =1, 2, . . . , 10 as given below:

Table 24: Similarity matrix using S^3M metric with p =0.5

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
T1	1	0	0	0	0.21	0.29	0	0	0	0
T2	0	1	0	0.47	0.17	0.17	0.17	0	0.15	0.15
T3	0	0	1	0	0	0.25	0.33	0.33	0	0.21
T4	0	0.47	0	1	0.17	0	0.45	0.27	0.24	0.5
T5	0.21	0.17	0	0.17	1	0.18	0	0	0	0
T6	0.29	0.17	0.25	0	0.18	1	0.18	0.21	0	0.17
T7	0	0.17	0.33	0.45	0	0.18	1	0.58	0.17	0.62
T8	0	0	0.33	0.27	0	0.21	0.58	1	0	0.5
T9	0	0.15	0	0.24	0	0	0.17	0	1	0.24
T10	0	0.15	0.21	0.5	0	0.17	0.62	0.5	0.24	1

$\overline{R}$ (T1) = {T1, T5, T6}

$\overline{R}$(T2) = {T2, T4}

$\overline{R}$(T3) = {T3, T6, T7, T8, T10}

$\overline{R}$ (T4) = {T2, T4, T7, T8, T9, T10}

$\overline{R}$ (T5) = {T1, T5}

$\overline{R}$(T6) = {T1, T3, T6, T8}

$\overline{R}$(T7) = {T3, T4, T7, T8, T10}

$\overline{R}$(T8) = {T3, T4, T6, T7, T8, T10}

$\overline{R}$(T9) = {T4, T9, T10}

$\overline{R}$(T10) = {T3, T4, T7, T8, T9, T10}

In the first step, the second similarity upper approximation of all the 10 web log transactions is given as follows:

$\overline{R}\ \overline{R}$(T1) = { T1, T3, T5, T6, T8 }

$\overline{R}\ \overline{R}$(T2) = {T2, T4, T7, T8, T9, T10}

$\overline{R}\ \overline{R}$(T3) = {T1, T3, T4, T6, T7, T8, T10}

$\overline{R}\ \overline{R}$(T4) = {T2, T3, T4, T6, T7, T8, T9, T10}

$\overline{R}\ \overline{R}$(T5) = $\{T1, T5, T6\}$

$\overline{R}\ \overline{R}$(T6) = {T1, T3, T4, T5, T6, T7, T8, T10}

$\overline{R}\ \overline{R}$(T7) = {T1, T2, T3, T4, T6, T7, T8, T9, T10}

$\overline{R}\ \overline{R}$(T8) = {T2, T3, T4, T6, T7, T8, T9, T10}

$\overline{R}\ \overline{R}$(T9) = {T2, T3, T4, T7, T8, T9, T10}

$\overline{R}\ \overline{R}$(T10) = {T2, T3, T4, T6, T7, T8, T9, T10}

The computation for third similarity upper approximation will be continued as the value of two consecutive similarity upper approximation is not same.
the third similarity upper approximation will be given as follows:

$\overline{RR}\ \overline{R}$(T1) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T2) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T3) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T4) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T5) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T6) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T7) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T8) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T9) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

$\overline{RR}\ \overline{R}$(T10) = { T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

The fourth upper similarity approximation value is given as follows:

$\overline{RRR}\ \overline{R}$(T1) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }

$\overline{RRR}\ \overline{R}$(T2) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }

$\overline{RRR}\,\overline{R}$(T3) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }
$\overline{RRRR}$(T4) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }
$\overline{RRRR}$(T5) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }
$\overline{RRRR}$(T6) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }
$\overline{RRRR}$(T7) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }
$\overline{RRRR}$(T8) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }
$\overline{RRRR}$(T9) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }
$\overline{RRRR}$(T10) = { **T1, T2, T3, T4, T5, T6, T7, T8, T9, T10** }

Since the third and fourth similarity upper approximation is same hence the final set of cluster is given as
{ T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 }

In the above approach the major drawback is that for smaller threshold values, all the web logs will form a single cluster and for higher threshold values all will form a distinct clusters. In the next section, we overcome this problem by introducing the concept of relative similarity.

7.4 Proposed constrained rough clustering algorithm

" *The concept must have a sharp boundary. To the concept without a sharp boundary there would correspond an area that* $\underline{does\ not}$ *had a sharp boundary-line all around.*" Frege as translated in Pawlak [175].

In many data mining applications, the class attributes of most objects are not distinct but vague. Vagueness in data has attracted mathematicians, philosophers, logicians and recently computer scientists. Rough set theory is an approach to deal with vagueness. In the literature, considerable work has been carried out using rough sets to deal with this kind of uncertainty [45, 46, 47, 50, 71, 188, 198].

As discussed in section 7.2, the core concept of rough set theory is the indiscernibility relation which posses reflexive, symmetry and transitivity properties. Indiscernibility relation partitions the universe into equivalence classes, which form the basic granules of knowledge.

We compute the value of the second and the higher similarity upper approximations under the condition of relative similarity. Let $x_i, x_j \in U$. The relative

similarity of x_i with respect to x_j is given by

$$RelSim(x_i, x_j) = \frac{|\overline{R}(x_i) \cap \overline{R}(x_j)|}{|\overline{R}(x_i) - \overline{R}(x_j)|} \tag{14}$$

Now we define the proposed constrained-similarity upper approximation in the following definition.

Definition 3.

Let $X = \{x_1, x_2, \ldots, x_n\}$, $X \subseteq U$. For a fixed non-negative value σ $\in$(0,1], the constrained-similarity upper approximation of x_i is given by

$$\bar{R}\bar{R}(\{x_i\}) = \left\{ x_j \in \bigcup_{x_l \in \bar{R}(x_i)} \bar{R}(x_l) \mid RelSim(x_i, x_j) \geq \sigma \right\}$$

where, $\overline{R}(x_i) \not\subseteq \overline{R}(x_j)$. ■

In other words, all the sequences x_j which belong to the similarity upper approximations of elements of $\overline{R}(x_i)$ that are relatively similar to x_i are constrained (or merged) into the next similarity upper approximation of x_i.

We repeat the process of computing successive constrained-similarity upper approximations for a given σ until two consecutive constrained-similarity upper approximations remain the same. Here, σ is a user-defined parameter called *relative similarity*, used to merge two upper approximations for the formation of the second and higher upper approximations. δ is a user defined threshold parameter use to define the similarity between two objects and is utilized to find the first upper approximation. The constrained-similarity upper approximation is computed for all transactions of *U*. The complete algorithm for computation of rough set based agglomerative clustering is given in algorithm 6.

Unlike other traditional agglomerative algorithms, in our approach more than two transactions may merge to form clusters. Also, the number of upper approximation computations for similarity sets decreases as the number of iterations increases. Thus, the proposed rough agglomerative clustering converges faster.

Algorithm 6 Rough Set based agglomerative clustering

Input:
T: A set of n transactions $\in U$
Threshold $\delta \in$ (0,1]
Relative similarity $\sigma \in$ (0,1]
Output:
Cluster scheme C
Begin
Step 1: Construct the similarity matrix using S^3M measure.
Step 2: For each $x_i \in U$, Compute $S_i = \overline{R}(x_i)$ using definition 2 for given threshold δ.
Step 3: Let $US = \bigcup_i S_i$,
$C = \emptyset$
Step 4: For all $S_i \in US$ Compute the next constrained-similarity upper approximation S' using definition 3 for relative similarity σ

if $S_i = S'_i$ **then**
$C = C \bigcup S'_i$
$US = US \backslash \{S_i\}$
end if
Step 5: Repeat step 4 until $US \neq \emptyset$
Step 6: Return C
End

7.4.1 Complexity of the algorithm

The complexity analysis of our approach is as follows: Let *N* be the total number of transactions in *T* and *L* be the average length of the transaction. The complexity of similarity computation is in the order of $O(N^2 \log_2 L)$. Let *R* be relation defined over *T*, then the complexity of upper approximation is in the order of $O(|T|/|R|)$ [104], which is same as $O(N/|R|)$. Merging of clusters takes place at each iteration, based on the similarity upper approximation. Let *k* be the average number of clusters merging in each iteration. The complexity of merging k clusters is in the order of $O(k \log k)$ [44] and there may be a maximum of N/k iterations. Thus, the complexity of the merging process is $O((N/k) k \log k) = O(N \log k)$. So, the complexity of rough agglomerative clustering is of the order of $O(N^2 log_2 L) + O(N/|R|) + O(N log k)$.

7.4.2 An Example

To illustrate our approach, we considered *10* data sequences as shown in Figure 24. The similarity table was computed using S^3M similarity metric with *p=0.5* (Table 24).

The first similarity upper approximation at threshold value δ = 0.2 is given by $\overline{R}$ (t_i) for i =1, 2, . . . , 10 as given below:
$\overline{R}$ (T1) = {T1, T5, T6}
$\overline{R}$(T2) = {T2, T4}
$\overline{R}$(T3) = {T3, T6, T7, T8, T10}
$\overline{R}$ (T4) = {T2, T4, T7, T8, T9, T10}
$\overline{R}$ (T5) = {T1, T5}
$\overline{R}$(T6) = {T1, T3, T6, T8}
$\overline{R}$(T7) = {T3, T4, T7, T8, T10}
$\overline{R}$(T8) = {T3, T4, T6, T7, T8, T10}
$\overline{R}$(T9) = {T4, T9, T10}
$\overline{R}$(T10) = {T3, T4, T7, T8, T9, T10}

In the first step, the second similarity upper approximation of $\bar{R}(T1)$ is given by
$\overline{R\bar{R}}'$(T1) = {T1, T3, T5, T6, T8}

Now, constrained-similarity upper approximation is applied on $\bar{R}\bar{R}'$ using definition 3 and σ =1. It can be seen that only the elements T1, T5 and T6 qualify to be in the $\bar{R}\bar{R}(T1)$.

For example, consider element T3, $\overline{R}(T1) \cap \overline{R}(T3) = \{T6\}$ and $\overline{R}(T1) - \overline{R}(T3) = \{T1, T5\}$. Thus, the relative similarity between T1 and T3 is

$$RelSim(T1, T3) = \frac{|\bar{R}(T1) \cap \bar{R}(T3)|}{|\bar{R}(T1) - \bar{R}(T3)|} = \frac{1}{2} < \sigma(= 1)$$

Hence, T3 will not merge with $\overline{R}$(T1).

Thus, the family of constrained-similarity approximations is given as follows:
$\overline{R}\,\overline{R}$(T1) = {**T1**, **T5**, **T6**}
$\overline{R}\,\overline{R}$(T2) = {**T2**, **T4**}

$\overline{R}\,\overline{R}$(T3) = {**T3**, **T6**, **T7**, **T8**, **T10**}
$\overline{R}\,\overline{R}$(T4) = {**T2**, **T4**, **T7**, **T8**, **T9**, **T10**}
$\overline{R}\,\overline{R}$(T5) = {**T1**, **T5**}
$\overline{R}\,\overline{R}$(T6) = {T3, T6, T8}
$\overline{R}\,\overline{R}$(T7) = {**T3**, **T4**, **T7**, **T8**, **T10**}
$\overline{R}\,\overline{R}$(T8) = {**T3, T4, T6, T7, T8, T10**}
$\overline{R}\,\overline{R}$(T9) = {**T4**, **T9**, **T10**}
$\overline{R}\,\overline{R}$(T10) = {**T3**, **T4**, **T7**, **T8**, **T9**, **T10**}

In the above set family for the sets shown in bold the two consecutive similarity upper approximations are the same. In this case, the first and the second similarity upper approximation are the same. For example,
$\overline{R}$(T1) = $\overline{R}\,\overline{R}$(T1) = {T1, T5, T6}
Thus, the third similarity upper approximation will be computed for only those elements whose consecutive similarity upper approximations are not the same. Thus, only T6 needs to be considered for the third similarity upper approximation.
$\overline{R}\,\overline{R}\,\overline{R}$(T6) ={**T3**, **T6**, **T8**}

Now since there is no change in the constrained-similarity upper approximations for all the elements, the algorithm has converged. The final cluster family is given by:

{T1, T5, T6},
{T2, T4},
{T3, T6, T7, T8, T10},
{T2, T4, T7, T8, T9, T10},
{T1, T5},
{T3, T4, T7, T8, T10},
{T3, T4, T6, T7, T8, T10},
{T4, T9, T10} and
{T3, T4, T7, T8, T9, T10}

Figures 14, 15 and 16 show the first second and third similarity upper approximations respectively, for the 10 data sequence example set. Figure 16 shows the final overlapping clusters formed as per our proposed methodology. For the sake of clarity, we removed those clusters in Figure 16 that are the proper subsets of one cluster. That is, C_1 will be removed if and only if C_1 is a subset of any C_m. In Figure 14, transaction T1 is within the cluster shown by the dotted oval during the formation of second upper approximation whereas in Figure 14, transaction T1 has left the dotted cluster and it is no more a member of the dotted cluster during the third upper approximation formed due to constrained merging.

It can be clearly seen that many transactions belong to multiple clusters, for example T1 belongs to first as well as fourth cluster.

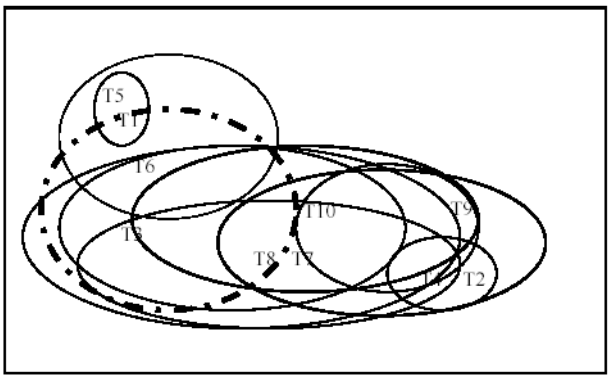

Figure 14: First Upper approximation

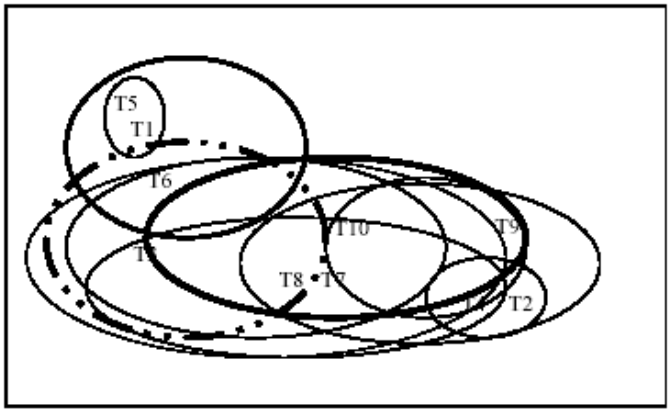

Figure 15: Second Upper approximation

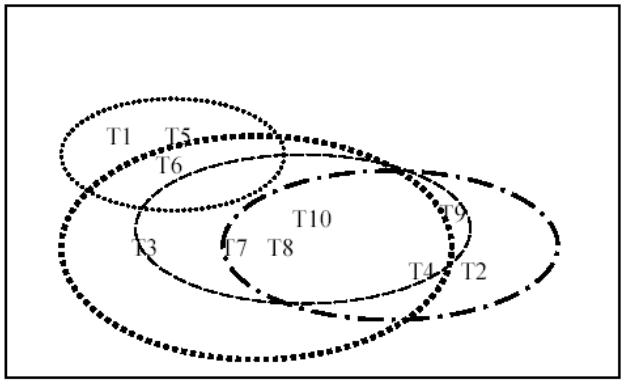

Figure 16: Third Upper Approximation

7.5 Experimental results and discussion

We implemented our approach in Java 1.4 and performed experiments on 2.4 GHz, 256 MB RAM, Pentium-IV machine running on Microsoft Windows XP 2002. Experiments were conducted on *msnbc* web log dataset.

We collected $msnbc$ web log data from the UCI dataset repository [http://kdd.ics.uci.edu/] that consists of Internet Information Server (IIS) logs for *msnbc.com* and news-related portions of *msn.com* for the entire day of September 28, 1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are not recorded at the finest level of detail but at the level of page categories as determined by the site administrator. There are 17 page categories. The detail preprocessing step and description of the dataset is provided in Appendix B. Table 31 present in the appendix, shows the characteristics of the dataset. As the average length of user session is 5.7 we used the sequences of length 6 in our experimentation.

Experiments were conducted on randomly selected samples of 100, 200, 300, 400 and 500 records from the preprocessed data set with the S^3M metric. We considered the value of p as 0.5 with the S^3M metric. The value of relative similarity (σ) used to define the constrained-similarity upper approximation is taken as 1. Figure 17 shows the number of clusters formed for 100, 200, 300, 400 and 500 records for threshold values 0.8 and 0.9. Table 25 shows the size of lower approximation sets for our pilot dataset consisting of 100, 200, 300, 400 and 500 records at δ value

of *0.8* and *0.9*. As the number of records increases the number of clusters formed also increases with the same amount of variation for both the threshold values.

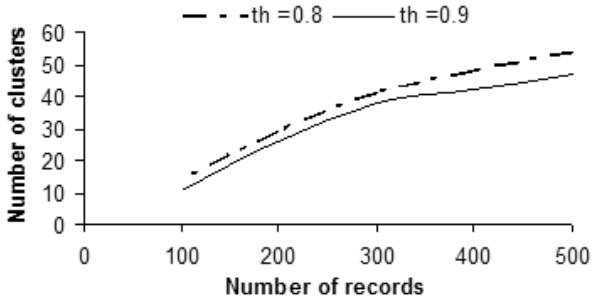

Figure 17: Graph between numbers of clusters formed with different number of records

Table 25: Lower approximation for different number of records at different threshold values

Number of records	δ =0.8	δ =0.9
100	9	9
200	22	23
300	25	28
400	27	31
500	30	31

Table 26: Number of records with null Lower approximation for different number of records at different threshold values

Number of records	δ =0.8	δ =0.9
100	2	2
200	6	5
300	6	7
400	8	11
500	9	9

In order to capture the implicit grouping of the dataset, the size of the lower approximation must be as small as possible. From Table 25, it is evident that for the 500 sample size dataset, the size of the lower approximation is 30 and 31 at δ values of 0.8 and 0.9, respectively. The nature of several clusters formed in the proposed approach is such that all the members of a cluster C_i belong to other clusters C_j

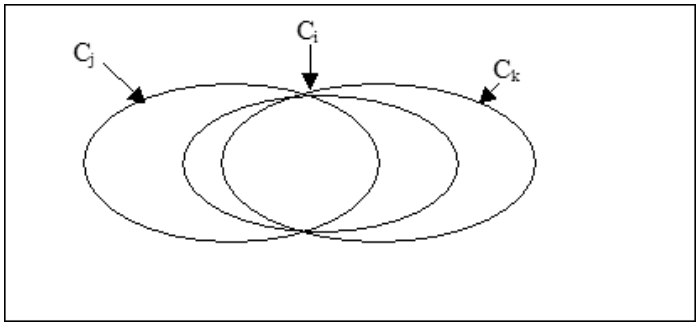

Figure 18: Cluster example scenario in rough clustering. Here C_i has a null lower approximation

and C_k as shown in figure 18. These type of clusters retain the ambiguity of the clustering scheme hence justifying the application of rough set based clustering for web transaction data. Table 26 summarizes the number of clusters where the lower approximation set is null for our pilot dataset at δ values of 0.8 and 0.9.

Further, we randomly selected 5000 transactions from the preprocessed dataset and executed our proposed approach. We obtained 2674 overlapping clusters at threshold value(δ) 0.8 and relative similarity (σ) of 1. For this work, p value taken was 0.8. The size of lower approximations set is 603. There were 1628 clusters in total that did not have any lower approximation.

Effectiveness of the clustering technique was also measured quantitatively. We calculated the cluster representative by maximizing the average pairwise similarity among the members of the jth cluster using the formula given below:

$$\hat{C_j} = \arg\max_{j_l} \left\{ \frac{1}{|C_j|} \sum_{j_l, j_s=1}^{|C_j|} S^3M(C_{j_l}, C_{j_s}) \right\} \tag{15}$$

where, $j = 1, 2, \ldots, k$.

Above formula finds the cluster representative that maximizes intra-cluster distance using S^3M similarity measure. As these clusters are composed of sessions that are sequential in nature, the cost associated in converting the sequences within a cluster to the cluster representative must be minimum (intra-cluster distance). At the same time, the cost of converting the sequences from two different clusters must be high (inter-cluster distance). We computed the conversion cost of sequences using the Levensthein Distance (LD) for each cluster. The Average Levensthein Distance reflects the goodness of the clusters with respect to the intra-cluster

distance. Average Levenshtein Distance (ALD) is expressed as,

$$ALD = \frac{1}{k}\sum_{j=1}^{k}\frac{\sum_{j_l=1}^{|C_j|} LD(\hat{C_j}, C_{j_l})}{|C_j|} \tag{16}$$

ALD computations was carried out on seven big clusters (those clusters whose size is more than 10% of the dataset size). ALD for seven rough clusters is 3.756. This means the sessions preserving the sequential nature have been grouped together. The inter-cluster distance (LD) for the seven clusters are shown in Table 27. Together, these results suggest that the rough set based clustering resulted in clusters with desirable intra-cluster distance (ALD = 3.756) and inter-cluster distances (LD = 3-6).

Table 27: Inter cluster distance in rough set based clustering

	C1	C2	C3	C4	C5	C6	C7
C1	0	3	4	3	3	5	4
C2	3	0	3	5	3	4	**6**
C3	4	3	0	3	4	**6**	4
C4	3	5	3	0	**6**	**6**	5
C5	3	3	4	**6**	0	4	3
C6	5	4	**6**	**6**	4	0	4
C7	4	**6**	4	5	3	4	0

We compared our results with the traditional complete linkage based hierarchical clustering algorithm. To compare the algorithms the sessions falling in the big clusters (1587 sessions) were clustered using the complete linkage clustering algorithm with a frequency based encoding of the sessions.

The ALD value for the complete linkage based hierarchical clustering algorithm is 4.178. Since the length of the sessions is 6, this result indicates that the cost of converting two sequences within a cluster is more than 65%. Thus, it can be seen that complete linkage hierarchical algorithm results in poor intra-cluster distances. Table 28 shows the inter-cluster distances obtained using the complete linkage based hierarchical algorithm. LD which indicates inter-cluster distance value ranges from 2-5 whereas for rough clustering the value was 3-6. It is interesting to note that a maximum value of inter-cluster distance of 6 was observed in the rough set based clustering among four clusters whereas the maximum value

achieved in complete linkage clustering algorithm was 5 and was observed only between two clusters.

These results establish the desirability of the proposed rough set based clustering over complete linkage based hierarchical clustering.

Table 28: Inter cluster distance in complete linkage based hierarchical clustering

	C1	C2	C3	C4	C5	C6	C7
C1	0	3	4	4	3	**2**	4
C2	3	0	3	5	4	**2**	3
C3	4	3	0	3	3	4	**2**
C4	4	5	3	0	3	4	3
C5	3	4	3	3	0	3	**2**
C6	**2**	**2**	4	4	3	0	3
C7	4	3	**2**	3	**2**	3	0

7.6 Chapter Summary

Rough set theory, originally proposed by Pawlak in 1982, has attracted many researchers from various domains and led to successful applications in various fields. In this chapter, we applied the concept of rough sets to cluster objects using the notion of similarity upper approximations. Usually, the clusters resulting from the web usage mining algorithms may not necessarily have crisp boundaries, rather they have fuzzy or rough boundaries [142, 143]. Membership of a record in a cluster may not be precisely defined, that is, a record may be a candidate of more than one cluster. The approach presented in this chapter results in rough clusters and the experiments on user navigation data produced meaningful clusters that enable discovering of navigation patterns. Rough clusters are helpful to get early warnings of potentially significant changes in the clustering patterns.

Traditional clustering methods such as the k-means approach generate groups describing the members of each cluster whereas clustering techniques based on rough set theory generate clusters describing the main characteristics of each cluster [144, 221]. These concepts provide interpretations of different web page visit sessions. Such concepts can be an aid to web miners attempting to describe potentially new groups of web users.

Rough clustering produces more clusters than standard cluster analysis [221].

The number of clusters required to describe the data is a function of the inter-object distance. More clusters means that an object has a higher chance of being in more than one cluster by moving from the lower approximation to the boundary region and reducing the size of the lower approximation.

In this work, we used the S^3M similarity measure for web data, which explicitly exploits the sequential nature of data. S^3M considers both the order of occurrence of items as well as the content of the set. Further, we introduced the concept of constrained-similarity upper approximation and presented a new rough agglomerative clustering approach for sequential data. The proposed approach enables merging of two or more clusters at each iteration and hence facilitates fast hierarchical clustering. We experimented our approach on a web navigation dataset collected from the UCI dataset repository. We successfully performed the experiments to form rough clusters. The size of lower approximations as well as the size of clusters with zero lower approximation were also listed. The proposed clustering approach is useful to mine sequential data and helps web miners in describing the characteristics of potentially new groups of web users. The effectiveness of the proposed clustering approach has been established by comparing with the traditional hierarchical clustering algorithm that uses vector encoding of sequences.

CHAPTER VIII

CONCLUSIONS

Either you decide to stay in the shallow end of the pool or you go out in the ocean. *-Christopher Reeve*

The area of KDD deals with analyzing large data collections to extract interesting, potentially useful, hidden and valid patterns. Data mining is the most important step within the process of KDD. The data being analyzed are either sequential or non-sequential in nature. Generally, sequential data when considered for classification and clustering tasks are converted into frequency vectors thus losing sequential information. Therefore, data mining algorithms should be modified to handle sequential data without losing sequential information embedded in them. This thesis contributes to the development of clustering and classification algorithms for sequential data. A new similarity metric has been devised which considers both the order of occurrence as well as the content information while computing similarity between them. This chapter concludes the thesis by summarizing the proposed classification and clustering algorithms with the proposed similarity measure and presents several directions for future work.

8.1 Summary of contributions

Recent technological advances have tremendously increased the amount of collected data. Besides the sheer amount of collected information, these data exhibit sequentiality. To analyze these sequential data collections, new data mining methods are needed that are capable of drawing maximum advantage of the sequential nature. This thesis contributes to the field of data mining of sequential information by introducing methods for classification and clustering of sequential data. In particular, it provides solutions for important application area namely, computer security and web mining. In the following subsections, we give a summary of these contributions. The main contribution of the thesis can be summarized as

follows.

- **Sequential Information is important.** Researchers while dealing with sequential data usually, convert them into frequency vector treating all the events within the data sequence as independent of each other. Treating sequential data in this manner ignores the embedded sequential information thus resulting in inaccurate classification or clustering performance. The thesis first establishes the hypothesis that classification and clustering tasks, if carried out using sequential information of sequential data, lead to better results irrespective of the distance/similarity measure used. To the best of our knowledge, this has not been established earlier.

- **Content information is also important.** Apart from sequential information content information is also important for sequential data while performing classification and clustering. It was observed that the content information further enhances the capability of classification and clustering algorithms.

- **Devising a new similarity measure for sequential data.** A new similarity measure S^3M (Sequence and Set Similarity Metric) was devised for classification and clustering of sequential data. The designed metric qualifies as a similarity metric in the sense of obeying reflexivity, symmetry and normalization properties.

  S^3M similarity measure performed well for classification and clustering tasks for sequential data.

- **Efficient classifier for classification of abnormal system calls** *kNN* classification with proposed S^3M metric was used on DARPA'98 IDS dataset. We showed the accuracy of the classifier using ROC curve analysis method. The efficiency of *kNN* classifier was compared with the S^3M metric and the *cosine* as well as the *BWC* metric.

- **Design of a new partitional clustering algorithm for sequential data.** A new partitional clustering algorithm with a modification over *PAM* was designed and named as $SeqPAM$. $SeqPAM$ differs from PAM in two ways. Firstly, $SeqPAM$ uses a guided approach for mediod selection. Secondly,

cost optimization function used is the maximization of pairwise similarity of data sequences within the cluster. Additionally, $SeqPAM$ uses S^3M similarity measure thus capturing the embedded sequential information in the data sequences. The goodness of the clusters resulting from both the measures was computed using a cluster validation technique based on Levensthein Distance (*LD*).

The Average Levensthein Distance (*ALD*) values obtained on the *msnbc* dataset using *SeqPAM* clustering algorithm indicate that the groupings have preserved sequential information embedded in the *msnbc* dataset.

We also noted from the *LD* for each cluster that for *SeqPAM* the *LD* value was less than that for *PAM*. *LD* values are indicators of distance between cluster representatives. This *LD* measure indicate that in the *SeqPAM* clustering algorithm the cost of converting a sequence to its cluster representative is less as compared to *PAM* clustering algorithm in turn indicating the tightness of the clusters. All the experiments for clustering algorithms were carried out on benchmarking *msnbc* web navigational dataset.

- **Design of new hierarchical agglomerative clustering algorithm for sequential data.** We designed a new indiscernibility-based rough agglomerative hierarchical clustering algorithm for sequential data. We used S^3M similarity measure for computing similarity between web data sequences. The proposed rough agglomerative clustering approach for sequential data uses the concept of constrained-similarity upper approximation. The proposed approach enables the merging of two or more clusters at each iteration and hence facilitates fast hierarchical clustering. We tested our clustering algorithm on *msnbc* web navigation dataset that are intrinsically sequential in nature. The proposed approach is compared to the traditional complete linkage based hierarchical clustering algorithm.

8.2 Ideas for future work

In chapter 3, we demonstrated the usefulness of sequence as well as the content information for classifying and clustering sequential data. In chapter 4, we proposed a similarity measure that is a linear combination of the length of the longest common subsequence and Jaccard similarity measures. LCS calculation is computationally intensive as it uses dynamic programming approach. Hence a better and efficient combination or an alternative measure capturing sequentiality is to be explored.

The devised measure is for discrete sequential data. In the same line it needs to be modified and tested for continuous sequential data also. We have provided an insight for selection of *p* value (sequence similarity), a component of S^3M measure, but other methods based on the dataset distribution should also be explored.

In all the clustering experiments (in chapters 3, 6 and 7) we used web navigational sequences of length 6 from the *msnbc* dataset. This choice was reasonable since the average length of user navigational sessions was close to 6. However future work could include testing the algorithms with variable sequence length. We proposed a partitional clustering algorithm $SeqPAM$ in chapter 6 where the optimization function used was maximization of pairwise similarity measure. Alternate optimization functions could be devised based on the nature of the datasets.

Chapter 7 presents a hierarchical agglomerative clustering algorithm where the clusters are soft in nature. In the same line, an algorithm could be developed for obtaining hard clusters.

Although the current work is applied in the area of computer security and web usage mining, it can be adopted to other fields where data are of sequential nature such as bioinformatics, speech recognition and image compression.

REFERENCES

[1] A. Abraham and V. Ramos. Web usage mining using artificial ant colony clustering and linear genetic programming. In *Congress on Evolutionary Computation*, pages 1384–1391, Australia, 2003. IEEE Press.

[2] T. Abraham. IDDM: Intrusion detection using data mining techniques. Technical Report DSTO-GD-0286, DSTO Electronics and Surveillance Research Laboratory, Salisbury, South Australia, Australia, May 2001.

[3] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal on Parallel Distributed Computing*, 61(3):350–371, 2001.

[4] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient Similarity Search In Sequence Databases. In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, 1993. Springer Verlag.

[5] R. Agrawal, K. I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *21st International Conference on Very Large Databases*, pages 490–501, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *20th International Conference on Very Large Databases*, pages 487–499, Santiago, Chile, September 1994. Morgan Kaufmann.

[7] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Press.

[8] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the practice of intrusion detection technologies. Technical Report CMU/SEI $-$

$99TR-028$, Carnegie Mellon Software Engineering Institute, Pittsburg, PA, January 2000.

[9] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes. Next-generation intrusion detection expert system (NIDES), software users manual, beta-update release. Technical Report SRI–CSL–95–07, Computer Science Laboratory, SRI International, Ravenswood Avenue, Menlo Park, CA, May 1994.

[10] J. P. Anderson. Computer security threat monitoring and surveillance. Technical Report CONTRACT 79F296400, James P. Anderson Company, Fort Washington, PA, Feburary 1980.

[11] A. Apostolico and C. Guerra. The longest common subsequence problem revisited. *Algorithmica*, 2:315–336, 1987.

[12] J. G. Carbonell B. Y. M. Cheng and J. Klein-Seetharaman. Protein classification based on text document classification techniques. *Proteins*, 58(1):955–970, 2005.

[13] J. S. Balasubramaniyan, J. O. G. Fernandez, D. Isacoff, E. H. Spafford, and D. Zamboni. An architecture for intrusion detection using autonomous agents. Technical Report Coast TR 98-05, Department of Computer Science, Purdue University, June 1998.

[14] A. Banerjee and J. Ghosh. Clickstream clustering using weighted longest common subsequences. In *Web Mining Workshop at the 1st SIAM Conference on Data Mining*, pages 33–40, Chicago, USA, 2001.

[15] A. G. Bchner and M. D. Mulvenna. Discovering internet marketing intelligence through online analytical web usage mining. *SIGMOD Records*, 27(4):54–61, 1998.

[16] S. Berchtold, D. A. Keim, and H. P. Kriegel. The X-tree: An index structure for high-dimensional data. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proceedings of the 22nd International Conference on Very Large Databases*, pages 28–39, San Francisco, U.S.A., 1996. Morgan Kaufmann Publishers.

[17] S. Berchtold, D. A. Keim, and H. P. Kriegel. Using extended feature objects for partial similarity retrieval. *Very Large Databases Journal*, 6(4):333–348, 1997.

[18] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis 2nd edition*. Springer-Verlag, New York, 1985.

[19] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithm. In *Seventh International Symposium on String Processing and Information Retriveal SPIRE 2000*, pages 39–48, Atlanta, 2000. IEEE Computer Society.

[20] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 104–111, New York, USA, 1998. ACM Press.

[21] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[22] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1998.

[23] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, New York, 1984.

[24] H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6(2):153–172, 2002.

[25] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[26] J. B. D. Cabrera, L. Lewis, and R. K. Mehra. Detection and classification of intrusions and faults using sequences of system calls. *SIGMOD Records*, 30(4):25–34, 2001.

[27] I. V. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals and objects. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 140–149, New York, NY, USA, 2000. ACM Press.

[28] I. V. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model-based clustering. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 280–284, New York, USA, 2000. ACM Press.

[29] I. V. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7(4):399–424, 2003.

[30] J. Cannady. Artificial neural networks for misuse detection. In *Proceedings of the 1998 National Information Systems Security Conference*, pages 443–456, Arlington, VA., 1998.

[31] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems*, 27(2):188–228, 2002.

[32] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, San Francisco, 2002.

[33] E. Charniak. *Statistical Language Learning*. MIT Press, Cambridge, MA, USA, 1996.

[34] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): theory and results. In *Advances in knowledge discovery and data mining*, pages 153–180, Menlo Park, CA, USA, 1996. American Association for Artificial Intelligence.

[35] M. Y. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a hidden markov model type stochastic network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):481–496, 1994.

[36] N. Christianini and J. Shawe-Taylor. *An Introduction to support vector machines and other kernel based learning methods*. Cambridge University Press, 2000.

[37] P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Databases*, pages 426–435, Athens, Greece, September 1997.

[38] E. Cohen, B. Krishnamurthy, and J. Rexford. Improving end-to-end performance of the web using server volumes and proxy filters. In *ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 241–253, New York, USA, 1998. ACM Press.

[39] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, 22(16):10881– 10890, 1988.

[40] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learing*, 20(3):273–297, 1995.

[41] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic networks and expert systems*. Springer-Verlag, New York, USA, 1999.

[42] T. Darrell and A. Pentland. Space-time gestures. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 335–340. IEEE Computer Society Press, 1993.

[43] B. V. Dasarathy. *Nearest Neighbor: Pattern Classification Techniques*. IEEE Computer Society Press, Los Alomitos, CA, 1991.

[44] M. Dash, H. Liu, P. Scheuermann, and K. L. Tan. Fast hierarchical clustering and its validation. *Data and Knowledge Engineering*, 44(1):109–138, 2003.

[45] S. K. De. A rough set theoretic approach to clustering. *Fundamenta Informaticae*, 7(2):335–344, 1999.

[46] S. K. De, R. Biswas, and A. R. Roy. Finding dependency of attributes in information system. *Journal of Fuzzy Mathematics*, 64(3-4):409–417, 2004.

[47] S. K. De and P. R. Krishna. Clustering web transactions using rough approximation. *Fuzzy Sets and Systems*, 148(1):131–138, 2004.

[48] H. Debar, M. Becker, and D. A. Siboni. Neural network component for an intrusion detection system. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 240–250, Los Alamitos, CA, May 1992.

[49] A. Douglas, B. Lorna, M. Siety, R. L. Humphreys, and L. Sadler. *Machine Translation: An Introductory Guide.* Blackwells-NCC, London, England, 1994.

[50] D. Dubois and H. Prade. Putting rough sets and fuzzy sets together. In R. Slowinski, editor, *Intelligent Decision Support-Handbook of Applications and Advances of the Rough Set Theory*, pages 203–232. Kluwer Academic, 1992.

[51] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification 2nd Edition*. John Wiley and Sons, New York, 2001.

[52] S. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(4):325–327, 1976.

[53] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of proteins and nucleic acids*. Cambridge University Press, New York, USA, 1999.

[54] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *ACM SIGMOD international conference on Management of data*, pages 419–429, New York, USA, 1994. ACM Press.

[55] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[56] U. M. Fayyad, G. P. Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 82–88, Portland, Oregon, 1996. AAAI Press.

[57] D. F. Feng and R. F. Doolittle. Progressive sequence alignment as a a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, 1987.

[58] E. W. Forgey. Cluster analysis of multivariate data:eciency vs. interpretability of classifications. *Biometrics*, 21(3):768–769, 1965.

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

[60] A. Foss, W. Wang, and O. R. Zaane. A non-parametric approach to web log analysis. In *Workshop on Web Mining in First International SIAM Conference on Data Mining*, pages 41–50, Chicago, Illinois, April 2001.

[61] K. L. Fox, R. R. Henning, J. H. Reed, and R. P. Simonian. A neural network approach towards intrusion detection. In *13th National Computer Security Conference. Information Systems Security. Standards—the Key to the Future*, pages 124–134, July 1990.

[62] Y. Fu, K. Sandhu, and M. Shih. Clustering of web users based on access patterns. In *KDD Workshop on Web Mining*, pages 21 – 38, San Diego, CA., 1999. Springer-Verlag.

[63] J. Gehrke, R. Ramakrishnan, and V. Ganti. RainForest - a framework for fast decision tree construction of large datasets. In *Proceedings of the 24th International Conference on Very Large Databases*, pages 416–427, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[64] Z. Ghahramani. An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):9–42, 2001.

[65] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: an approach based on dynamical systems. *Very Large Databases Journal*, 8(3-4):222–236, 2000.

[66] P. Gludici. *Applied Data Mining , Statistical methods for business and industry*. Wiely publication, 2003.

[67] B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. John Wiley and Sons, New York, USA, 2000.

[68] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, 1982.

[69] O. Gotoh. Consistency of optimal sequence alignments. *Bulletin of mathematical biology*, 52(4):509–525, 1990.

[70] T. Graepel, R. Herbrich, P. B. Sdorra, and K. Obermayer. Classification on pairwise proximity data. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 438–444, Cambridge, MA, USA, 1999. MIT Press.

[71] S. Greco, B. Matarazzo, and R. Slowinski. Rough set processing of vague information using fuzzy similarity relations. In C. S. Calude and G. Paun, editors, *Finite Versus Infinite – Contributions to an Eternal Dilemma*, pages 149–173, London, 2000. LNCS Springer-Verlag.

[72] R. L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. R. Namburu. *Data Mining for Scientific and Engineering Applications*. Kulwer Academic Press, 2001.

[73] Y. Guan, A. Ghorbani, and N. Belacel. Y-means: A clustering method for intrusion detection. In *Canadian Conference on Electrical and Computer Engineering*, pages 616 – 620, Montral, Qubec, Canada, 2003. Springer-Verlag.

[74] V. Guralnik and G. Karypis. A scalable algorithm for clustering sequential data. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 179–186, Washington, DC, USA, 2001. IEEE Computer Society.

[75] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, USA, 1997.

[76] R. W. Hamming. *Coding and Information Theory*. Lawrence Erlbaum Associates, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[77] E. H. Han and G. Karypis. Centroid-based document classification: Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, pages 424–431, Lyon, France,, March 2000. Springer-Verlag.

[78] E. H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Research Issues on Data Mining and Knowledge Discovery*, pages 9–13, Tucson, Arizona, 1997. ACM.

[79] E. H. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph based clustering in high-dimensional data sets: A summary of results. *Data Engineering Bulletin*, 21(1):15–22, 1998.

[80] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[81] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[82] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

[83] B. Hay, G. Wets, and K. Vanhoof. Clustering navigation patterns on a website using a sequence alignment method. In *Intelligent techniques for web personalization: 17th International Joint Conference on Artificial Intelligence*, pages 1–6, Seattle, Washington, USA, 2001.

[84] P. Helman and G. Liepins. Statistical foundations of audit trail analysis for the detection of computer misuse. *IEEE Transactions on Software Enggineering*, 19(9):886–901, 1993.

[85] S. Hirano and S. Tsumoto. Rough clustering and its application to medicine. *Information Science*, 124(1):125–137, 2000.

[86] S. Hirano and S. Tsumoto. Indiscernibility-based clustering : Rough clustering. In *International Fuzzy Systems Association World Congress*, pages 378–386, Heidelberg, 2003. LNCS Springer-Verlag.

[87] S. Hirano and S. Tsumoto. An indiscernibility-based clustering method with iterative refinement of equivalence relations -rough clustering. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 7(2):169–177, 2003.

[88] M. Hirosawa, Y. Totoki, M. Hoshida, and M. Ishikawa. Comprehensive study on iterative algorithms of multiple sequence alignment. *Computational applied biosciences*, 11(1):13–18, 1995.

[89] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communication of ACM*, 18(6):341–343, 1975.

[90] T. B. Ho and N. B. Nguyen. Nonhierarchical document clustering based on a tolerance rough set model. *International Journal of Intelligent Systems*, 17(2):199–212, 2002.

[91] J. Hochberg, K. Jackson, C. Stallings, J. F. McClary, D. DuBois, and J. Ford. NADIR: An automated system for detecting network intrusion and misuse. *Computers and Security*, 12(3):235–248, 1993.

[92] J. Hoey. Clustering contextual facial display sequences. In *FGR '02: Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 354 – 359, Washington, DC, USA, 2002. IEEE Computer Society.

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

[94] W. J. Hsu and M. W. Du. Computing a longest common subsequence for a set of strings. *BIT*, 24:45–59, 1984.

[95] W. Hu, Y. Liao, and V. R. Vemuri. Robust support vector machines for anomaly detection in computer security. In *International Conference on Machine Learning and Applications*, pages 168–174, Los Angeles, CA, July 2003. CSREA Press.

[96] J. W. Hunt and T. G. Szymanski. A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20(5):350–353, 1977.

[97] E. F. Ian and H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Series in Data Management Systems, 1999.

[98] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, 1995.

[99] D. W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):583–600, 2000.

[100] H. V. Jagadish. A retrieval technique for similar shapes. In *ACM SIGMOD international conference on Management of data*, pages 208–217, New York, USA, 1991. ACM Press.

[101] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[102] A. K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1386–1391, 1997.

[103] M. James. *Classification Algorithms*. John Wiely and Sons Inc., New York, 1985.

[104] S. Jamil and S. D. Jitender. Concept approximations based on rough sets and similarity measures. *International Journal on Applied Mathematics and Computer Science*, 11(3):655 – 674, 2001.

[105] R. C. Jancey. Multidimensional group analysis. *Australian Journal on Botany*, 14(1):127–130, 1966.

[106] T. Joachims, D. Freitag, and T. M. Mitchell. Web Watcher: a tour guide for the world wide web. In *International Joint Conference on Artificial Intelligence*, pages 770–777, Nagoya, Japan, August 1997. Morgan Kaufmann.

[107] A. Joshi and R. Krishnapuram. Robust fuzzy clustering methods to support web mining. In S. Chaudhuri and U. Dayal, editors, *Workshop in Data Mining and Knowledge Discovery, SIGMOD*, pages 15:1 – 15:8, Seattle, 1998.

[108] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[109] S. Kawasaki, N. B. Nguyen, and T. B. Ho. Hierarchical document clustering based on tolerance rough set model. In *4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 458–463, London, UK, 2000. LNCS Springer-Verlag.

[110] R. A. Kemmerer. NSTAT: a model-based real-time network intrusion detection system. Technical Report TRCS-97-18, Department of Computer Science, University of California, Santa Barbara, 1998.

[111] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1999.

[112] A. Ketterlin. Clustering sequences of complex objects. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Third International Conference on Knowledge Discovery and Data Mining*, pages 215–218, Newport Beach, CA, USA, 1997. AAAI Press, Menlo Park, California.

[113] M. Khan, Q. Ding, and W. Perrizo. k-nearest neighbor classification on spatial data streams using p-trees. In *6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 517–518, London, UK, 2002. Springer-Verlag.

[114] G. H. Kim and E. H. Spafford. The design and implementation of tripwire: a file system integrity checker. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 18–29, New York, NY, USA, 1994. ACM Press.

[115] J. Kim, S. Pramanik, and M. J. Chung. Multiple sequence alignment using simulated annealing. *Computer Applications in the Biosciences*, 10(4):419–426, 1994.

[116] C. Ko, G. Fink, and K. Levitt. Automated detection of vulnerabilities in privileged programs by execution monitoring. In *1Oth Annual Computer Security Applications Conference*, pages 134–144, Orlando, FL, December 1994. IEEE Computer Society Press.

[117] T. Kohonen. Median strings. *Pattern Recognition Letters*, 3:309–313, 1985.

[118] T. Kohonen. *Self-organizing maps, 2^{nd} Edition*. Springer Verlag, Berlin, 1997.

[119] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of ACM*, 40(3):77–87, 1997.

[120] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler. Hidden markov models in computational biology: Applications to protein modeling. Technical Report UCSC-CRL-93-32, Electronics Institute, Technical University of Denmark, Lyngby, Denmark, August 1993.

[121] M. Kryszkiewicz. Rough set approach to incomplete information systems. *Information Science*, 112(1-4):39–49, 1998.

[122] S. Kumar and E. Spafford. An Application of Pattern Matching in Intrusion Detection. Technical Report 94-013, Department of Computer Sciences, Purdue University, March 1994.

[123] K. Labib and V. R. Vemuri. An Application of Principal Component Analysis to the Detection and Visualization of Computer Network Attacks. *Annals of Telecommunications*, 61(1-2):218 –234, 2006.

[124] MIT Lincoln Laboratory. *Intrusion Detection Datasets*. http://www.ll.mit.edu/IST/ideval/data/data index.html.

[125] M. Law and J. Kwok. Rival penalized competitive learning for model-based sequence clustering. In *Fifteenth International Conference on Pattern Recognition*, pages 195–198, Barcelona, Spain, September 2000.

[126] W. Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Dept of Computer Science, Columbia University, June 1999.

[127] W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, pages 79 – 93, San Antonio, TX, 1998.

[128] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang. Real time data mining-based intrusion detection. In *Second DARPA Information Survivability Conference and Exposition*, pages 85–100, Anaheim, California, June 2001.

[129] W. Lee and D. Xiang. Information-theoretic measures for anomaly detection. In *IEEE Symposium on Security and Privacy*, pages 130 – 143, Oakland, CA, USA, May 2001. IEEE Computer Society.

[130] N. Lesh, M. J. Zaki, and M. Ogihara. Mining features for sequence classification. In S. Chaudhuri and D. Madigan, editors, *Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 342 – 346, San Diego, August 1999. ACM Press.

[131] C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467– 476, 2004.

[132] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science*, pages 333–342, Darlinghurst, Australia, 2005. Australian Computer Society, Inc.

[133] L. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10(7):707–710, 1966.

[134] C. Li and G. Biswas. Clustering sequence data using hidden markov model representation. In B. V. Dasarathy, editor, *SPIE'99 Conference on Data Mining and Knowledge Discovery: Theory, Tools, and Technology*, pages 14–21, Orlando, FL, USA, April 1999. Springer-Verlag.

[135] C. Li and G. Biswas. Temporal pattern generation using hidden markov model based unsupervised classification. In *IDA '99: Proceedings of the Third*

International Symposium on Advances in Intelligent Data Analysis, pages 245–256, London, UK, 1999. Springer-Verlag.

[136] C. Li and G. Biswas. A bayesian approach to temporal data clustering using hidden markov models. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 543–550, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[137] C. Li and G. Biswas. Applying the hidden markov model methodology for unsupervised learning of temporal data. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 3(6):152–160, 2002.

[138] Y. Liao and V. Vemuri. Use of k-nearest neighbor classifier for intrusion detection. *Computers and Security*, 21(5):439–448, 2002.

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

[140] H. Lieberman. Letizia: An agent that assists web browsing. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, Montreal, Quebec, Canada, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[141] K. I. Lin, H. V. Jagadish, and C. Faloutsos. The TV-tree: An index structure for high-dimensional data. *Very Large Databases Journal*, 3(4):517–542, 1994.

[142] P. Lingras and C. West. Interval set clustering of web users with rough k-means. *Journal of Intelligent Information Systems*, 23(1):5–16, 2004.

[143] P. Lingras, R. Yan, and A. Jain. Web usage mining: Comparison of conventional, fuzzy, and rough set clustering. In Y. Zhang and Y. Yao, editors, *Computational Web Intelligence: Intelligent Technology for Web Applications*, pages 133 – 148. World Scientific Press, 2004.

[144] P. Lingras and Y. Y. Yao. Time complexity of rough clustering: Gas versus k-means. In *Third International Conference on Rough Sets and Current Trends in Computing*, pages 263–270, London, UK, 2002. LNCS Springer-Verlag.

[145] R. P. Lippmann. An introduction to computing with neural nets. *SIGARCH Computer Architecture and News*, 16(1):7–25, 1988.

[146] T. F. Lunt. A survey of intrusion detection techniques. *Computers and Security*, 12(4):405–418, 1993.

[147] H. Mannila and C. Meek. Global partial orders from sequential data. In *Sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 161–168, New York, NY, USA, 2000. ACM Press.

[148] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Knowledge Discovery and Data Mining*, pages 146–151, Portland, Oregon, August 1996. AAAI Press.

[149] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining Knowledge Discovery*, 1(3):259–289, 1997.

[150] W. J. Masek and M. S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer Science and Systems*, 20(1):18–31, 1980.

[151] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Workshop on Learning for Text Categorization*, pages 41–48, Madison, WI, 1998. AAAI Press.

[152] R. Mehrotra and J. E. Gary. Feature-based retrieval of similar shapes. In *Proceedings of the Ninth International Conference on Data Engineering*, pages 108–115, Washington, DC, USA, 1993. IEEE Computer Society.

[153] T. M. Mitchell. *Machine learning*. Mc Graw Hill, 1997.

[154] M. A. Mittermayer. Forecasting intraday stock price trends with text mining techniques. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 3*, pages 30064.2/1–30064.2/10, Washington, DC, USA, 2004. IEEE Computer Society.

[155] B. Mobasher, R. Cooley, and J. Srivastava. Creating adaptive web sites through usage-based clustering of urls. In *Workshop on Knowledge and Data Engineering Exchange*, page 19, Washington, DC, USA, 1999. IEEE Computer Society.

[156] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6(1):61–82, 2002.

[157] T. Morzy, M. Wojciechowski, and M. Zakrzewicz. Pattern-oriented hierarchical clustering. In *Advances in Databases and Information Systems*, pages 179–190, Maribor, Slovenia, 1999. Springer-Verlag.

[158] D. W. Mount. *Bioinformatics: Sequence and Genome Analysis 2nd Edition*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 2004.

[159] S. Muggleton and L. D. Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[160] B. L. Mukherjee, T. Heberlein, and K. N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, 1994.

[161] E. W. Myers. An o(nd) difference algorithm and its variations. *Algorithmica*, 2:251–266, 1986.

[162] R. Nag, K. H. Wong, and F. Fallside. Script recognition using hidden markov models. In *IEEE International Conference on Acoustics, Speech, Signal Processing*, pages 2071–2074, 1986.

[163] M. Nakagawa and B. Mobasher. A hybrid web personalization model based on site connectivity. In *Proceedings of WebKDD*, pages 59–70, Washington DC, USA, 2003.

[164] N. Nakatsu, Y. Kambayashi, and S. Yajima. A longest common subsequence algorithm suitable for similar text strings. *Acta Informatica*, 18(2):171 – 179, 1982.

[165] C. L. Ngo and H. S. Nguyen. A method of web search result clustering based on rough sets. In A. Skowron, R. Agrawal, M. Luck, T. Yamaguchi, P. M. Mahoudeaux, J. Liu, and N. Zhong, editors, *ACM International Conference on Web Intelligence*, pages 673–679, Compiegne, France, September 2005. IEEE Computer Society.

[166] D. S. Weng Ngu and X. Wu. Sitehelper: a localized agent that helps incremental exploration of the world wide web. In *Selected papers from the sixth international conference on World Wide Web*, pages 1249–1255, Essex, UK, 1997. Elsevier Science Publishers Ltd.

[167] C. Notredame and D. G. Higgins. SAGA: sequence alignment by genetic algorithm. *Nucleic Acid Research*, 24:1515–1524, 1996.

[168] T. Oates, L. Firoiu, and P. R. Cohen. Using dynamic time warping to bootstrap hmm-based clustering of time series. In *Sequence Learning - Paradigms, Algorithms, and Applications*, pages 35–52, London, UK, 2001. Springer-Verlag.

[169] J. J. Oliver, R. A. Baxter, and C. S. Wallace. Unsupervised Learning using MML. In *Machine Learning: Proceedings of the Thirteenth International Conference (ICML 96)*, pages 364–372. Morgan Kaufmann Publishers, 1996.

[170] N. Osato, M. Itoh, H. Konno, S. Kondo, K. Shibata, P. Carninci, T. Shiraki, A. Shinagawa, T. Arakawa, S. Kikuchi, K. Sato, J. Kawai, and Y. Hayashizaki. A computer-based method of selecting clones for a full-length cDNA project: Simultaneous collection of negligibly redundant and variant cDNAs. *Genome Research*, 12(7):1127–1134, 2002.

[171] S. K. Pal and P. Mitra. Case generation using rough sets with fuzzy representation. *IEEE Transactions on Knowledge and Data Engineering*, 16(3):292–300, 2004.

[172] S. K. Pal and A. Skowron. *Rough Fuzzy Hybridization: New Trends in Decision Making*. LNCS Springer Verlag, New York, 1999.

[173] G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. D. Spyropoulos. Clustering the users of large web sites into communities. In *Seventeenth International Conference on Machine Learning*, pages 719–726, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[174] M. Paterson and V. Danc'ik. Longest common subsequences. In *19th International Symposium Mathematical Foundations of Computer Science*, pages 127–142, Kosice, Slovakia, 1994. LNCS, Springer-Verlag.

[175] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 2:341–346, 1982.

[176] Z. Pawlak. *Rough Sets – Theoretical aspects of reasoning about data*. Kulwer Academic, 1991.

[177] E. Pekalska and R. P. W. Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23(8):943–956, 2002.

[178] E. Pekalska and R.P.W. Duin. Automatic pattern recognition by similarity representations. *Electronic Letters*, 37(3):159–160, 2001.

[179] F. Peng, D. Shuurmans, and S. Wang. Augmenting naive bayes classifier using statistical n-gram language modeling. *Information Retrieval*, 34(7):317–345, 2004.

[180] M. Perkowitz and O. Etzioni. Adaptive web sites: an AI challenge. In *Fifteenth International Joint Conference on Artificial Intelligence*, pages 16–23, Nagoya, Japan, 1997. Morgan Kaufmann.

[181] M. Perkowitz and O. Etzioni. Adaptive web sites. *Communications of ACM*, 43(8):152–158, 2000.

[182] P. Pevzner and M. S. Waterman. Multiple filtration and approximate pattern matching. *Algorithmica*, 13(1/2):135–154, 1995.

[183] P. A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, Cambridge, Mass, 2000.

[184] D. Pierrakos, G. Paliouras, C. Papatheodorou, and C. D. Spyropoulos. Web usage mining as a tool for personalization: A survey. *User Modeling and User-Adapted Interaction*, 13(4):311–372, 2003.

[185] J. E. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *USENIX Symposium on Internet Technologies and Systems*, pages 139–150, Boulder, Colorado, October 1999.

[186] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGS for multiclass classification. In S. A. Solla, T. K. Leen, and K. R. Mueller, editors, *Advances in Neural Information Processing Systems*, pages 547–553, Cambridge, 2000. MIT Press.

[187] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research Technology, Redmond, Washington, April 1998.

[188] L. Polkowski, A. Skowron, and J. Zytkow. Tolerance based rough sets. In T. Y. Lin and A. M. Wildberger, editors, *Soft Computing: Rough Sets, Fuzzy Logic, Neural Networks, Uncertainty Management*, pages 55–58. San Diego: Simulation Councils, Inc., 1995.

[189] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th NIST-NCSC National Information Systems Security Conference*, pages 353–365, Baltimore, Maryland, October 1997. NIST/NCSC.

[190] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *ACM Workshop on Data Mining Applied to Security*, pages 1–14, Philadelphia, USA, November 2001.

[191] G. Qian, S. Sural, Y. Gu, and S. Pramanik. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1232–1237, New York, USA, 2004. ACM Press.

[192] J. R. Quinlan. *C4.5: Programms of Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[193] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, USA, 1993.

[194] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[195] S. Rawat, V. P. Gulati, A. K. Pujari, and V. R. Vemuri. Intrusion detection using processing techniques with a binary-weighted gosine metric. *Journal of Information Assurance and Security*, 1(1):43–58, 2006.

[196] C. Rick. New algorithms for the longest common subsequence problem. Technical Report 85123–CS, Department of Computer Science, University of Bonn, Germany, October 1994.

[197] D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*. Reading, MA: Addison-Wesley Publishing Company, 1983.

[198] M. Sarkar. Rough-fuzzy functions in classification. *Fuzzy Sets and Systems*, 132(3):353–369, 2002.

[199] G. Schultz, C. Endorf, and J. Mellander. *Intrusion detection and prevention*. McGraw-Hill, New York: Osborne Media, 2003.

[200] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[201] M. M. Sebring, E. Shellhouse, M. Hanna, and R. Whitehurst. Expert systems in intrusion detection: A case study. In *11th National Computer Security Conference*, pages 74–81, Baltimore, MD, October 1988.

[202] C. Shahabi, A. M. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users web-page navigation. In *7th International Workshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications*, pages 20–29, Birmingham, England, April 1997. IEEE Computer Society.

[203] I. Simon. Sequence comparison: some theory and some practice. In M. Gross and D. Perrin, editors, *Electronic Dictionaries and Automata in Computational*

Linguistics, pages 79–92, Saint Pierre d'Oléron, France, 1987. Springer-Verlag, Berlin.

[204] R. Slowinski and D. Vanderpooten. Similarity relations as a basis for rough approximations. Technical Report ICS Research Report 53/95, Polish Academy of Sciences, Poland, 1995.

[205] R. Slowinski and D. Vanderpooten. A generalized definition of rough approximations based on similarity. *IEEE Transactions on Data and Knowledge Engineering*, 12:331–336, 2000.

[206] P. Smyth. Clustering sequences with hidden markov models. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, pages 648 – 654. The MIT Press, 1997.

[207] P. Smyth. Probabilistic model-based clustering of multivariate and sequential data. In *Artificial Intelligence and Statistics*, pages 299–304. Morgan Kaufman, San Mateo CA, 1999.

[208] A. Somayaji and S. Forrest. Automated response using System-Call delays. In *9th USENIX Security Symposium*, pages 185–198, Denver, Colorado, August 2000. ACM Press.

[209] M. Spiliopoulou and L. C. Faulstich. WUM: A tool for Web utilization analysis. In *6th International Conference on Extending Database Technology*, Valencia, Spain, March 1999. Springer Verlag.

[210] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In Peter M. G. Apers, Mokrane Bouzeghoub, and Georges Gardarin, editors, *5th International Conference on Extending Database Technology*, pages 3–17. Springer-Verlag, 1996.

[211] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – A graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, pages 361–370, Baltimore, Maryland, October 1996.

[212] T. Starner. Visual recognition of american sign language using hidden markov models. Master's thesis, Program in Media Arts, 1995.

[213] R. Sun and C. L. Giles. *Sequence Learning: Paradigms, Algorithms, and Applications*. Lecture Notes in Artificial Intelligence, Springer, 2001.

[214] K. M. C. Tan and R. A. Maxion. "why 6?" defining the operational limits of stide, an anomaly-based intrusion detector. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 188 – 201, Washington, DC, USA, 2002. IEEE Computer Society.

[215] H. S. Teng, K. Chen, and S. C. Y. Lu. Security audit trail analysis using inductively generated predictive rules. In *Proceedings of the sixth conference on Artificial intelligence applications*, pages 24–29, Piscataway, NJ, USA, 1990. IEEE Press.

[216] A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In *Third International Workshop on Recent Advances in Intrusion Detection*, pages 80–92, London, UK, 2000. Springer-Verlag.

[217] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[218] M. Vingron and P. Argos. A fast and sensitive multiple sequence alignment algorithm. *Computational Applied Biosciences*, 5(2):115–121, 1989.

[219] M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *Journal of Molecular Biology*, 218(1):33–43, 1991.

[220] T. K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):81–88, 1968.

[221] K. E. Voges, N. K. Ll. Pope, and M. R. Brown. Cluster analysis of marketing data examining online shopping orientation: A comparison of k-means and rough clustering approaches. In H. A. Abbass, R. A. Sarker, and C. S. Newton, editors, *Heuristics and Optimization for Knowledge Discovery*, pages 207–224. Idea Group Publishing, Hershey, 2002.

[222] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of ACM*, 21(1):168–173, 1974.

[223] J. T. L. Wang, M. J. Zaki, H. T. T. Toivonen, and D. Shasha. *Data mining in Bioinformatics*. Springer-Verlag, 2005.

[224] W. Wang and O. R. Zaane. Clustering web sessions by sequence alignment. In *13th International Workshop on Database and Expert Systems Applications*, pages 394–398, Washington, DC, USA, 2002. IEEE Computer Society.

[225] S. M. Weiss and C. A. Kulikowski. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems (Machine Learning Series)*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1991.

[226] A. Wespi, M. Dacier, and H. Debar. Intrusion detection using variable-length audit trail patterns. In *Third International Workshop on Recent Advances in Intrusion Detection*, pages 110–129, London, UK, 2000. Springer-Verlag.

[227] I. H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[228] Y. Xiong and D. Y. Yeung. Mixtures of arma models for model-based time series clustering. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, pages 717 – 737, Washington, DC, USA, 2002. IEEE Computer Society.

[229] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[230] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in timesequential images using a hidden markov model. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 379–385, Champaign, IL, June 1992.

[231] C. Yan, D. Dobbs, V. Honavar, and D. Dobbs. A two-stage classifier for identification of proteinprotein interface residues. *Bioinfromatics*, 20(S1):i371–i378, 2004.

[232] T. W. Yan, M. Jacobsen, H. G. Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. In *Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems*, pages 1007–1014, Amsterdam, Netherlands, 1996. Elsevier Science Publishers.

[233] C. Zhang, J. Jiang, and M. Kamel. Intrusion detection using hierarchical neural networks. *Pattern Recognition Letters*, 26(6):779–791, 2005.

[234] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. pages 103–114, Montreal, Canada, June 1996. ACM Press.

APPENDIX A

DARPA '98 IDS DATASET

In this appendix we present the description of the *DARPA'98 IDS* dataset [124]. All the experimentation reported in the thesis for classification task is performed on *DARPA'98 IDS* dataset.

A.0.1 Collection of the datset

DARPA'98 IDS dataset contains seven weeks of training data and two weeks of testing data. Information about when a specific attack started and ended is provided for the training data only such that the performance of a the algorithm is evaluated during training phase. The simulated network represents thousands of UNIX hosts and hundreds of users. There are three UNIX machines designated as victim machines running on three different operating systems: SunOS, Solaris OS, and Linux.

A.0.2 Description of the dataset

The data set contains more than three hundred instances of thirty eight different attacks from four attack categories. Out of the thirty eight attacks, only fourteen were present in test data. The four attack categories are:

- TCP packets: Network level information
- BSM audit data: Host level information for Solaris OS
- File system dumps: Copies of system directories (generated every day)
- PS output: Lists of running processes (generated every minute)

We used the Basic Security Module (BSM) audit data collected from a victim Solaris machine. The BSM audit logs contain information on system calls produced by programs running on the Solaris machine.

Table 29: List of 55 attacks used in testing dataset

$1.1itffbclear$	$1.1itformatclear$	$2.2itipsweep$
$2.5itftpwrite$	$2.5itftpwritetest$	$3.1itffbclear$
$3.3itftpwrite$	$3.3itftpwritetest$	$3.4itwarez$
$3.5itwarezmaster$	$4.1it080520warezclient$	$4.2it080511warezclient$
$4.2it153736spy$	$4.2it153736spytest$	$4.2it153812spy$
$4.4it080514warezclient$	$4.4it080514warezclienttest$	$4.4it175320warezclient$
$4.4it180326warezclient$	$4.4it180955warezclien$	$4.4it181945warezclient$
$4.5it09221fb2f$	$4.5it141011loadmodule$	$4.5it162228loadmodule$
$4.5it174726loadmodule$	$4.5itformat$	$5.1it141020ffb$
$5.1it174729ffbexec$	$5.1itformat$	$5.2it144308ejectclear$
$5.2it163909ejectclear$	$5.3itejectsteal$	
$5.5iteject$	$5.5itfdformat$	$5.5itfdformatchmod$
$6.4it090647ffb$	$6.4it093203eject$	$6.4it095046eject$
$6.4it100014eject$	$6.4it122156eject$	$6.4it144331ffb$
$test.1.2format$	$test.1.2format2$	$test.1.3eject$
$test.1.3httptunnel$	$test.1.4eject$	$test.1.5processtable$
$test.2.1111516ffb$	$test.2.1format$	$test.2.2xsnoop$
$test.2.3ps$	$test.2.3psb$	$test.2.5ftpwrite$
$test.2.4ejecta$	$test.2.2format1$	

Table 30: List of normal system calls

access	audit	auditon	chdir	chmod
chown	close	creat	execve	exit
fchdir	fchown	fcntl	fork	fork1
getaudit	getmsg	ioctl	kill	link
login	logout	lstat	memcntl	mkdir
mmap	munmap	nice	open	pathconf
pipe	putmsg	readlink	rename	rmdir
setaudit	setegid	seteuid	setgid	setgroups
setpgrp	setrlimit	setuid	stat	statvfs
su	sysinfo	unlink	utime	vfork

```
Thu Aug 10 22:01:29 2004 -> UID:root EUID:root RUID:root - From machine:log1
execve() + /usr/bin/sparcv7/ps + cmdline:ps,-ef + success
Thu Aug 10 22:01:50 2004 -> UID:root EUID:root RUID:root - From machine:log1
execve() + /usr/bin/tail + cmdline:tail,/etc/system + success
Thu Aug 10 22:11:18 2004 -> UID:root EUID:root RUID:root - From machine:log1
execve() + /usr/bin/pwd + cmdline:pwd + success
Thu Aug 10 22:11:20 2004 -> UID:root EUID:root RUID:root - From machine:log1
execve() + /usr/bin/ls + cmdline:ls,-l + success
Thu Aug 10 22:11:33 2004 -> UID:root EUID:root RUID:root - From machine:log1
execve() + /usr/bin/ls + cmdline:ls,-l + success
```

Figure 19: Example BSM system log data

From the training dataset, we extracted the 50 unique system calls. Table 30 lists all the normal system calls. For each day of data, a separate BSM file is provided with the 'BSM List File'. A sample BSM system log file data is shown in Figure 19. Each line of this file contains the information about one session such as time, service, source IP and destination IP. A '0' at the end of the line shows that the session is normal and the presence of a '1' at the end of the line shows the session as intrusive. All the intrusive sessions are labelled with the name of the attacks launched during the sessions. BSM commands *auditreduce* and *praudit* and a couple of shell scripts were used to extract the data. Five days were identified from whole list which were free from any type of attacks. These are, Tuesday of the third week, Thursday of the fifth week and Monday, Tuesday and Wednesday of the seventh week.

The first four days of data were chosen for creating the normal training dataset and the fifth day's data for creating the normal test dataset.

There are around *2000* normal sessions reported in the four days of data. We extract the processes occurring during these days and our normal dataset consists of *606* unique processes. There are *412* normal sessions on the fifth day and we extract *5285* normal processes for the testing data. In order to test the detection capability of our method, we incorporate *55* intrusive sessions into our test dataset. Table 29 lists all the attacks. A number in the beginning of the name denotes the week and day followed by the name of the attack. For example in an attack $1.1itffbclear$, *1* stands for first week second *1* stands for 1^{st} day that is, Monday and *ffb* is the attack in clear mode. These attack sessions consist of almost all types of attacks launched on the victim Solaris machine (in the simulated DARPA setup) during seven weeks of training period and two weeks of testing period and that can be detected using BSM logs.

APPENDIX B

WEB NAVIGATION DATASET

In this appendix, we describe the necessary preprocessing steps of the *msnbc* web navigation dataset. All the clustering results reported in this paper is performed on *msnbc* web navigation dataset. The appendix concludes with the description of the dataset used for experimentation.

B.0.3 Data Preprocessing

A prerequisite step in all of the techniques for providing users with recommendations is the identification of a set of user sessions from the raw usage data provided by the web server. Ideally, each user session gives an exact account of who accessed the web site, what pages were requested and in what order, and for how long each page was viewed.

In addition to identify user sessions, the raw log must also be cleaned or transformed into a list of page views. Cleaning the server log involves removing all of the file accesses that are redundant, leaving only one entry per page view. This includes handling page views that have multiple frames and dynamic pages that have the same template name for multiple page views. It may also be necessary to filter the log files by mapping the references to the site topology induced by physical links between pages. This is particularly important for usage-based personalization, since the recommendation engine should not provide dynamic links to "out-of-date" or non-existent pages.

Each user session can be thought of in two ways: either as a single transaction of many page references, or as a set of many transactions each consisting of a single page reference. The goal of transaction identification is to dynamically create meaningful clusters of references for each user. Based on an underlying model of the user's browsing behavior, each page reference can be categorized as a content reference, auxiliary (or navigational) reference, or hybrid. In this way, different types of transactions can be obtained from the user session file, including

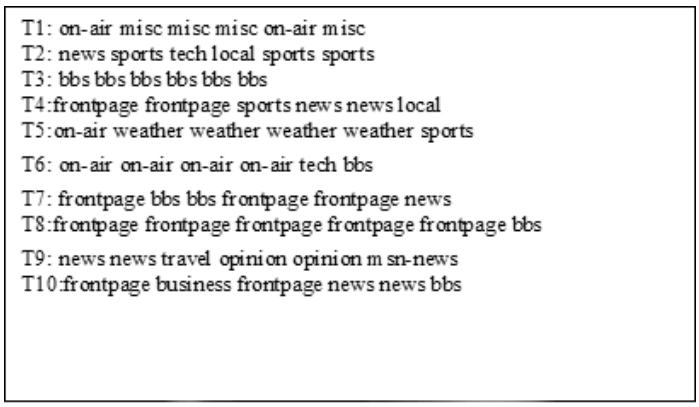

Figure 20: Example web navigation data

content-only transactions involving references to content pages and navigation-content transactions involving a mix of page types. The details of methods for transaction identification are discussed in Cadez et.al. [28]. For the purpose of this paper, we assume that each user session is viewed as a single transaction containing reference to multiple pages in a session. Finally, the session file may be filtered to remove very small transactions.

B.0.4 Description of the Dataset

We collected data from the UCI dataset repository [http://kdd.ics.uci.edu/] that consists of Internet Information Server (IIS) logs for *msnbc.com* and news-related portions of *msn.com* for the entire day of September 28, 1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are not recorded at the finest level of detail but at the level of page categories as determined by the site administrator. There are 17 page categories, namely, 'frontpage', 'news', 'tech', 'local', 'opinion', 'on-air', 'misc', 'weather', 'health', 'living', 'business', 'sports', 'summary', 'bbs' (bulletin board service), 'travel', 'msn-news' and 'msn-sports'. Table 31 shows the characteristics of the dataset.

Each page category is represented by an integer label. For example, 'frontpage' is coded as 1, 'news' as 2, 'tech' as 3, etc. Each row describes the hits of a single

Table 31: Description of the *msnbc* dataset

Total Dataset	
Number of users	989,818
Minimum session length	1
Maximum session length	500
Average number of visits per user	5.7

user. For example, the fourth user hits 'frontpage' twice, and the second user hits 'news' once and so on as shown in Figure 20.

In the total dataset, the length of user sessions ranges from 1 to 500 and the average length of session is 5.7. Keeping this in mind, we randomly selected 5,000 sessions from the preprocessed dataset (consisting of 44,062 sessions) all of length 6 for our experimentation.

APPENDIX C

FOLKFORE ALGORITHM FOR FINDING LONGEST COMMON SUBSEQUENCE

Longest Common Subsequence (LCS) problem can be solved by a simple folklore algorithm based on a dynamic programming approach. Folklore algorithm has been studied and improved by many researchers from various fields of computer science, biology and mathematics. The very first publication of this algorithm is due to Vintsyuk [220].

We present below the basic dynamic programming algorithm for computing the LCS between two sequences (namely, $X = x_1, x_2, \ldots, x_M$ and $Y = y_1, y_2, \ldots, y_N$) which is mapped in a matrix of $M + 1$ rows and $N + 1$ columns. Rows 1 to M are associated with the elements of sequence X, while columns 1 to N are associated with the elements of sequence Y. The first row and column, with index 0, are associated with no elements and are initialized to 0. Entry LCS(i, j) represents the LCS between the first i characters of X and the first j characters of Y, and can be incrementally computed from LCS(i - 1, j), LCS(i - 1, j -1) and LCS(i, j -1) by means of the following recursive equation:

$$LCS(i,j) = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ \max|LCS(i-1,j),\ LCS(i,j-1)| & if\ x_i \neq y_j\ and\ i,\ j > 0 \\ |LCS(i-1,j-1)| + 1 & if\ x_i = y_j\ and\ i,\ j > 0 \end{cases} \tag{17}$$

The entry $LCS(M, N)$, computed in $M \times N$ steps, represents the LCS between sequences X and Y. In order to construct the LCS itself, one needs to backtrack starting from the (M, N) entry in the LCS matrix. This can potentially take $O(M + N)$ steps.

The time complexity of folklore algorithm is of major concern to computer scientists. It requires time proportional to the product of the length of two sequences.

C.0.5 An Example

Consider two sequences $X = abbababa$ and $Y = ababbabba$. The algorithm to compute the LCS among the two sequences X and Y is shown in algorithm 7. The algorithm is of general lengths of sequences assumed here of lengths m and n respectively. C is the LCS matrix constructed using the recursive equation 17. The value at the rightmost bottom cell gives the value of length of the longest common subsequence. Table 32 shows the lcs matrix for two sequences X and Y of the example. The alphabets shown in the bold is the LCS of the two sequences. The LCS in this example is $bababa$.

Algorithm 7 Algorithm for finding Longest Common Subsequence

Input:
Sequence X of length m
Sequence Y of length n
Output:
Length of the lcs
Begin
for i =0 to i =m **do**
C[i,0] $\leftarrow$ 0
end for
for j =0 to j =n **do**
C[0,j] $\leftarrow$ 0
end for
for i =0 to i =m **do**
for j =1 to j =n **do**

if $x_i = y_j$ **then**
C[i,j] $\leftarrow$ c[i-1,j-1] + 1
else if C[i-1,j] $\geq$ C[i, j-1] **then**
C[i,j] $\leftarrow$ C[i-1,j]
else
C[i,j] $\leftarrow$ C[i,j-1]
end if
end for

end for
RETURN C[m,n]
End

Table 32: Longest Common Subsequence Matrix

	j	0	1	2	3	4	5	6	7	8	9
i		y	a	**b**	**a**	**b**	b	**a**	**b**	b	**a**
0	x	0	0	0	0	0	0	0	0	0	0
1	**b**	0	0	**1**	1	1	1	1	1	1	1
2	**a**	0	0	1	**2**	2	2	2	2	2	2
3	a	0	1	1	2	2	2	3	3	3	3
4	**b**	0	1	2	2	**3**	3	3	4	4	4
5	**a**	0	1	2	3	3	3	**4**	4	4	5
6	**b**	0	1	2	3	4	4	4	**5**	5	5
7	**a**	0	1	2	3	4	4	5	5	5	**6**
8	a	0	1	2	3	4	5	5	6	6	6

C.0.6 Improving the time complexity of the LCS algorithm

LCS problem can be solved by reducing it to longest increasing subsequence (LIS) problem. LIS problem finds the longest subsequence in a strictly increasing order of the sequence. In LIS problem, for each symbol that occurs once in first sequence, a list is created of the positions where these symbols occur in second sequence. This list is prepared in decreasing order. For two sequences made up from two different unique set of symbols of the alphabets will have disjoint lists. Then a list is created called $\prod$ of length r, in which each symbol instance in first sequence is replaced with the associated list for that symbol. For two sequences S_1 and S_2 $\prod(S_1,S_2)$ consists of r integers. If the length of longest increasing subsequence is l then finding the length of longest increasing subsequence problem can be solved in $O(rlogl)$ time. For two sequences, of lengths n and m where, $n \leq m$ the longest increasing subsequence is less than or equal to n. Thus, longest common subsequence problem can be solved in $O(rlogn)$ time [75].

Definition C.1 [75]. Given string S_1 and S_2 of lengths m and n respectively over an alphabet $\sum$, let $r(i)$ be the number of times that the character of string S_1 apperas in string S_2.

Definition C.2 [75]. Let r denote the sum $\sum_{i=1}^{m} r(i)$.

Consider the two sequences $X = abbababa$ and $Y = ababbabba$. Then $r(1) = 4$, $r(2) = 5$, $r(3) = 5$, $r(4) = 4$, $r(5) = 5$, $r(6) = 4$, $r(7) = 5$, $r(8) = 4$. Hence $r = 36$. For any two sequences r will fall between 0 and nm. Hence, we can solve the LCS

problem in $O(rlogn)$ time where $n \leq m$. r is often substantially small than nm, depending on the alphabet $\sum$.

APPENDIX D

ALGORITHM FOR LEVENSHTEIN DISTANCE

The Levenshtein distance or edit distance algorithm is not new. It was first published in 1966 by Levenshtein [133]. It is an approximate sequence matching algorithm, that is used to solve the problem of finding sequences $p = p_1 p_2$ in another sequence $s = s_1 s_2 \dots s_n$ which have at most $e > 0$ differences between two sequences. The basic principle of the algorithm is to measure the similarity between two sequences. This is done by calculating the number of basic operations necessary i.e. insertion, deletion and substitution in order to make the two sequences equal. The Levenshtein distance $d(p, m)$ between two sequences p and m is the minimum number of these edit operations (insertion, deletion and substitution) required to make two sequences similar. For example, if p = "moose" and m = "moody" the Levenshtein distance between sequences p and m is $d(p, m) = 2$ as the sequences can be made equal by using two substitutions. There are several versions of the algorithm, but the version used in this thesis is based on the dynamic programming algorithm.

We present below the basic dynamic programming algorithm for computing the levensthein distance between two sequences (namely, $m = m_1, m_2, \dots, m_k$ and $n = n_1, n_2, \dots, n_l$) can be mapped on a matrix of $k + 1$ rows and $l + 1$ columns, where k and l are the lengths of two sequences m and n respectively. This matrix is known as Levensthein Distance matrix and is abbreviated, as LD matrix. Rows of LD matrix 1 to k are associated with the elements of m, while columns 1 to l are associated with the elements of n. The first row and column of LD matrix, with index 0, are associated with no elements and are initialized to $0 \dots k$ and $0 \dots l$ respectively (see Table 33). LD(i, j) entry in the LD matrix represents the distance between the first i characters of m and the first j characters of n, and can be incrementally computed from LD(i - 1, j), LD(i - 1, j -1) and LD(i, j -1) by means of the following recursive equation:

$$\mathrm{LD(i,j)} = \begin{cases} 0 & if\ \ i = 0\ \ or\ \ j = 0 \\ \min \begin{cases} LD(i-1,j)+1 \\ LD(i,j-1)+1 \\ LD(i-1,j-1)+cost \end{cases} & if\ \ i \neq 0\ \ and\ \ j \neq 0 \end{cases} \tag{18}$$

where, $cost = \begin{cases} 1 & if\ \ m_i \neq n_j\ \ and\ \ i,j > 0 \\ 0 & if\ \ m_i = n_j\ \ and\ \ i,j > 0 \end{cases}$

The entry $LD(k,l)$, computed in $k \times l$ steps, represents the LD between sequences m and n.

D.0.7 An Example

Consider two sequences $m = abbababa$ and $n = ababbabba$. The algorithm to compute the LD among the two sequences m and n is shown in algorithm 8. The algorithm is of general lengths of sequences assumed here of lengths k and l respectively. LD is the LD matrix constructed using the recursive equation 18. The value at the rightmost bottom cell gives the levensthein distance between sequences. Table 33 shows the LD matrix for two sequences m and n of the example. The levensthein distance in this example is 3.

Table 33: Levensthein Distance Matrix

i	j	0 y	1 a	2 b	3 a	4 b	5 b	6 a	7 b	8 b	9 a
0	x	0	1	2	3	4	5	6	7	8	9
1	b	1	1	1	2	3	4	5	6	7	8
2	a	2	1	2	1	2	3	4	5	6	7
3	a	3	2	1	2	2	3	3	4	5	6
4	b	4	3	2	3	2	2	3	3	4	5
5	a	5	4	3	2	3	3	2	3	4	4
6	b	6	5	4	3	2	3	3	2	3	4
7	a	7	6	5	4	3	3	3	3	3	3
8	a	8	7	6	5	4	4	3	4	4	**3**

The complexity of the algorithm is $O(k * l)$, where k and l are the lengths of two sequences. Space-complexity of the algorithm is $O(k * l)$, if complete matrix is kept. However, this may be reduced by allocating only two columns in the matrix,

Algorithm 8 Algorithm for finding Levensthein Distance

Input:
Sequence m of length k
Sequence n of length l
Output:
Levensthein distance between sequences m and n
Begin
for i =0 to i = k **do**
LD[i,0] ← i
end for
for j =0 to j = l **do**
LD[0,j] ← j
end for

if $m_i \neq n_i$ **then**
Cost = 1
else
Cost = 0
end if
for i = 0 to i = m **do**
for j = 1 to j = n **do**
LD[i,j] ← Min { LD[i-1,j] + 1, LD[i,j-1]+1, LD[i-1,j-1] + Cost}
end for

end for
RETURN LD[k,l]
End

and reuse this in order to reduce the space complexity to $O(k)$, where $k \geq l$ if only the distance score is needed.

LIST OF PUBLICATIONS

1 Kumar, P., Krishna, P. R., Bapi, R. S. and De. S. K. : *Rough clustering of sequential data*, Journal of Data and Knowledge Engineering 2006. (Under Revision).

2 Kumar, P., Krishna, P. R., and Bapi, R. S.: *SeqPam: A clustering algorithm for sequential data*, International Journal of Data Warehousing and Mining 2006. (Accepted)

3 Kumar, P., Krishna, P. R., Bapi, R. S. and De, S. K. : *Clustering using Similarity Upper Approximation*, In IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2006, Vancouver, Canada, pages 4230 - 4235, July 2006.

4 Kumar, P., Rao, M. V., Krishna, P. R., and Bapi, R. S. :*Using Sub-sequence Information with kNN for Classification of Sequential Data*. In Proceedings of International Conference on Distributed Computing and Internet Technology (ICDCIT), LNCS Springer Verlag, Vol 3816, Bhubneshwar, India, December 2005, pp. 536-546.

5 Kumar, P., Rao, M. V., Krishna, P. R., Bapi, R. S., and Laha, A. :*Intrusion detection system using sequence and set preserving metric.* In Proceedings of IEEE International Conference on Intelligence and security informatics, IEEE-ISI 2005. LNCS Springer Verlag, Vol 3495, Atlanta, Georgia, May 2005, pp. 498-504.

6 Kumar, P., Krishna, P. R., Bapi, R. S. and De, S. K. :*Web Usage Mining Using Rough Agglomerative Clustering*. In proceedings of 7^{th} International Conference on Enterprise Information Systems (ICEIS (2)), Miami, USA, May 2005, pp. 315-320.

VITA

March 24, 1977	Born at Bokaro Steel City, Jharkhand, INDIA
2001	Graduated in Computer Science and Engineering from Magadh University, Bodh Gaya, INDIA
2003	M.Tech (Computer Science) from B.I.T, Mesra, Ranchi, INDIA
2006	Ph.D (Computer Science) from University of Hyderabad, INDIA (Thesis Submitted)

PROFESSIONAL EXPERIENCE

July 2003 - till Date	Institute for Development and Research in Banking Technology, Hyderabad, INDIA

AREA OF INTEREST

Data Mining, Web Mining, Computer Security

The man who follows the crowd will usually get no further than the crowd. The man who walks alone is likely to find himself in places no one has ever been.

- Alan Ashley-Pitt