ON-LINE FAST KERNEL BASED METHODS FOR CLASSIFICATION OVER STREAM DATA (WITH CASE STUDIES FOR CYBER-SECURITY)

YE (GARY) CHEN

A thesis submitted to
Auckland University of Technology
in fulfillment of the requirements for the degree of
Doctor of Philosophy (PhD)

February, 2012

Knowledge Engineering and Discovery Research Institute
Faculty of Design & Creative Technologies

Primary Supervisor:	Prof. Nikola Kasabov
Secondary Supervisor:	Dr. Russel Pears
Third Supervisor:	Dr. Tao Ban

CONTENTS

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extend has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Auckland, February, 2012

Ye (Gary) Chen

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I have the pleasure to acknowledge here some of the many people who have inspired, supported, and educated me over the past three years. First and foremost, I am very grateful to my primary supervisor *Prof. Nikola Kasabov* who gave me the chance to start my PhD at Knowledge Engineering and Discovery Research Institute (KEDRI) which is a high-profile, well known and successful international research group.

I must especially thank the National Institution of Information and Communication Technology (NICT) who granted me a scholarship during the first two and a half years of my PhD study. The KEDRI PhD Scholarship supported me for the last but most important 6 months study.

I wish to express my sincere appreciation to *Dr. Shaoning Pang*, who used to be my primary supervisor, and who enlightened me at the beginning of my postgraduate study.

Many thanks to my secondary supervisor *Dr. Russel Pears* and to my third supervisor *Dr. Tao Ban* , whose enthusiasm, patience and intellectual spirits have been invaluable for my personal and professional growth.

I am very indebted to *Joyce D'Mello*, the manager and the soul of the KEDRI for all her kindness and the tremendous support she has offered me since I joined KEDRI.

I would also like to thank the past and present members of KEDRI at the Auckland University of Technology, who provided a vibrant and multicultural research environment. I am especially grateful to the members of NICT project group:

1. *Lei Song* who worked with me on visualisation systems;
2. *Nuwan Amila Gunasekara* who helped me on string kernels;
3. *Kshitij Dhoble* who developed active learning methods together with me and corrected my writings;

4. *Yiming Peng*, *Fan Liu*, *Alex Song*, *Chelsea Ji* and *Bassie Chen* who helped to finish the NICT project successfully.

I would also like to acknowledge a few people with whom I had many productive discussions during the course of my study: to *Dr. Raphael Hu* for the many conversations on technical and family issues; *Dr Stefan Schliebs* for his jokes which relaxed me during the breaks; *Dr Ammar Mohemmed*, *Linda Liang*, *Marin Karaivanov*, *Harya Widiputra*, *Haza Nuzly* and *Nuttapod Nuntalid* for many scientific and non-scientific discussions.

Last but not least, I am delighted to thank my *wife* Yu Yingying and my *parents* who supported me for almost 10 years since I first came to New Zealand. Without their supports and encouragement, I would not have been able to complete my study.

ABSTRACT

This thesis proposes and presents several novel methods to address some of the real world stream data modelling issues through the use of global and local modeling approaches. A set of real world stream data modelling issues such as dealing with large size and, high dimensionality data, skewed class distribution, different formats of data and visualisation problem are reviewed and their impact on various models are analysed.

The thesis has made nine major contributions to information science, that include four evolving modelling methods, three real world application systems that apply these methods and two stream data visualisation software prototypes. Four novel methods have been developed and published in the course of this study. They are: (1) Online Core Vector Machines (OCVM); (2) Hierarchical CVMs (HCVM) - a local modelling system based on hierarchical labelling data; (3) Dynamic Evolving CVMs (DE-CVM) - a kernel based dynamic evolving learning system; (4) Meta-Learning String Kernel CVM.

OCVM addresses the issue of one-pass, large size, high dimensionality stream data through a kernel-based online learning process. OCVM is proposed for large-scale classification by leveraging connections between learning and computational geometry. It imposes the constraint that only a single pass over the data is allowed. Standard support vector machines (SVM) training has $O(m3)$ time and $O(m2)$ space complexities, where m is the training set size. It is thus computationally infeasible on very large data sets. Our proposed OCVM inherits the advantage of the Core Vector Machine (CVM) algorithm which can be used with non-linear kernels and has a time complexity that is *linear* in m and a space complexity that is *independent* of m.

HCVM solves the skewed-class distribution problem for hierarchical stream data by identifying them through the sub-classes clustering process, creating child CVMs based on the hierarchical labels and applies supervised learning to update the core vectors. This puts strong emphasis on the unique problem subspaces and allows easy to discriminate parent classes by local modelling on their child classes.

DE-CVM takes HCVM a step further by implementing an evolving clustering process. DE-CVM evolves through incremental, hybrid learning and accommodates new input stream data, including new features, new classes, etc. through local element tuning. New core vectors are created and updated while the system is operating. In contrast to HCVM, DE-CVM can work not only on hierarchical data but also on any numerical stream data.

Meta Learning String Kernel CVM is proposed to satisfy the string format stream data learning. Recently, string kernel based support vector machines have shown competitive performance in tasks such as text classification and protein homology detection. Meta Learning String Kernel CVM improves the effectiveness of traditional string kernels SVMs by learning the meta knowledge and adopting CVMs.

The novel stream learning methods outlined above have been applied to the following three real world data modelling problems:

1. Hierarchical network data intrusion detection;
2. Face Membership Authentication;
3. String data (i.e. Spam email, news and malicious software) classification.

These solutions constitute the main contribution of this research to the area of applied information science.

In addition to the above contributions, two stream data visualisation systems were developed: the network intrusion detection visualisation system (NIDVS) and the HCVM prototype system. These systems overcome the difficulty of monitoring stream data learning progress and also provide a better understanding of local modelling.

In summary, real world problems consist of many smaller problems. It was found beneficial to acknowledge the existence of these sub-problems and address them through the use of local models.

The core vectors extracted from the local models also brought about the availability of new knowledge for researchers and would allow more in-depth study of the sub-problems to be carried out in future research.

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Data mining is concerned with the process of computationally extracting hidden knowledge structures represented in models and patterns from large data repositories. It is an interdisciplinary field of study that has its roots in databases, statistics, machine learning, and data visualisation. Data mining has emerged as a direct outcome of the data explosion that resulted from the success in database and data warehousing technologies over the past two decades (Frawley, Piatetsky-Shapiro, & Matheus, 1992; Fayyad, 1998).

The conventional focus of data mining research was on mining resident data stored in large data repositories. The growth of technologies such as wireless sensor networks (Akyildiz, Su, Sankarasubramaniam, & Cayirci, 2002) have contributed to the emergence of data streams (Muthukrishnan, 2003). The distinctive characteristic of such data is that it is unbounded in terms of continuity of data generation. This form of data has been termed as data streams to express its flowing nature (Henzinger & Raghavan, 1998).

Table 1.1 shows the major differences between data stream processing and traditional data processing. The objective of this table is to clearly differentiate between traditional stored data processing and stream processing as a step towards focusing on the data mining aspects of data stream processing systems.

Stream Processing	**Traditional Processing**
Real-time processing	Offline processing
Rapid data generation relative to the available computational resources	Normal or slow data generation relative to the available computational resources
Storage of data is not feasible	Storage of data is feasible
Approximate results are acceptable	Accurate results are required
Processing of samples of data is the usual task	Processing of every data item/record is the usual task
Linear and sublinear computational techniques are widely used	Techniques with high space and time complexity are used if necessary

Table 1.1:
Stream processing vs Traditional processing

1.2 RESEARCH GOAL AND OBJECTIVES

The goal of this research is to develop novel information methods and systems for stream data modelling and specifically for network intrusion detection and string classification applications. In order to achieve the goal, this study will investigate this new and promising area, and build an on-line modelling environment using evolving systems for stream data classification.

1.2.1 *Specific Research objectives*

More specifically, the research includes the following objectives:

1. To critically analyse the problems related to stream data mining. Although plenty of computational intelligent models have been developed so far for stream data learning, there are few effective and integrated systems. There are a variety of issues that have not been resolved yet.

2. To develop a new on-line kernel based learning method with low computational cost and to analyse its performance under different scenarios.

3. To develop evolving clustering methods based on a local modelling framework. Local modelling breaks down the entire problem into many smaller

sub-problems, based on its position in the problem space. Thus, evolving clustering is a fundamental step to the creation of a local modelling system.

4. To develop a novel string classification method for string format stream data mining. One major task for stream data mining is to categorise texture. This study aims to develop a kernel based classification method for texture categorisation and to investigate its performance over network string data.

5. To develop dynamic visualisation systems for monitoring stream data learning. These will present the stream data mining progress and local modelling evaluation.

In summary, the ultimate objective of this research is to develop new methods and systems for evolving stream data mining that leads to improved classification performance and various capabilities. Such methods and systems will integrate novel machine learning and modelling techniques for:

- on-line adaption;
- fast learning;
- evolving clustering;
- classification;
- knowledge discovery and model validation.

1.3 THESIS STRUCTURE

The structure of the thesis follows the research objectives presented in the previous section and is outlined below.

CHAPTER 2 reviews current developments in the area of stream data mining by providing background information, and describing existing modelling techniques, along with learning algorithms and applications.

CHAPTER 3 presents an overview of a range of computational intelligent techniques that are relevant to this research. A brief description of batch and on-line learning techniques that have been used for global, local and personalised modelling is provided.

CHAPTER 4 proposes an on-line, novel kernel based learning algorithm on-line core vector machine (OCVM), for which computational complexity is independent from the size of training samples. An experimental comparison between the proposed and traditional support vector machine is undertaken.

CHAPTER 5 proposes a novel learning method hierarchical core vector machine (HCVM) for local modelling of hierarchical structure stream data.

CHAPTER 6 proposes a novel kernel based method dynamic evolving core vector machine (DE-CVM), which improves the original HCVM algorithm by adopting evolving clustering processing that evolves through local element tuning.

CHAPTER 7 proposes a novel string classification method, Meta Learning String Kernels CVM, that is capable of string format stream data learning.

CHAPTER 8 demonstrates the inherent suitability of using HCVM on hierarchical structure stream data (i.e. network data flow). The experimental results validate the effectiveness of HCVM by comparing with the winner of KDD'99 cup (KDD99, 1999).

CHAPTER 9 presents a case study on face membership authentication with DE-CVM. The results attest that local modelling is able to capture more valuable knowledge from sub-spaces than global modelling does.

CHAPTER 10 presents experiments with Meta Learning String Kernel CVM for string classification. The results demonstrate the capability of Meta Learning String Kernel CVM for string format stream data learning.

CHAPTER 11 conclusions are drawn and future directions for research are given.

APPENDIX A AND B describes two dynamic visualisation systems developed for monitoring stream data learning progress.

1.4 CONTRIBUTIONS

During my PhD study, I have published or delivered 3 conference paper, 3 technical reports, 1 journal article, 4 novel methods and 2 prototype systems.

1.4.1 *Publications*

The material presented in this thesis was partially published in a number of peer-reviewed international conference and journal articles:

- Pang, S., Dhoble, K., Chen, Y., Kasabov, N., Ban, T. & Kadobayashi, Y. (2009) Active Mode Incremental Nonparametric Discriminant Analysis Learning. Proceedings of the Eighth International Conference on Information and Management Sciences. 407-412 July 2009 Kunming, China.

- Chen, Y., Pang, S., Kasabov, N., Ban, T. & Kadobayashi, Y. (2009) Hierarchical Core Vector Machines for Network Intrusion Detection. 16th International Conference on Neural Information Processing of the Asia & Pacific Neural Network Assembly, APNNA" (ICONIP09) 520-529 December Bangkok, Thailand.

- Chen, Y., Pang, S., Kasabov, N. & Ban, T. (2010) Factorizing class characteristics via group MEBs construction. Proceedings of the 17th international conference on Neural information processing: models and applications, 283-290, Sydney, Australia.

- Chen, Y., Kasabov, N. & Pang, S. (2012) Class Factorizing through Distinctive Core Vectors Extraction using Dynamic Evolving Minimum Enclosing Ball. *Evolving System* (submitted).

1.4.2 *Technical Reports*

3 technical reports have been completed and delivered to National Institution of Information and Communication Technology (NICT), Japan:

- Pang, S. Chen, Y. Kasabov, N. & Dhoble , K. (2009) FY-2008 High Speed Algorithms for Outlier Detection and Classification over Huge-size Network Data Streams. Auckland, New Zealand: Auckland University of Technology, Knowledge Engineering and Discovery Research Institute (KEDRI).

- Pang, S. Chen, Y. Kasabov, N. & Song, L. (2010) FY-2009 High Speed Algorithms for Outlier Detection and Classification over Huge-size Network Data Streams. Auckland, New Zealand: Auckland University of Technology, Knowledge Engineering and Discovery Research Institute (KEDRI).

- Pang, S. Chen, Y. Kasabov, N. & Song, L. (2011) FY-2010 High Speed Algorithms for Outlier Detection and Classification over Huge-size Network Data Streams. Auckland, New Zealand: Auckland University of Technology, Knowledge Engineering and Discovery Research Institute (KEDRI).

1.4.3 *Novel Methods*

Total 4 novel methods are presented in this thesis:

- On-line Core Vector Machine (OCVM) (Chapter 4)
- Hierarchical Core Vector Machine (HCVM) (Chapter 5)
- Dynamic Evolving Core Vector Machine (DE-CVM) (Chapter 6)
- Meta Learning for String Categorization (Chapter 7)

1.4.4 *Prototype Systems*

We developed 2 prototype systems:

- Network Intrusion Detection Visualization System (Appendix A)
- HCVM Prototype System (Appendix B)

Chapter 2

MINING DATA STREAM

This chapter reviews the theoretical foundations of data stream analysis. Techniques for mining stream data are critically reviewed. Finally, research problems in the streaming mining field of study are outlined and discussed. These research issues should be addressed in order to realize robust systems that are capable of fulfilling the needs of data stream mining applications.

Intelligent data analysis has passed through a number of stages. Each stage addresses new research issues that have arisen. Statistical exploratory data analysis represents the first stage. The goal was to explore the available data in order to test a specific hypothesis. With the advances in computing power, the field of machine learning has emerged. The objective was to find computationally efficient solutions to data analysis problems. Along with the progress in machine learning research, new data analysis problems have been addressed. Due to the increase in database sizes, new algorithms have been proposed to deal with the scalability issue. Moreover, machine learning and statistical analysis techniques have been adopted and modified in order to address the problem of very large databases. Data mining is an interdisciplinary field of study and is used to extract models and patterns from large amounts of information stored in data repositories (Hand, 1999; Hand, Mannila, & Smyth, 2001; Himberg, Tikanmaki, Toivonen, Korpiaho, & Mannila, 2001).

Advances in networking and parallel computation have lead to the introduction of distributed and parallel data mining. The goal is to extract knowledge from different subsets of a dataset and integrate these generated knowledge structures in order to build a global model of the whole dataset. Client/server, mobile agent based and hybrid models have been proposed to address the communication overhead issue. Different variations of algorithms have been developed in order to increase the accuracy of the generated global model. More details about distributed data mining could be found in (Park & Kargupta, 2002). Recently, the data generation rates in some data sources has become faster than ever before. This rapid generation of continuous streams of information has challenged the storage, computation and communication capabilities of computing systems. Systems, models and techniques have been proposed and developed over the past few years to address these challenges (Babcock, Babu, Datar, Motwani, & Widom, 2002).

The chapter is organised as follows. Section 2.1 presents the theoretical background of data stream analysis; mining data stream techniques and systems are reviewed in sections 2.2 and 2.3 respectively; finally section 2.4 summarizes this review chapter.

2.1 EXISTING TECHNIQUES

Research problems and challenges that have arisen in the area of mining data streams have been addressed by using well established statistical and computational approaches. We can categorise these solutions into data-based and task-based ones. In data-based solutions, the idea is to examine only a subset of the whole dataset or to transform the data either vertically or horizontally to an approximate, smaller size data representation. On the other hand, in task-based solutions, techniques from computational theory have been adopted to achieve time and space efficient solutions. In this section we review these theoretical foundations.

2.1.1 *Data-based Techniques*

Data-based techniques refer to summarising the whole dataset or choosing a subset of the incoming stream to be analysed. Sampling, load shedding and sketching techniques represent the former one, while synopsis data structures and aggregation represent the latter one. Here is an outline of the basics of these techniques with pointers to their applications in the context of data stream analysis.

Sampling

Sampling refers to the process of probabilistic choice of a data item to be processed or not. Sampling is an old statistical technique that has been used for a long time. Boundaries of the error rate of the computation are given as a function of the sampling rate. As a fast Machine Learning technique, Hoeffding bound has been used to measure the sample size according to some derived loss functions (Domingos & Hulten, 2001).

The problem with using sampling in the context of data stream analysis is the unknown dataset size. Thus the treatment of data streams should follow a special type of analysis to find the error bounds. Another problem with sampling is that it would be important to check for anomalies for surveillance analysis as an application in mining data streams. Sampling may not be the right choice for such an application. Sampling also does not address the problem of fluctuating data rates. It would be worth investigating the relationship among the three parameters: data rate, sampling rate and error bounds.

Load Shedding

Load shedding refers to the process of dropping a sequence of data streams (Mayur, Babcock, Datar, & Motwani, 2003). Load shedding has been used successfully in querying data streams. It has the same problems as sampling. Load shedding is difficult to be used with mining algorithms because it drops chunks of data streams that could be used in the structuring of the generated models or it might represent a pattern of interest in time series analysis.

Sketching

Babcock et al. (2002) describe the process of randomly projecting a subset of features. It is the process of vertically sampling the incoming stream. Sketching has been applied in comparing different data streams and in aggregate queries. The major drawback of sketching is that it has low accuracy. It is hard to use it in the context of data stream mining. Principal Component Analysis (PCA) would be a better solution in streaming applications.

Synopsis Data Structures

Creating synopsis of data refers to the process of applying summarisation techniques that are capable of summarising the incoming stream for further analysis. Wavelet analysis, histograms, quantiles and frequency moments have been proposed as synopsis data structures. Since synopsis of data does not represent all the characteristics of the dataset, approximate answers are produced when using such data structures (Gilbert, Kotidis, Muthukrishnan, & Strauss, 2003).

Aggregation

Aggregation is the process of computing statistical measures such as means and variance that summarise the incoming stream. This aggregated data could be used by the mining algorithm. The problem with aggregation is that it does not perform well with highly fluctuating data distributions. Merging online aggregation with offline mining has been studied by Aggarwal, Han, Wang, and Yu (2003, 2004).

2.1.2 *Task-based Techniques*

Task-based techniques are methods that modify existing techniques or invent new ones in order to address the computational challenges of data stream processing. Approximation algorithms, sliding window and algorithm output granularity represent this category. The following subsections, examine each of these techniques and its application in the context of data stream analysis.

Approximation algorithms

Approximation algorithms have their roots in algorithm design (Muthukrishnan, 2003). It is concerned with design algorithms for computationally hard problems. These algorithms can result in an approximate solution with error bounds. The idea is that mining algorithms are considered hard computational problems given their features of continuality and speed and the generating environment that is featured by being resource constrained. Approximation algorithms have attracted researchers as a direct solution to data stream mining problems. However, the problem of data rates with regard with the available resources could not be solved using approximation algorithms. Other tools should be used along with these algorithms in order to adapt to the available resources. Approximation algorithms have been used by Cormode and Muthukrishnan (2003).

Sliding Window

The inspiration behind the technique of sliding window is that the user is more concerned with the analysis of most recent data streams. Thus a detailed analysis is done over the most recent data items and summarised versions of the old ones. This idea has been adopted in many techniques in the comprehensive data stream mining system MAIDS project (Dong et al., 2003).

Algorithm Output Granularity

The algorithm output granularity technique (AOG) (Bandyopadhyay, Holder, & Maulik, 2006) introduces the first resource-aware data analysis approach that can cope with fluctuating very high data rates according to the available memory and the processing speed represented in time constraints. The AOG performs the local data analysis on a resource constrained device that generates or receives streams of information. AOG has three main stages. Mining followed by adaptation to resources and data stream rates represent the first two stages. Merging the generated knowledge structures when running out of memory represents the last stage. AOG has been used in clustering, classification and frequency counting (Bandyopadhyay et al., 2006). Having discussed the different existing approaches to data stream analysis problems, the following section is devoted to stream mining techniques that use the above approaches in different ways.

2.1.3 *Mining Techniques*

Mining data streams has attracted the attention of the data mining community for the last three years. A number of algorithms have been proposed for extracting knowledge from streaming information. This section reviews clustering, classification, frequency counting and time series analysis techniques.

Clustering

Guha, Mishra, Motwani, and O'Callaghan (2000) have studied analytically clustering data streams using the k-median technique. The proposed algorithm makes a single pass over the data stream and uses small space. It requires $O(nk)$ time and $O(n\epsilon)$ space where k is the number of centres, n is the number of points and $\epsilon < 1$. They have proved that any k-median algorithm that achieves a constant factor approximation can not achieve a better run time than $O(nk)$. The algorithm starts by clustering a calculated size sample into $2k$, and then, at a second level, the algorithm clusters the above points for a number of samples into $2k$. This process is repeated until it clusters the $2k$ clusters into k clusters.

Babcock, Datar, Motwani, and O'Callaghan (2003) have used an exponential histogram (EH) data structure to improve the algorithm proposed by Guha et al. (2000). They use the same method described above, however they address the problem of merging clusters when the two sets of cluster centres to be merged are far apart by maintaining the EH data structure. They have studied their proposed algorithm analytically.

Charikar, O'Callaghan, and Panigrahy (2003) have proposed another k-median algorithm that overcomes the problem of increasing approximation factors in the algorithm (Guha et al., 2000) with the increase in the number of levels used to result in the final solution of the divide and conquer algorithm. The algorithm has also been studied analytically.

Domingos and Hulten (2000) have proposed a general method for scaling up machine learning algorithms. They have termed this approach Very Fast Machine Learning VFML. This method depends on determining an upper bound for the learners loss as a function in a number of data items to be examined in each step of the algorithm. They have applied this method to K-means clustering VFKM and decision tree classification VFDT techniques. These algorithms have been implemented and evaluated using synthetic data sets as well as real web data streams. VFKM uses the Hoeffding bound to determine the number of examples needed in each step of k-means algorithm. The VFKM runs as a sequence of k-means executions where each run uses more examples than the previous one until a calculated statistical bound (Hoeffding bound) is satisfied.

Ordonez (2003) has proposed several improvements to k-means algorithm to cluster binary data streams. He has developed an incremental k-means algorithm. The experiments were conducted on real data sets as well as synthetic ones. He has demonstrated experimentally that the proposed algorithm outperforms the scalable k-means in the majority of cases. The proposed algorithm is a one pass algorithm in $O(Tkn)$ complexity, where T is the average transaction size, n is number of transactions and k is number of centres. The use of binary data simplifies the manipulation of categorical data and eliminates the need for data normalisation. The main idea behind the proposed algorithm is that it updates the cluster centres and weights after examining a batch of transactions which equalizes the square root of the number of transactions rather than updating them one by one.

O'Callaghan, Mishra, Meyerson, Guha, and Motwani (2002) have proposed stream and local search algorithms for high quality data stream clustering. The stream algorithm starts by determining the size of the sample and then applies the local search algorithm if the sample size is larger than a pre-specified equation result. This process is repeated for each data chunk. Finally, the local search algorithm is applied to the cluster centers generated in the previous iterations.

Aggarwal, Han, Wang, and Yu (2003) have proposed a framework for clustering data steams called CluStream algorithm. The proposed technique divides the clustering process into two components. The online component stores summarised statistics about the data streams and the offline one performs clustering on the summarised data according to a number of user preferences such as the time frame and the number of clusters. A number of experiments on real datasets have been conducted to prove the accuracy and efficiency of the proposed algorithm. Aggarwal et al. (2004) have recently proposed HP-Stream; a projected clustering for high dimensional data streams. HPStream has outperformed CluStream in recent results.

Keogh and Lin (2005) have shown empirically, that most highly cited clustering of time series data streams algorithms proposed so far in the literature come out with meaningless results in subsequence clustering. They have proposed a solution approach using k-motif to choose the sub-sequences that the algorithm can work on to produce meaningful results.

Bandyopadhyay, Holder, and Maulik (2006) have developed Lightweight Clustering LWC. It is an AOG-based algorithm. AOG has been discussed in section 2.1. The algorithm adjusts a threshold that represents the minimum distance measure between data items in different clusters. This adjustment is done regularly according to a pre-specified time frame. It is done according to the available resources by monitoring the input-output rate. This process is followed by merging clusters when the memory is full.

2.1.4 *Classification*

H. Wang, Fan, Yu, and Han (2003) have proposed a general framework for mining drifting data streams. They have observed that data stream mining algorithms proposed so far have not addressed the concept of drifting in the evolving data. The proposed technique uses weighted classifier ensembles to mine data streams. The expiration of old data in their model depends on the data distribution. They use synthetic and real life data streams to test their algorithm and compare between the single classifier and classifier ensembles. The proposed algorithm combines multiple classifiers weighted by their expected prediction accuracy. Also the selection of number of classifiers instead of using all is an option in the proposed framework without loosing accuracy in the classification process.

Ganti, Gehrke, and Ramakrishnan (2002) have developed analytically an algorithm for model maintenance using insertion and deletion of blocks of data records. This algorithm can be applied to any incremental data mining model. They have also described a generic framework for change detection between two data sets in terms of the data mining results they induce. They formalise the above two techniques into two general algorithms: GEMM and FOCUS. The algorithms have been applied to decision tree models and the frequent itemset model. The GEMM algorithm accepts a class of models and an incremental model maintenance algorithm for the unrestricted window option, and outputs a model maintenance algorithm for both window-independent and window-dependent block selection sequence. The FOCUS framework uses the difference between data mining models as the deviation in data sets.

Domingos and Hulten (2000) have developed VFDT. It is a decision tree learning systems based on Hoeffding trees. It splits the tree using the current best attribute taking into consideration that the number of examined data items used satisfies a statistical measure which is Hoeffding bound. The algorithm also deactivates the least promising leaves and drops the non-potential attributes.

Papadimitriou, Brockwell, and Faloutsos (2003) have proposed Arbitrary Window Stream mOdeling Method (AWSOM) for interesting pattern discovery from sensors. They developed a one-pass algorithm to incrementally update the patterns. Their method requires only $O(logN)$ memory where N is the length of the sequence. They conducted experiments with real and synthetic data sets. They use wavelet coefficients as compact information representation and correlation structure detection, and then apply a linear regression model in the wavelet domain.

Aggarwal et al. have adopted the idea of microclusters introduced in CluStream in On-Demand classification (Aggarwal et al., 2004) and it shows a high accuracy. The technique uses clustering results to classify data using statistics of class distribution in each cluster.

Last (2002) has proposed an online classification system that can adapt to concept drift. The system rebuilds the classification model with the most recent examples. Using the error rate as a guide to concept drift, the frequency of model building and the window size are adjusted. The system uses info-fuzzy techniques for model building and information theory to calculate the window size.

Ding, Ding, and Perrizo (2002) have developed a decision tree based on Peano count tree data structure. It has been shown experimentally that it is a fast building algorithm that is suitable for streaming applications.

Bandyopadhyay et al. (2006) have developed Lightweight Classification LW-Class. It is a variation of LWC. It is also an AOG-based technique. The idea is to use K nearest neighbours with updating the frequency of class occurrence given the data stream features. In case of contradiction between the incoming stream and the stored summary of the cases, the frequency is reduced. In the case when the frequency is equal to zero, all the cases represented by this class are released from the memory.

2.1.5 *Frequency Counting*

Giannella, Yang, Zhang, Yan, and Yu (2008) have developed a frequent item sets mining algorithm over data stream. They have proposed the use of tilted windows to calculate the frequent patterns for the most recent transactions based on the fact that users are more interested in the most recent transactions. They use an incremental algorithm to maintain the FP-stream which is a tree data structure to represent the frequent itemsets. They conducted a number of experiments to prove the algorithm efficiency.

Manku and Motwani (2002) have proposed and implemented an approximate frequency count in data streams. The implemented algorithm uses all the previous historical data to calculate the frequent patterns incrementally.

Cormode and Muthukrishnan (2003) have developed an algorithm for counting frequent items. The algorithm uses group testing to find the hottest k items. The algorithm is used with the turnstile data stream model which allows addition as well as deletion of data items. An approximation randomized algorithm has been used to approximately count the most frequent items. It is worth mentioning that this data stream model is the hardest to analyse. Time series and cash register models are computationally easier. The former does not allow increments and decrements and the latter one allows only increments.

Bandyopadhyay et al. (2006) have developed one more AOG-based algorithm: Lightweight frequency counting (LWF). It has the ability to find an approximate solution to the most frequent items in the incoming stream using adaptation and releasing the least frequent items regularly in order to count the more frequent ones.

2.1.6 *Time Series Analysis*

Indyk, Koudas, and Muthukrishnan (2000) have proposed approximate solutions with probabilistic error bounding to two problems in time series analysis: relaxed periods and average trends. The algorithms use dimensionality reduction sketching techniques. The process starts with computing the sketches over an arbitrarily chosen time window and creating the so called sketch pool. Using this pool of sketches, relaxed periods and average trends are computed. The algorithms have shown experimentally efficiency in running time and accuracy.

Perlman and Java (2002) have proposed a two phase approach to mine astronomical time series streams. The first phase clusters sliding window patterns of each time series. Using the created clusters, an association rule discovery technique is used to create affinity analysis results among the created clusters of time series.

Zhu and Shasha (2002) have proposed techniques to compute some statistical measures over time series data streams. The proposed techniques use the discrete Fourier transform. The system is called StatStream and is able to compute approximate error bounded correlations and inner products. The system works over an arbitrarily chosen sliding window.

Lin, Keogh, Lonardi, and Chiu (2003) have proposed the use of symbolic representation of time series data streams. This representation allows dimensionality/numerosity reduction. They have demonstrated the applicability of the proposed representation by applying it to clustering, classification, indexing and anomaly detection. The approach has two main stages. The first one is the transformation of time series data to Piecewise Aggregate Approximation followed by transforming the output to discrete string symbols in the second stage.

Y. Chen, Dong, Han, Wah, and Wang (2002) have proposed the application of so called regression cubes for data streams. Due to the success of OLAP technology in the application of static stored data, it has been proposed to use multidimensional regression analysis to create a compact cube that could be used for answering aggregate queries over the incoming streams. This research has been extended to be adopted in an undergoing project Mining Alarming Incidents in Data Streams (MAIDS).

Himberg et al. (2001) have presented and analysed randomised variations of segmenting time series data streams generated by on-board mobile phone sensors. It has been proven in this study that Global Iterative Replacement provides approximately an optimal solution with high efficiency in running time.

Guralnik and Srivastava (1999) have developed a generic event detection approach of time series streams. They have developed techniques for batch and online incremental processing of time series data. The techniques have proven efficiency with real and synthetic data sets.

2.2 SYSTEMS

Several applications have stimulated the development of robust streaming analysis systems. They are discussed below.

Burl, Fowlkes, Roden, Stechert, and Mukhtar (1999) have developed Diamond Eye for NASA and JPL. The aim of this project was to enable remote computing systems as well as observing scientists to extract patterns from spatial objects in real time image streams. The success of this project enables "a new era of exploration using highly autonomous spacecraft, rovers, and sensors?" (Burl et al., 1999). This project represents an early development in streaming analysis applications.

Kargupta et al. (2002) have developed the first ubiquitous data stream mining system: MobiMine. It is a client/server PDA-based distributed data stream mining application for stock market data. It should be pointed out that the mining component is located at the server side rather than the PDA. There are different interactions between the server and PDA untill the results are finally displayed on the PDA screen. The tendency to perform data mining at the server side has been changed with the increase of the computational power of small devices.

Tanner, Alshayeb, Criswell, Iyer, and Mcdowell (2008) have developed EnVironmen for On-Board Processing (EVE). The system mines data streams continuously generated from measurements of different on-board sensors in astronomical applications. Only interesting patterns are transferred to the ground stations for further analysis preserving the limited bandwidth. This system represents the typical case for astronomical applications. Huge amounts of data are generated and there is a need to analyse this streaming information in real time.

Srivastava (2003) have developed a NASA project for onboard detection of geophysical processes represented in snow, ice and clouds using kernel clustering methods. These techniques are used for data compression. The motivation of the project is to preserve the limited bandwidth needed to send image streams to the ground centers. The kernel methods have been chosen due to their low computational complexity in such resource-constrained environments.

2.3 RESEARCH ISSUES

Data stream mining is a stimulating field of study that has raised challenges and research issues to be addressed by the database and data mining communities. The following is a discussion of both addressed and open research issues (Dong et al., 2003; Bandyopadhyay et al., 2006; Golab & Özsu, 2003; Kargupta et al., 2002). The following is a brief discussion of previously addressed issues.

- Unbounded memory requirements due to the continuous flow of data streams: machine learning techniques represent the main source of data mining algorithms. Most of machine learning methods require data to be resident in memory while executing the analysis algorithm. Due to the huge amounts of data in the generated streams, it is absolutely crucial to deign space efficient techniques that can have only one-pass learning of the incoming stream. (Chapter 4)

- Required result accuracy: designing space and time efficient techniques should be accompanied with acceptable result accuracy. Approximation algorithms as mentioned earlier can guarantee error bounds. Also sampling techniques adopt the same concept as it has been used in VFML (Domingos & Hulten, 2001). (Chapter 5)

- Modelling changes of mining results over time: in some cases, the user is not interested in mining data stream results, but how these results change over time. For example, if the number of clusters generated changes, it might represent some changes in the dynamics of the arriving stream. Dynamics of data streams using changes in the knowledge structures generated would benefit many temporal-based analysis applications. (Chapter 6)

- Visualisation of data mining results on small screens of mobile devices: visualisation of traditional data mining results on a desktop is still a research issue. Visualisation in small screens of a PDA for example is a real challenge. Imagine a businessman and data is being streamed and analysed on his PDA. Such results should be efficiently visualised in a way that enables him to take a quick decision. This issue has been addressed in (Kargupta et al., 2002). (Appendix A and B)

The above issues represent the grand challenges to the data mining community in this essential field. There is a real need inspired by the potential applications in astronomy and scientific laboratories as well as business applications to address the above research problems.

2.4 SUMMARY

The dissemination of the data stream phenomenon has necessitated the development of stream mining algorithms. The area has attracted the attention of the data mining community. The proposed techniques have their roots in statistics and theoretical computer science. Data-based and task-based techniques are the two categories of data stream mining algorithms. Based on these two categories, a number of clustering, classification, frequency counting and time series analysis have been developed. Systems have been implemented to use these techniques in real applications. Mining data streams is still in its infancy. They are discussed in this thesis along with open issues in data stream mining.

Chapter 3

MODELING TECHNIQUES FOR COMPUTATIONAL INTELLIGENCE

This chapter gives an overview of different categories of computational intelligence models with their modelling methods in batch mode and online mode. Several research questions are formulated at the end of the chapter.

"Complex problems usually require a more complex intelligent system for their solution, consisting of several models. Some of these models can be evolving models." Kasabov (2007b)

Kasabov (2007c) put predictive models into three different categories.

1. A Global Model is a single model that learns from the entire dataset. The developed model is then applied on future data.

2. A Local Model is a fixed mixture of models trained on the entire dataset. However, when it is applied to future data, only one or a subset of relevant models will contribute to the prediction.

3. A Personalised Model is an individualised model that is created dynamically for each prediction, using only relevant input vectors through transductive reasoning.

A graphical representation of an integrated multimodel system is depicted in Figure 3.1 For every single input vector, the outputs of the tree models are weighted. The weights can be adjusted and optimised for every new input vector in a similar way to the parameters of a personalised model (Kasabov, 2007a).

$$y_i = w_{i,g} y_i(x_i)^{(global)} + w_{i,l} y_i(x_i)^{(local)} + w_{i,p} y_i(x_i)^{(personalised)} \tag{3.1}$$

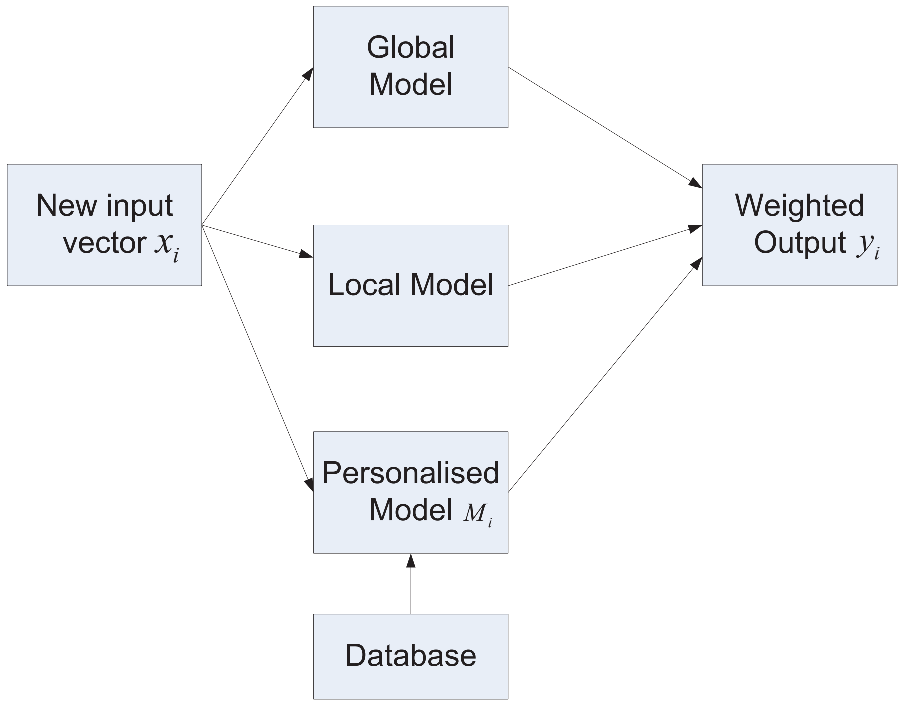


Figure 3.1: A graphical representation of an integrated global, local, and personalised multimodel system. For every single input vector, the outputs of the tree models are weighted (Kasabov, 2007a)

The chapter is organised as follows. Section 3.1 presents the global model. Local and personalised models are reviewed in sections 3.2 and 3.3 respectively. Section 3.4 describes the differences between batch mode and on-line mode learning. Sections 3.5 and section 3.6 reviews batch and on-line mode methods. Finally section 3.7 summarises this review chapter and gives open questions.

3.1 GLOBAL MODEL

The global model is a single, fixed, reusable model, trained with the entire dataset and can be applied to future data.

Most of today's predictive models are global (inductive) models, where the model learns from the training data and then is applied to future data. Linear Regression (Hastie, Tibshirani, & Friendman, 2001), Multi-layer Perceptron (MLP) ("Multilayer feedforward networks are universal approximators", 1989; S. Yang, Ho, & Lee, 2006), Support Vector Machine (SVM) (Vapnik, 1998), Adaptive-Network-Based Fuzzy Inference System (ANFIS) (Jang, 1993; Jang & Sun, 1995; Jang, Sun, & Mizutani, 1997) and Echo State Network (ESN) are examples of global models. MLP and SVM machine learning algorithms were proposed many years ago and are still the two most widely used neural network models.

There are two limitations with this type of model:

First, if a new pattern emerges in the future, the existing model will not be able to handle it as the model has not been trained to recognise this pattern and a new model may need to be developed. This can be time consuming depending on the model and the complexity of the problem.

Second, as the model is developed based on all available data with the objective of minimising overall prediction error, it will be biased toward the majority of the data. A pattern without enough support will have little influence on the model.

This is similar to the issue with interpolation versus extrapolation. If the new pattern is similar to some existing pattern, then it is considered interpolation, where there is enough support of the prediction made for this pattern. However, if the new pattern is very different from any of the existing patterns, then it is considered extrapolation, where the prediction made for this new pattern is less meaningful and subject to greater uncertainty.

Recent research in the field of machine learning has focused on model ensembles that use a mixture of models to achieve better overall accuracy. Several studies have reported that an ensemble of models works better than a global model (Cevikalp & Polikar, 2008; Islam, Yao, Shahriar Nirjon, Islam, & Murase, 2008; H. C. Kim, Pang, Je, Kim, & Bang, 2002; Nguyen, Abbass, & McKay, 2008; Pang & Kasabov, 2004; Yao & Liu, 1998, 1996; Zhou & Jiang, 2003).

There are many strategies that are commonly used to create an ensemble: bagging (H. C. Kim et al., 2002), boosting (Islam et al., 2008) and clustering (Kasabov & Song, 2002) are well known strategies. Depending on the strategy used, the ensembles generally try to either generate different views of one problem or break down the problem into smaller problems and tackle each problem independently. Sometimes both approaches are used.

3.2 LOCAL MODEL

The local model (Fontenla-Romero, Alonso-Betanzos, Castillo, Principe, & Guijarro-Berdias, 2002; Kasabov, 2001; Kasabov & Song, 2002; Lucks & Oki, 1999; Song & Kasabov, 2005; Yamada, Yamashita, Ishii, & Iwata, 2006) is a type of model ensemble that breaks down the problem into many smaller sub-problems, based on their position in the problem space. The sub-problems can be defined through a clustering process such as k-means, fuzzy c-means and hierarchical clustering that group similar input vectors based on their similarity (distance measure).

This type of model assumes that each cluster is a unique problem subspace and a sub-model should be developed for it. The quality of the cluster is, therefore, the foundation of this type of model.

The data clustering parameters often need to be adjusted, according to the sub-model's requirements or the characteristic of the problem. Many models, such as linear regression, need the number of input vectors to be significantly greater than the number of variables and, therefore, the clusters must be large enough to support this type of sub-model. Hence, local models may require more training data than the global model to ensure that each sub-model is trained with a sufficient amount of input vectors.

In addition, the clustering process is strongly affected by the amount of noise in the data of irrelevant or redundant features, as it affects the distance measure used by most clustering methods.

3.3 PERSONALISED MODEL

Transductive reasoning (Vapnik, 1998; Kasabov & Pang, 2003; Kasabov, 2007b) was originally proposed by Vapnik in 1998 for the development of an individualised model through transductive reasoning for a given input vector without first developing a generalised model in the intermediate stage.

This approach has been widely used to solve various real life problems like text classification (“Learning with progressive transductive support vector machine”, 2003; Joachims, 1999), speech recognition (Joachims, 2003), image recognition (Li & Chua, 2003) and language translation (Ueffing, 2007).

The main difference here is that transductive reasoning focuses on finding a solution for each prediction instead of creating a generalised solution for the problem and then uses it for each prediction.

The model is created dynamically for each prediction, which utilises all available data and uses the most suitable parameters, features or model to make the prediction.

3.4 BATCH VS ON-LINE MODE LEARNING

The on-line and batch modes are slightly different, although both will perform well for parabolic performance surfaces. One major difference is that the batch algorithm keeps the system weights constant while computing the error associated with each sample in the input. Since the on-line version is constantly updating its weights, its error calculation (and thus gradient estimation) uses different weights for each input sample. This means that the two algorithms visit different sets of points during adaptation. However, they both converge to the same minimum.

An online model can be: global (e.g. Incremental Support Vector Machine (SVM), Incremental Principal Component Analysis (PCA), Incremental Linear Discriminant Analysis (LDA)), local (e.g. Evolving Takagi-Sugeno (eTS+), Evolving Fuzzy Neural Networks (EFuNN), Dynamic Evolving Neural-Fuzzy Inference System (DENFIS)) or personalised (e.g. Weighted Nearest Neighbor (wkNN and wwKNN)).

A batch model can be global (e.g. SVM) or local but no adoption(e.g. SVM tree).

3.5 BATCH MODE METHODS

3.5.1 *Support Vector Machines*

Support vector machines (Vapnik, 1998) are a set of related supervised learning methods that are used for classification. In Support Vector machine, each instance in the training set contains one “target value” (class labels) and several “attributes” (features). The goal of SVM is to produce a model that predicts instances in the testing set given only their features.

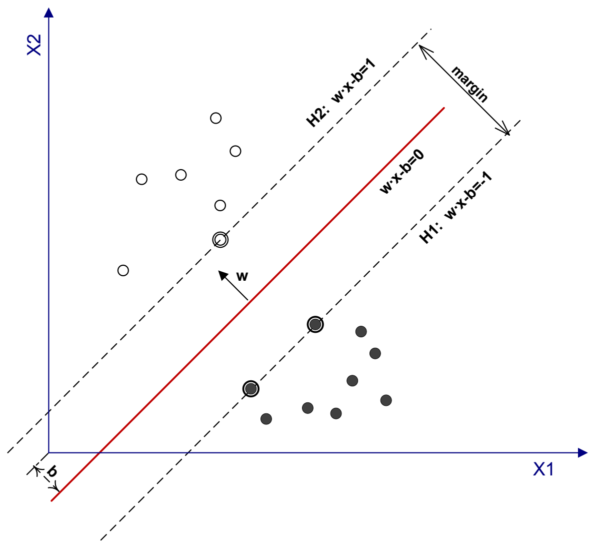


Figure 3.2: An example of the linear separating hyperplanes in SVM. Note: the support vectors are encircled

SVM works on the principle that it tries to form the hyperplane between the data points given it separates these data points into two sets in higher dimension space by mapping these data points into the higher dimension using the feature vectors that are obtained using the attributes of the data. One set contains positive class labels and the other set contains negative class labels.

The training data is given as $X = \{x_i, y_i\}, i = 1, \ldots, n, y_i \in \{-1, 1\}, x_i \in \mathbb{R}^m$, where x_i is an *m-dimensional* data vector, y_i is the corresponding class label.

In mapping the point from one dimension to a higher dimension in linear SVM, a dot product is used between them but there exist non-linear kernels that use other functions to achieve this task. This is needed because at times the distribution of the data is such that we cannot find a linear separating hyperplane even in higher dimensions. The kernel functions that we explored are as follows:

1. Polynomial : $(x.x^{'} + 1)^{d}$;
2. Radial : $exp(-\gamma||x - x^{'}||^{2})$;
3. Sigmoid Kernel : $tanh(\kappa x.x^{'} + c)$.

Assume there exist some hyperplanes that separate positive (label '+1') and negative (label '-1') samples. The data points x_i falling on such a hyperplane should satisfy the following equation:

$$w \cdot x_i + b = 0 \tag{3.2}$$

where w is a normal vector perpendicular to the hyperplane, a parameter b specifies the perpendicular offset from the hyperplane to the origin, and $\|w\|$ is an Euclidean normal vector of w.

The shortest distances from the separating hyperplane to the closest positive and negative data points are denoted by d_+ and d_-, respectively. Let d_+ and d_- be the "margin" of a separating hyperplane. Then, the given problem is simplified by using a SVM algorithm to find the separating hyperplane with the largest margin. If the training data are linearly separable, all the training data samples should satisfy the following constraints:

$$x_i \cdot w + b \geq +1, \forall y_i = +1 \tag{3.3}$$

$$x_i \cdot w + b \leq -1, \forall y_i = -1 \tag{3.4}$$

They can be further combined and written as:

$$y_i(x_i \cdot w + b) - 1 \geq 0, \forall i \in \{1, 2, \cdots, n\} \tag{3.5}$$

The data points satisfying the equality in Eq.3.3 will fall on the hyperplane $H1 : x_i \cdot w + b = +1$, with vector w and perpendicular distance from the origin

$|1-b|/\|w\|$. In the same way, the data points satisfying the equality in Eq.3.4 will fall on the hyperplane $H2 : x_i \cdot w + b = -1$, with vector w and perpendicular distance from the origin $|-1-b|/\|w\|$. The margin can be calculated by $2/\|w\|$, as $d_+ = d_- = 1/\|w\|$. Thus two parallel hyperplanes $H1$ and $H2$ are constructed, and there are no data points lying between them. Consequently, the pair of hyperplanes giving the maximum margin through minimising $\|w\|^2$ will be found and subjected to Eq.3.5. Finally, an optimal separation can be achieved by the hyperplane that has the greatest distance to the neighbouring data points of both classes, as is illustrated in Figure 3.2. The data points are referred as *support vectors* if they satisfy the equality in Eq.3.3 or 3.4 and their removal would change the solution to the discovered hyperplane. In Figure 3.2, support vectors are indicated by *extra circles*. Generally, the larger the margin, the lower the generalisation error of the classifier (Burges, 1998a).

For non-linear classification problems, a kernel function is introduced into SVM to find the maximum-margin hyperplane (Boser, Guyon, & Vapnik, 1992). The SVM based classifiers can be mathematically formulated by:

$$y(x) = \text{sign}\left[\sum_{i=1}^{n} a_i y_i \Phi(x, x_i) + b\right] \tag{3.6}$$

where a_i is a positive real constant and b is a real constant, Φ is a mapping function used for SVM kernel function construction (Suykens & Vandewalle, 1999),which typically has the choices from linear, polynomial and radial basis function (RBF) function. The solution to a nonlinear optimisation problem with inequality constraints is given by the saddle point of the Lagrangian, which is computed by:

$$\max_{\alpha_i, \upsilon_i} \min_{w,b,\xi_i} \mathfrak{L}(w, b, \xi_i; \alpha_i, \upsilon_i) \tag{3.7}$$

where $\mathfrak{L}$ is the Lagrangian constructed by:

$$\mathfrak{L}(w, b, \xi_i; a_i, \upsilon_i) = \mathfrak{J}(w, \xi_i) - \sum_{i=1}^{n} a_i\{y_i[w^T \varphi(x_i) + b] - 1 + \xi_i\} - \sum_{i=1}^{n} \upsilon_i \xi_i \tag{3.8}$$

where $a_i \geq 0$, $b_i \geq 0 (i = 1, \cdots, n)$ are Lagrange multipliers, $\mathfrak{J}$ is the risk bound minimised by:

$$\min_{w,\xi_i} \mathfrak{J}(w, \xi_i) = \frac{1}{2} w^T w + c \sum_{i=1}^{n} \xi_i \tag{3.9}$$

where the parameter ξ_i is introduced by:

$$y_i[w^T \varphi(x_i) + b] \leq 1 - \xi_i, \quad i = 1, \cdots, n \qquad \xi_i \leq 0 \tag{3.10}$$

Although SVM has been extensively used for solving real world problems in different research areas, there are some issues that we have to consider if we would like to have a successful implementation. One main limitation of SVM methods lies in the choice of kernel for solving real world problems, which remains an open research question in computer science and engineering. Another concern of SVM implementation for real world problems is speed and size, especially during training stage. This issue might make the learning process for a very large dataset (a large number of support vectors) particularly difficult (Burges, 1998a). Additionally, SVM is difficult to adapt to new data and the knowledge discovered by it is very limited (Kasabov, 2007c).

3.5.2 *SVM tree*

The SVM tree is constructed by a divide-and-conquer approach using a binary class-specific clustering and SVM classification technique; see, for example, Figure 3.3 (Pang & Kasabov, 2004; Pang, Havukkala, & Kasabov, 2006).

Basically, we perform two procedures at each node in the above tree generation. First, the class-specific clustering performs a rough classification because it splits the data into two disjoint subsets based on the global features. Next, the SVM classifier performs a 'fine' classification based on training supported by the previous separation result.

Figure 3.3 is an example of the SVM tree which is derived from the above SVM tree construction. As mentioned, the SVM test starts at the root node 1. If the test $T1(x) = +1$ is observed, the test $T2(x)$ is performed. If the condition

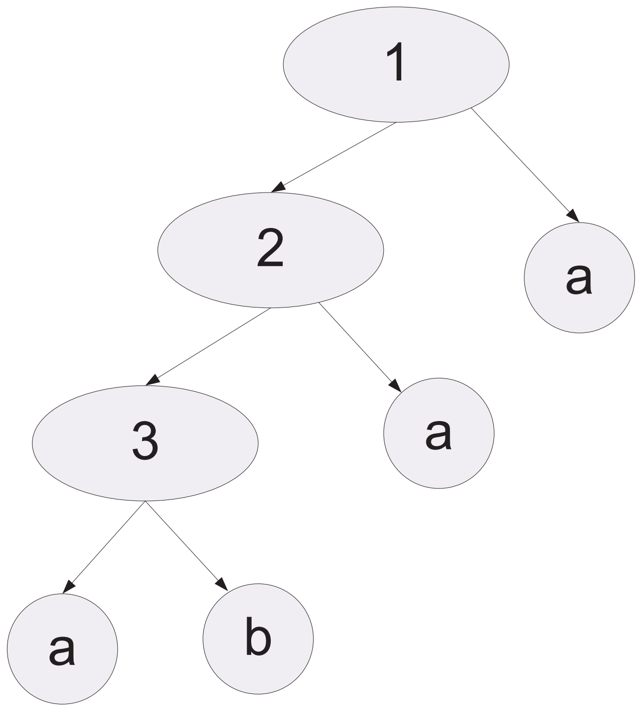


Figure 3.3: A SVM tree, where each node is a SVM (Pang & Kasabov, 2004)

$T1(x) = +1$ and $T2(x) = -1$ is observed, then the input data x are assigned to class a, and so forth.

SVM trees can evolve new nodes, new local SVM to accommodate new data from an input data stream. An example of an evolving SVM tree for CNS tumor is shown in Figure 3.4 (Pang et al., 2006). Most of class 1 patients were classified into one node of 26 patients. Class 2 patients were classified into two main nodes with 16 and 8 individuals, suggesting a potential difference in these two subsets of patients. The rest of individuals were classified into nodes of only a few patients each, so the question arises, whether these are especially hard to classify patients, misclassifications or maybe represent some other types of cancer. This exemplifies the potentially valuable additional information that the 2-SVMT can provide, compared to other algorithms.

Other cancer datasets produced similar trees, some simpler, some more complex. It appears some cancers are inherently more difficult to classify than others, and this is reflected in the complexity of the classification trees.

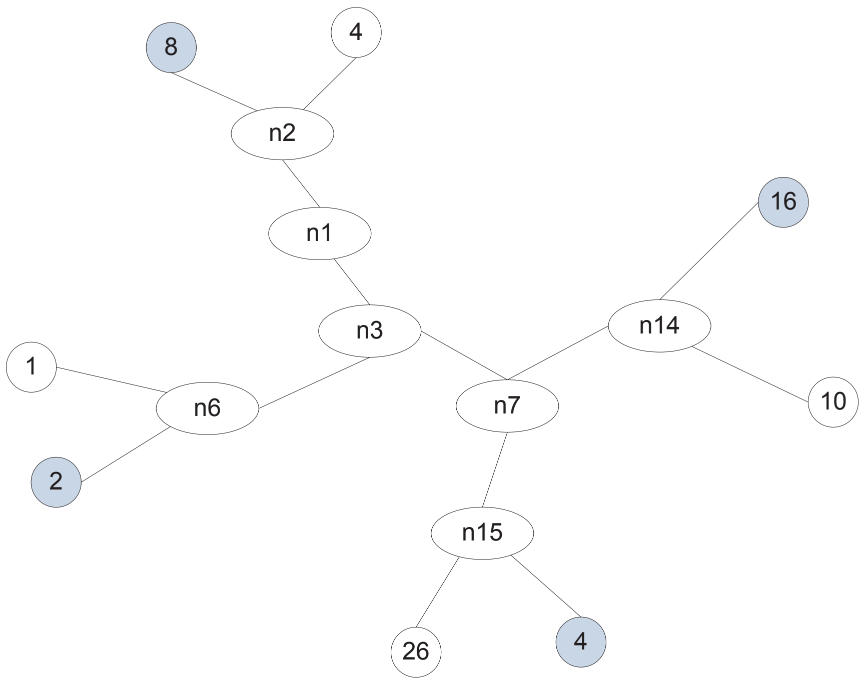


Figure 3.4: An example of two-class support vector machine decision tree (2-SVMT) for CNS tumor. Ellipses: Partitioning and SVM decision nodes, numbering indicates the sequence of decisions. Open circles: samples (patients) assigned to class 1, filled circles: class 2, corresponding to Table 1. Numbers inside circles indicates the number of patients in the end of each decision path. (Pang et al., 2006)

3.6 ON-LINE MODE METHODS

3.6.1 *Incremental Support Vector Machine*

Katagiri and Abe (2006a, 2006b) proposed an incremental training method which based on the assumption that candidates for support vectors exist near the separating hyperplane and are close to the surface of a region that includes training data of each class.

The incremental SVM method generates the minimum-volume hypersphere in the feature space that includes the training data of class $j (j = 1, 2)$ with radius R_j. Next, it defines a concentric hypersphere with radius ρR_j, where $\rho (0 < \rho < 1)$ is the user-defined parameter. Next, it defines the hypercone whose vertex is at the centre of the hyperspheres and which opens in the opposite direction of the separating hyperplane. The user-defined parameter $\theta(-90 < \theta < 90)$ defines the angle between the separating hyperplane and the surface of the hypercone.

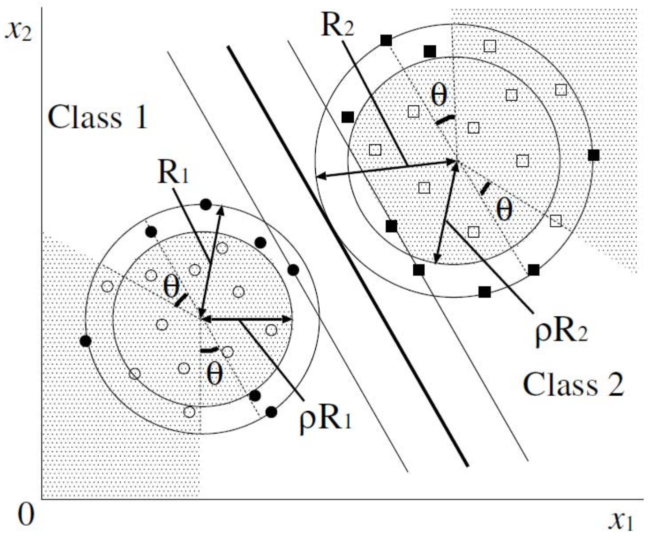


Figure 3.5: Deletion of the data using the hyperspheres (Katagiri & Abe, 2006a).

If the added data are in the shaded regions in Figure 3.5, they will be deleted. Figure 3.6 shows the progress of deleting such new data. If the distance $r_j(x)$ between $\phi(x)$ and the center of the hypersphere, a_j, is smaller than ρR_j, where $\phi(x)$ is the mapping function to the feature space:

$$r_j(x) < \rho R_j \tag{3.11}$$

the data is deleted. Otherwise, if the angle between $\phi(x) - a_j$ and the separating hyperplane, $\psi_j(x)$, is larger than θ:

$$\psi_j(x) > \theta \tag{3.12}$$

then $\phi(x)$ exists inside of the hypercone and x is deleted.

But even if equation 3.11 or 3.11 is satisfied, if x satisfies

$$y(x)D(x) \leq 1, \tag{3.13}$$

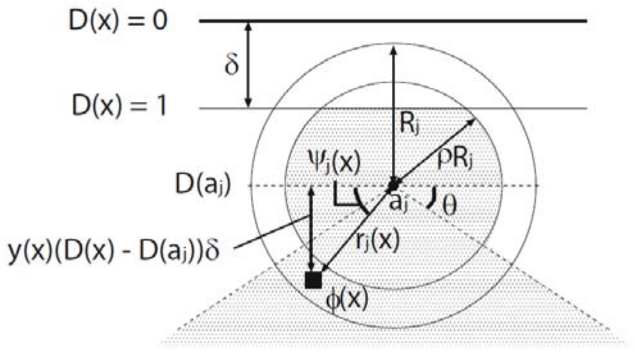


Figure 3.6: Judging whether the data are inside of the hypercone or not (Katagiri & Abe, 2006a).

x is a candidate for support vectors, where $D(x) = w^T\phi(x) + b$ is the decision function. In such a case, x will not be deleted. In addition, the data will be saved that are support vectors for hyperspheres because the support vectors for hyperspheres are candidates for the support vectors for hyperspheres at the next training step.

The general procedure for incremental training is as follows:

1. Train the support vector machine using the initial data set X_a.

2. Add the additional data set X_b to $X_a : X_a = X_a \cup X_b$.

3. If for $x \in X_a$, equation 3.13 is not satisfied and x satisfies $r_j(x) < \rho R_j$, where j is the class label for x or $\psi j(x) > \theta$, delete x from $Xa : Xa = Xa - \{x\}$.

4. If for $x \in X_a$ equation 3.13 is satisfied, retrain the support vector machine.

5. Repeat (2), (3), and (4).

3.6.2 *Incremental Principal Component Analysis*

Ozawa, Pang, and Kasabov (2010) proposed a new scheme of incremental learning in which feature extraction and classifier learning are simultaneously carried out online. Incremental Principal Component Analysis (IPCA) (Hall, Marshall, & Martin, 1998) is adopted as the feature extraction method, and k-nearest neighbour classifier with evolving clustering Method (ECM) (Kasabov & Song, 2002). The capability of this on-line learning system was verified as the classification accuracy was improved constantly even with a small set of training samples.

Assume that N training samples $x^i \in R^n (i = 1, \ldots, N)$ are initially provided to a system and eigenspace model $\Omega = (\bar{x}, U_k, \delta_k, N)$ is obtained by applying Principal Component Analysis (PCA) to the training samples. In the eigenspace model Ω, $\bar{x}$ is a mean vector of $x^i \in R^n (i = 1, \ldots, N)$, U_k is an $n \times k$ matrix whose column vectors correspond to eigenvectors, and $\delta_k = diag\{\lambda_1, \ldots, \lambda_k\}$ is a $k \times k$ matrix whose diagonal elements are non-zero eigenvalues. Here, k is the number of eigen-axes spanning the eigenspace and the value of k is determined based on a certain criterion. After calculating Ω, the system holds the information on Ω and all the training samples are thrown away.

Now assume that the $(N+1)^{th}$ training sample $x^{N+1} = y \in R^n$ is given. The addition of this new sample results in changes in the mean vector and the covariance matrix; therefore, the eigenspace model $\Omega = (\bar{x}, U_k, \delta_k, N)$ should be updated. Let us define the new eigenspace model by $\Omega' = (\bar{x}', U'_{k'}, \delta'_{k'}, N+1)$. Note that the eigenspace dimensions might be increased from k to $k+1$; thus, k' in Ω' is either k or $k+1$. Intuitively, if y includes almost all energy in the current eigenspace spanned by the eigenvector U'_k, there is no need to increase its dimensions. However, if y includes a certain energy in the complementary eigenspace, the dimensional augmentation is inevitable; otherwise, crucial information on the new sample y might be lost. Regardless of the necessity in eigenspace augmentation, the eigen axes should be rotated to adapt to variation in the data distribution.

To build a classifier under a dynamic enviroment, Ozawa et al. (2010) propose a nearnest-neighbor classifier whose prototypes are evolved by the ECM (Kasabov & Song, 2002)

3.6.3 *Incremental Linear Discriminant Analysis*

Linear Discriminant Analysis (LDA) finds the linear projections of data that best separate two or more classes under the assumption that the classes have equal covariance Gaussian structure (Fukunaga, 1990). LDA is an effective and widely employed technique for dimension reduction and feature extraction. It is often beneficial to learn the LDA basis from large training sets, which may not be available initially. This motivates techniques for incrementally updating the discriminant components when more data becomes available.

As noted by Fukunaga (1990), there are equivalent variants of Fisher's criterion to find the projection matrix U to maximise class separability of the data set:

$$max_{\arg U}\frac{U^T S_B U}{U^T S_W U} = max_{\arg U}\frac{U^T S_T U}{U^T S_W U} = max_{\arg U}\frac{U^T S_B U}{U^T S_T U}, \qquad (3.14)$$

where

$$S_B = \sum_{i=1}^{C} n_i (m_i - \mu)(m_i - \mu)^T \qquad (3.15)$$

is the between-class scatter matrix,

$$S_W = \sum_{i=1}^{C} \sum_{x \in C_i} (x - m_i)(x - m_i)^T \qquad (3.16)$$

is the within-class scatter matrix,

$$S_T = \sum_{\forall x} (x - \mu)(x - \mu)^T = S_B + S_W \qquad (3.17)$$

is the total scatter matrix; C the total number of classes; n_i the sample number of class i; m_i is the mean of class i, and μ is the global mean.

T. K. Kim, Wong, Stenger, Kittler, and Cipolla (2007) use the third criterion in equation 3.14 and separately update the principal components as the minimal sufficient spanning sets of S_B and S_T . The scatter matrix approximation with a small number of principal components (corresponding to significant eigenvalues) allows an efficient update of the discriminant components. The S_T matrix rather than S_W is used to avoid losing discriminatory data during the update. If we only kept track of the significant principal components of S_B and S_W, any discriminatory information contained in the null space of SW would be lost (note that any component in the null space maximises the LDA criterion). However, as $S_T = S_B + S_W$ and both S_B and S_W are positive semi-definite, vectors in the null space of S_T are also in the null space of S_B, and are thus being ignored in the update. The two steps of the algorithm are: (1) Update the total scatter matrix S_T , (2) Update the between-class scatter matrix S_B.

Given two sets of data represented by eigenspace models

$$\{\mu_i, M_i, P_i, \Lambda_i\}_{i=1,2} \tag{3.18}$$

where μ_i is the mean, M_i the number of samples, P_i the matrix of eigenvectors and Λ_i the eigenvalue matrix of the i-th data set, the combined eigenspace model $\{\mu_3, M_3, P_3, \Lambda_3\}$ is computed.

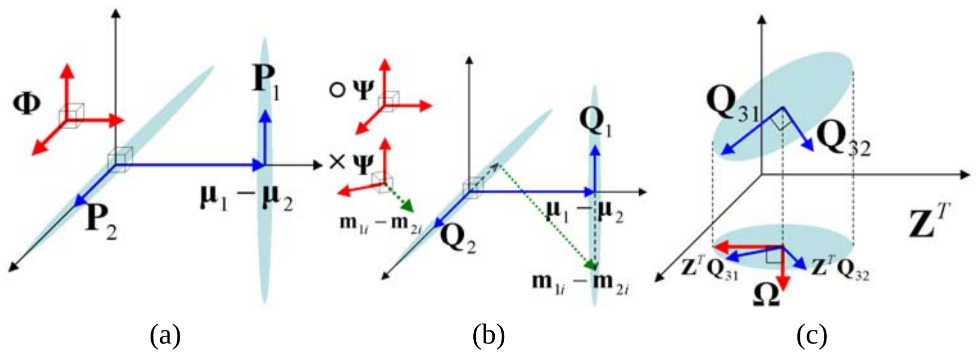


Figure 3.7: Concept of sufficient spanning sets of the total scatter matrix (a), the between-class scatter matrix (b) and the projected matrix (c). The union set of the principal components P_1, P_2 or Q_1, Q_2 of the two data sets and the mean difference vector $\mu_1 - \mu_2$ can span the respective total or between-class scatter data space (a and b). The dimension for the component $m_{1i} - m_{2i}$ should not be removed (cross=incorrect) from the sufficient set of the between-class scatter data but retained in the set (circle=correct) (b). The projection and orthogonalisation of the original components Q_{31}, Q_{32} yields the principal components of the projected data up to rotation (c). See the corresponding sections for detailed explanations

As visualised in Figure 3.7a, the union of the two principal components and the mean difference vector can span all data points of the combined set in the three dimensional space. The principal components of the combined set are found by rotating this sufficient spanning set. Note that this use of the sufficient spanning set is only possible in the case of merging generative models where the scatter matrix of the union set is represented as the sum of the scatter matrices of the two sets explicitly as

$$S_{T,3} = S_{T,1} + S_{T,2} + M_1 M_2 / M_3 \cdot (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T, \tag{3.19}$$

where $\{S_{T,i}\}_i = 1, 2$ are the scatter matrices of the first two sets. The method can therefore not be used to directly merge the discriminant components of LDA models.

The between-class scatter matrix $S_{B,i}$ can be written as

$$\begin{array}{ll} S_{B,i} & = \sum_{j=1}^{C_i} n_{ij}(m_{ij} - \mu_i)(m_{ij} - \mu_i)^T \\ & = \sum_{j=1}^{C_i} n_{ij} m_{ij} m_{ij}^T - M_i \mu_i \mu_i^T. \end{array}, \tag{3.20}$$

The combined between-class scatter matrix can further be written as the original between-class scatter matrices and an auxiliary matrix A as

$$S_{B,3} = S_{B,1} + S_{B,2} + A + M_1 M_2 / M_3 \cdot (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T, \tag{3.21}$$

where

$$A = \sum_{k \in s} \frac{-n_{1k} n_{2k}}{n_{1k} + n_{2k}} (m_{2k} - m_{1k})(m_{2k} - m_{1k})^T. \tag{3.22}$$

The set $s = \{k | k = 1, 2, \ldots, c\}$ contains the indices of the common classes of both data sets.

The incremental LDA allows highly efficient learning to adapt to new data sets. A solution closely agreeing with the batch LDA result can be obtained with far lower complexity in both time and space. The incremental LDA algorithm can also be incorporated into a classic semi-supervised learning framework and applied to many other problems in which LDA-like discriminant components are required.

3.6.4 *Evolving Takagi-Sugeno model*

Angelov and Buswell (2003) proposed the evolving Takagi-Sugeno (eTS) fuzzy system; after that, Angelov (2010) presented an enhanced version of eTS algorithm which is called eTS+. This algorithm has been tested on time-series prediction and a data stream from a real engine.

Evolving Takagi-Sugeno (TS) fuzzy systems (Angelov, 2004) can be of multi-input-multi-output (MIMO) type. As seen from Figure 3.8, eTS+ is represented as a six-layer neuro-fuzzy system. The first aim of data space partitioning can be achieved by forming clusters around focal points that have high density.

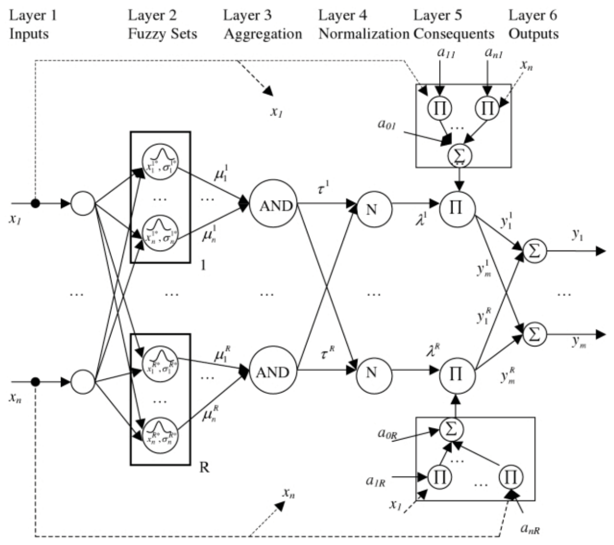


Figure 3.8: eTS+ as a neural network (Angelov, 2010)

The aim of the eTS+ clustering method differs from typical clustering approaches as the clusters associated with antecedent parts of fuzzy systems significant overloap. In this case, (Angelov, 2010) adopted centre of gravity (CoG) aggregation (Yager & Filev, 1994) which weighs the average outputs of individual fuzzy rules to produce the overall output. By doing this, the fuzzy system is able to cope with uncertainties and has a cooperative nature.

Once clusters are generated, eTS+ can monitor the quality of the clusters online. Five quality elements are estimated in total: support, age, utility, zone of influence and local density.

Once the structure of eTS+ is defined and established, (Angelov, 2004) proposed weighted least square method (fwRLS) for local or global optimisation. The test results show that local optimum, usually provides a lower error rate than globally optimum.

3.6.5 *Evolving Fuzzy Neural Networks (EFuNN)*

Evolving Fuzzy Neural Networks (EFuNN) (Kasabov, 2002) is a connectionist model with neuro-fuzzy inference systems for implementing ECOS. EFuNNs are fuzzy neural network structures that evolve based on Evolving Connectionist Systems (ECOS) principles. Fuzzy neural networks are connectionist structures that can be interpreted by a set of fuzzy rules and a fuzzy inference system (Jang, 1993). EFuNN has a five-layer structure in which all nodes represent *membership functions* (MF) and can be modified during learning. Figure 3.9 illustrates an example of an EFuNN with a short term memory and feedback connections.

The input layer is the first layer that contains input variables. The second layer is a fuzzy input layer where each input variable is represented by a group of neurons. These neurons denote the fuzzy quantisation of the input variable, e.g. three neurons can be used to represent “best”, “good” and “bad” fuzzy values of a variable. Different MFs can be attached to the neurons, such as triangular or Gaussian MF. This layer aims to assign the input variables into membership degrees to which they belong to the corresponding MF. Within this layer, new neurons are created, when the corresponding variable value of a given input vector does not belong to any of the existing MFs. An optional short-term memory layer can be introduced through feedback connections from the rule node layer.

The rule (case) layer is the third layer in EFuNN which contains rule nodes that evolve through supervised or unsupervised learning. The rule nodes represent prototypes of the associations between input and output data. Each rule

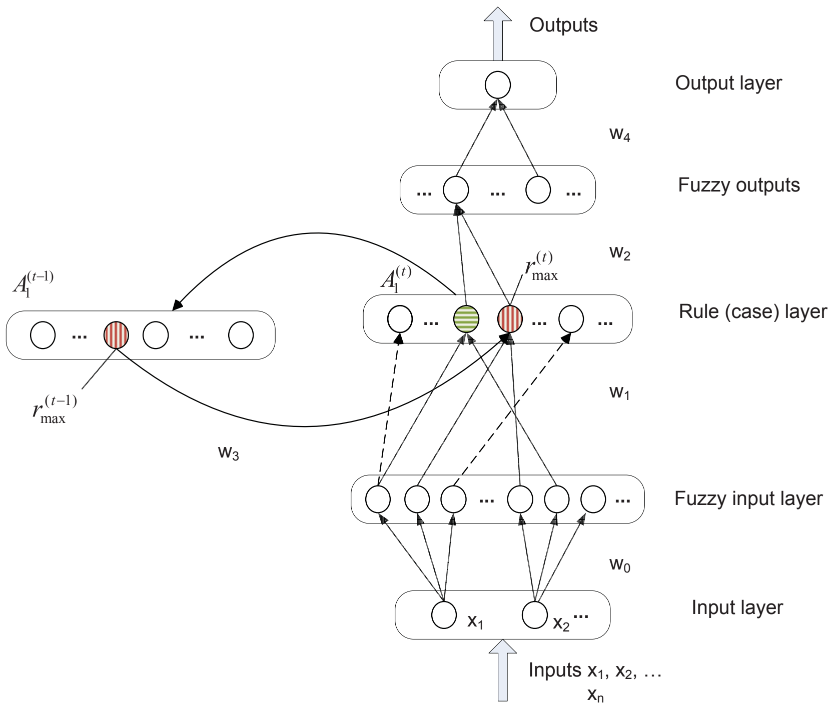


Figure 3.9: An example of an EFuNN with a short term memory and feedback connections, adapted from Kasabov (2001)

node r is defined by two vectors of connection weights: $\omega_1(r)$ and $\omega_2(r)$. The former is adjusted by an unsupervised learning model based on the similarity measurement within a local problem space, while the latter is adjusted by a supervised learning model based on the estimation of output error. The neurons in the fourth layer represent the fuzzy quantisation of the output variables. Finally, the fifth layer gives the value of the output variables.

3.6.6 *Dynamic Evolving Neural-Fuzzy Inference System*

A Dynamic Evolving Neural-Fuzzy Inference System (DENFIS) (Kasabov & Song, 2002) is a fuzzy inference system that is capable of on-line and offline learning through on-line clustering. This paper had over 380 citations on Google Scholar at the time of writing.

DENFIS starts by clustering the data and creates a fuzzy inference system that is based on the clusters. A maximum distance-based clustering algorithm, Evolving Clustering Method, is used to cluster the input data. Once the clusters are derived, a Takagi-Sugeno fuzzy rule is created for each cluster. These rules are then optimised through the back-propagation method. For each prediction, m most activated rules are dynamically chosen to derive the final output. New rule sets can be inserted into or extracted from the model. The DENFIS algorithm is described in Algorithm 1

```
Input: Input vectors $x_i \in X$
Output: $n$ clusters' centers $C_{c_j}$ and corresponding radius $R_{u_j}$ where
    $j = 1, 2, \ldots, n$
 1: Create the initial cluster $C_1$ and set the position of the first training data as
    a cluster center $C_{c_1}$ with the cluster radius $R_{u_1} 0$.
 2: while  do
 3:   if $\forall x_i \in C = \{C_1, C_2, \ldots, C_n\}$ then
 4:     Terminates the algorithm.
 5:   else
 6:     Calculate the distance between the current training input vector $x_i$
        and the cluster center $C_{c_j}$. $D_{ij} = \left\|x_i - C_{c_j}\right\|, j = 1, 2, \ldots, n.$
 7:   end if
 8:   if $\exists \|x_i - C_{c_m}\| \leq R_{u_m}$ then
 9:     $x_i$ is assumed to belong to cluster $C_m$ and no new cluster is created
        and no existing cluster is updated.
10:     Go back to step $3$.
11:   end if
12:   Find a cluster $C_a$ from all existing cluster centers with $S_{ij} = D_{ij} + R_{u_j}$,
      $j = 1, 2, \ldots, n$, and select the cluster center $C_{c_a}$ with the smallest
      $S_{ia} = D_{ia} + R_{u_a} = \min\{S_{ij}\}, j = 1, 2, \ldots, n$.
13:   if $S_{ia} > 2 \times Dthr$ then
14:     $x_i$ does not belong to any of the existing clusters and a new cluster is
        created as described in first Step.
15:     Go back to step $3$.
16:   end if
17:   if $S_{ia} \leq 2 \times Dthr$ then
18:     The cluster $C_a$ is updated by moving $C_{c_a}$ and enlarging the cluster
        radius $R_{u_a} = S_{ia}/2$ and the new cluster center $C_{c_a}$ is set as follows:
        $C_{c_a^{new}} = x_i - ((C_{c_a} - x_i) \times \frac{S_{ia}/2}{D_{ia}}$.
19:   end if
20: end while
```

Algorithm 1: Evolving Clustering Method (ECM): a fast one-pass algorithm

$Dthr$ is the distance threshold of the cluster, which defines the maximum radius of the cluster. S_{ia} defines inverse the level how much the xi belongs to cluster centers C_a. The smaller the S_{ia}, the more x_i belongs to cluster C_a. Note that x_i can belong to multiple clusters as there may be overlapping of the cluster radius. Euclidean distance is used as the distance measuring method in ECM.

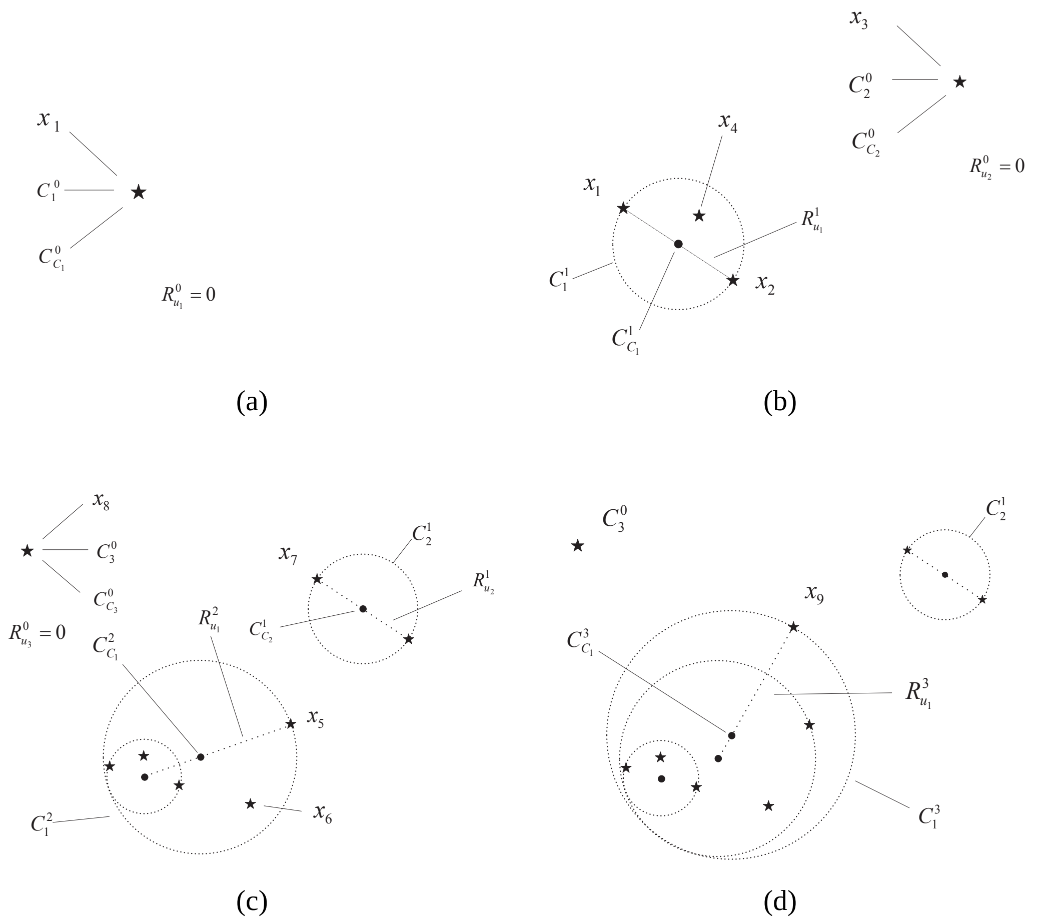


Figure 3.10: Example of ECM Clustering algorithm. x_i: input vector (*), $C_{c_j}^k$: cluster center, C_j^k: cluster, $R_{u_j}^k$ cluster radius (Kasabov & Song, 2002)

Figure 3.10 shows the ECM clustering process step by step.

(a) The initial cluster is created for the first input vector x_1.

(b) x_2: update cluster $C_1^0 \rightarrow C_1^1$

x_3: create a new cluster C_2^0

x_4: belongs to C_1^1, no action required.

(c) x_5: update cluster $C_1^1 \rightarrow C_1^2$

x_6: belongs C_2^1, no action required

x_7: update cluster $C_2^0 \rightarrow C_2^1$

x_8: create a new cluster C_3^0

(d) x_9: update cluster $C_1^2 \rightarrow C_1^3$

ECM processes input vectors in a one-input-vector-at-a-time manner, and therefore the order of the input vectors being processed affects the final output. This is evident in the way the first cluster is created for the first input vector. This design was necessary since ECM is an on-line clustering method where data is made available one input vector at a time. However, this does not prove to be a significant problem in practice as the cluster centre may be slightly different based on the order of input vectors being process, when inspected closely by visualising the input vectors in each cluster, the input vectors were very similar, as ECM originally intended.

The consequence of the Takagi-Sugeno fuzzy rule is created and updated by a (weighted) least-square estimator. The linear function is expressed as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q \tag{3.23}$$

The coefficient β is obtained through the following formula

$$\begin{aligned} \beta &= [b_1\ b_2\ \ldots\ b_q]^T \\ b &= (A^T A)^{-1} 1 A^T y \end{aligned}, \tag{3.24}$$

or for the weighted version of the LSE

$$b = (A^T W A)^{-1} A^T W y \tag{3.25}$$

where

$$A = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1q} \\ 1 & x_{21} & x_{22} & \cdots & x_{2q} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{p1} & x_{p2} & \cdots & x_{pq} \end{pmatrix}$$

and

$$y = [y_1 \; y_2 \; \ldots \; y_p]^T . \tag{3.26}$$

Assume W is a diagonal matrix:

$$W = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & w_p \end{pmatrix} \tag{3.27}$$

equation 3.24 and 3.25 can be rewritten as

$$Weighted - LSE = \begin{cases} P & = (A^T A)^{-1} \\ b & = PA^T y \end{cases} \tag{3.28}$$

$$LSE = \begin{cases} P_w & = (A^T W A)^{-1} \\ b_w & = P_w A^T W y \end{cases} \tag{3.29}$$

In the DENFIS on-line mode, the weighted recursive LSE is used with the following equation:

$$WRLSE = \begin{cases} b_{k+1} & = b_k + w_{k+1} P_{k+1} a_{k+1} (y_{k+1} - a_{k+1}^T b_k) \\ P_{k+1} & = \frac{1}{\lambda}(P_k - \frac{w_{k+1} P_k a_{k+1} a_{k+1}^T P_k}{\lambda + a_{k+1}^T P_k a_{k+1}}) \end{cases} \tag{3.30}$$

The forgetting factor λ is set between 0.8 and 1. The DENFIS on-line model learning procedure is explained below:

1. Perform ECM clustering on the initial set of data n0 to obtain M clusters
2. For every cluster C_i, find p_i data points are closest to $C_i, i = 1, 2, \ldots, M$;
3. Create a fuzzy rule for each cluster. The antecedent of the fuzzy rule is the cluster center. The consequent function is created using equation 3.28 or 3.29. The distance between p_i and the cluster center is used to create the weight matrix.
4. The size of p_i is a model training parameter. It defines the number of data points used to derive the consequent function of the fuzzy rules.

As new input vector enters the system, new fuzzy rules may be created and some rules updated. A new fuzzy rule is created if a new cluster is found in ECM. If no new clusters are created, one or more fuzzy rules are updated by using equation 3.30.

For each input vector, the DENFIS on-line model dynamically creates a Takagi-Sugeno fuzzy inference system using m activated rules. m is a model training parameter that should be adjusted based on the characteristic of the problem. The rules are chosen based on the position of the input vector. Since the rules are updated constantly, two input vectors with the same values at different time points may have different inferences as the fuzzy rule may have been updated before the second input vector entered the system.

3.6.7 *Weighted Nearest Neighbor: WKNN & WWKNN*

In a weighted distance KNN algorithm (WKNN) , the output y_i is calculated not only based on the output values (e.g. class label) y_j, but is also dependent on the weight w_j measured by the distance between the nearest neighbours and the new data sample x_i:

$$y_i = \frac{\sum\limits_{j=1}^{K_i} w_j \cdot y_j}{\sum\limits_{j=1}^{K_i} w_j} \tag{3.31}$$

where:

- y_i is the predicted output for the new vector x_i;
- y_j is the class label of each sample in the neighborhood of x_i.
- K_i is the number of K nearest samples to x_i;
- w_j is the is the weight value calculated based on the distance from the new input vector x_j to its K nearest neighbours.

The weight w_j can be calculated as follows:

$$w_j = \frac{\max(d) - (d_j - \min(d))}{\max(d)}, \quad j = 1, \cdots, K \tag{3.32}$$

where:

- the value of weights w_j ranges from $\frac{\min(d)}{\max(d)}$ to 1;
- $d = [d_1, d_2, \cdots, d_K]$ denotes the distance vector between the new input data d_i and its K nearest neighbouring samples;
- $\max(d)$ and $\min(d)$ are the maximum and minimum values for vector d.

The distance vector d is computed as:

$$d_j = \sqrt{\sum_{l=1}^{m}(x_{i,l} - x_{j,l})^2}, \quad j = 1, \cdots, K \tag{3.33}$$

where m is the number of variables (features) representing the new input vector x_i within the problem space; $x_{i,l}$ and $x_{j,l}$ are the l^{th} variable values corresponding to the data vector x_i and x_j, respectively.

The output from a WKNN classifier for the new input vector x_i is a "*personalised probability*" that indicates the probability of vector x_i belonging to a given class. For a two-class classification problem, a WKNN classifier requires a threshold θ to determine the class label of x_i, i.e., if the output (*personalised probability*) is less than the threshold θ, then x_i is classified into the group with "small" class label, otherwise into the group with "big" class label.

Weighted distance and weighted variables K-nearest neighbors (WWKNN) is a personalised modeling algorithm introduced by Kasabov (2007c). The main idea behind WWKNN algorithm is: the K nearest neighbor vectors are weighted based on their distance to the new data vector x_i, and also the contribution of each variable is weighted according to their importance within the local area where the new vector belongs (Kasabov, 2007c). In WWKNN, the assumption is made that the different variables have different importance to classifying samples into different classes when the variables are ranked in terms of their discriminative power of class samples over the whole m-dimensional space. Therefore, it will be more likely that the variables have different ranking scores if the discriminative power of the same variables is measured for a sub-space (localised space) of the entire problem space. The calculation of Euclidean distance d_j between a new vector x_i and a neighbor x_j is mathematically formulated by:

$$d_j = \sqrt{\sum_{l=1}^{K} c_{i,l}(x_{i,l} - x_{j,l})^2}, \quad j = 1, \cdots, K \tag{3.34}$$

where: $c_{i,l}$ is the coefficient weighting x_l in relation with its neighborhood of x_i, and K is the number of the nearest neighbors. The coefficient $c_{i,l}$ can be calculated by a SNR function that ranks variables across all vectors in the neighbourhood set $D_{nbr}(x_i)$:

$$\begin{aligned} c_{i,l} &= \{c_{i,1}, c_{i,2}, \ldots, c_{i,K}\} \\ c_{i,l} &= \frac{\left|\bar{x}_l^{class1} - \bar{x}_l^{class2}\right|}{\sigma_l^{class1} + \sigma_l^{class2}} \end{aligned}, \tag{3.35}$$

where $\bar{x}_l^{classi}, i = \{1, 2\}$ and $\sigma_l^{classi}, i = \{1, 2\}$ are the mean value and standard deviation of the l^{th} feature belonging to class i across the neighborhood $D_{nbr}(x_i)$ of x_j, respectively.

Comparing to a conventional KNN algorithm, the contribution of WWKNN lies in the new distance measurement: all variables are weighted according to their importance as discriminating factors in the neighborhood area (personalised sub-space), which might provide more precise information for classification or prediction of the new data vector.

3.7 SUMMARY AND OPEN QUESTIONS

This chapter reviews methods and techniques that are used or are highly related to the research in this PhD study. Each of the work carried out in this PhD study are either improvements made on previous studies reviewed in this chapter or integrate existing methods in a new way to achieve better results.

As a result of the reviews presented in chapter 2 and 3, several research questions related to stream data mining can be formulated as follows:

1. Can we (successfully or accurately) train/test large amount of data at real-time? (Chapter 4, page 51)

2. Can we extract more knowledge from hierarchically labelled data that is often seen in stream datasets? (Chapter 5, page 67)

3. Can the local knowledge discovered from stream data help us in achieving better results? (Chapter 6, page 80)

4. How can we produce more accurate on string classification? (Chapter 7, page 95)

5. How to monitor the process of dynamic stream data learning? (Appendix A and B, page 143-154)

These question will be answered in the rest of the thesis.

In the next chapter, a novel method is proposed for on-line kernel based stream data learning which related to research question 1.

Chapter 4

ON-LINE CORE VECTOR MACHINES - A NOVEL ONE-PASS, FAST KERNEL BASED LEARNING METHOD

This chapter presents a one-pass, fast kernel based learning method called On-line Core Vector Machine (OCVM). It is based on the minimum enclosing ball (MEB) of streaming data. We show that the MEB updates for the streaming case can be easily adapted to learn the CVM weight vector in a way similar to using on-line stochastic gradient updates.

The rest of chapter is structured as follows: Section 4.1 discusses the reason for using kernel methods for stream data mining. Section 4.2 reviews the batch mode MEB and CVM; Section 4.3 presents the novel OCVM method based on CVM. The evaluation of OCVM on 12 benchmark datasets is presented in Section 4.4. Section 4.5 gives the conclusion.

4.1 WHY USE ON-LINE KERNEL METHODS FOR STREAM DATA MINING (E.G. NETWORK INTRUSION DETECTION)? - ANSWER FOR QUESTION 1

The existing machine learning methods for Network Intrusion Detection (NID) fall into two categories: unsupervised learning and supervised learning. Unsupervised learning methods, such as clustering, are useful for NIDS, as malicious activities could be clustered, and better distinguished from non-malicious activities. Some successful stories have been reported in literature, Frank and Mda-c (1994) demonstrated that clustering is an effective way to find hidden patterns in data that humans might otherwise miss; Eskin, Arnold, Prerau, Portnoy, and Stolfo (2002) detected network connection logs outliers which represent anomalies in the network traffic using fixed-width and k-nearest neighbour; and Marin, Ragsdale, and Sirdu (2001) detected network intrusions by learning vector quantization (LVQ) and locating the Bayes Optimal boundary between classes.

Unlike unsupervised learning capturing threats through outliers detection, supervised learning detects network intrusion by using classification methods, categorising network traffics into normal and abnormal groups. Diversity of classification methods have been proposed for NID applications. Frank and Mda-c (1994) used recurrent neural networks, and some other neural networks, such as Kohonen, Hoppfield for intrusion detection. Dickerson, Juslin, Koukousoula, and Dickerson (2001) classified intrusions based on the combination of various statistical metrics and fuzzy logic rules. Apart from that, it is worth noting that SVM, because of its promising classification performance, is popularly used for NIDS. W.-H. Chen, Hsu, and Shen (2005) used a single polynomial kernel SVM for classifying network intrusions, which outperformed artificial neural networks (ANNs) method on both computational time and classification accuracy. Also, Mukkamala and Sung (2003) used multiple SVM's for identifying normal traffic against four types of malicious activity, which demonstrated the super discriminability of SVM on intrusion detection.

In spite of the current success of SVM in NID applications, it is too slow for NIDS due to the high-speed computing requirement from the Internet. Thus, we need on-line kernel based methods because on-line learning is concerned with learning data as the system operates (usually in real time) and the data might exist only for a short time (Kasabov & Song, 2002).

4.2 CORE VECTOR MACHINES

Standard SVM training has $O(m^3)$ time and $O(m^2)$ space complexities, where m is the training set size. It is thus computationally infeasible on very large data sets. By observing that practical SVM implementations only approximate the optimal solution by an iterative strategy, Tsang, Kwok, and Cheung (2005) scaled up kernel methods by exploiting such "approximateness"'. They found that many kernel methods can be equivalently formulated as MEB problems in computational geometry. Then, by adopting an efficient approximate MEB algorithm, Tsang obtained provably approximately optimal solutions with the idea of core sets. The CVM algorithm can be used with non-linear kernels and has a time complexity that is linear in m and a space complexity that is independent of m.

4.2.1 *Minimum Enclosing Ball*

Given a set of points $S = \{x_1, \ldots, x_m\}$, $x_i \in \Re^d$, the minimum enclosing ball of S is the smallest ball that contains all the points in S (denoted by MEB(S)). Welzl (1991) proposed $(1+\epsilon)$-approximation MEB which can be efficiently obtained based on those called Core Set. The core set is a subset of given dataset that contains the instances located in the outer area.

Let $B_S(\mathbf{c}, r)$ be an exact MEB of the data set S with center $\mathbf{c}$ and radius r, and $B_Q(\widetilde{\mathbf{c}}, \widetilde{r})$ be another exact MEB with center $\widetilde{\mathbf{c}}$ and radius $\widetilde{r}$. Note that, different from B_S, MEB B_Q is constructed on the Core Set of $S : \mathbf{Q}, \mathbf{Q} \subset S$. Given an $\epsilon > 0$, a ball $B_Q(\widetilde{\mathbf{c}}, (1+\epsilon)\widetilde{r})$ is a $(1+\epsilon)$-approximation of $B_S(\mathbf{c}, r)$, if $S \supset B_Q(\widetilde{\mathbf{c}}, (1+\epsilon)\widetilde{r})$ and $\widetilde{r} \leq r$.

Formally, subset $\mathbf{Q}$ is judged as the core set of S, if an expansion by a factor $(1+\epsilon)$ of its MEB contains S (i.e. $S \subset B_Q(\widetilde{\mathbf{c}}, (1+\epsilon)\widetilde{r})$. Fig. 4.1 gives an example of exact MEB, Core set MEB, and Core set MEB expansion, where the dotted-line circle identifies the exact MEB of the entire dataset B_S, and the inner solid-line circle gives the exact MEB of Core set B_Q (denoted in square). B_Q does not cover the whole data points, but its $(1+\epsilon)$ expansion (the outside circle) does.

4.2.2 *Kernel MEB: Core Vector Machine*

By adopting the above MEB algorithm to enhance a support vector machine (SVM), Tsang developed the kernel MEB method called core vector machine.

Let φ be the feature map using kernel κ, and provides a set of φ-mapped points $S_\varphi = \{\varphi(x_1), \ldots, \varphi(x_n)\}$, the MEB of S_φ is such a smallest ball $B(\mathbf{c}^*, r^*)$

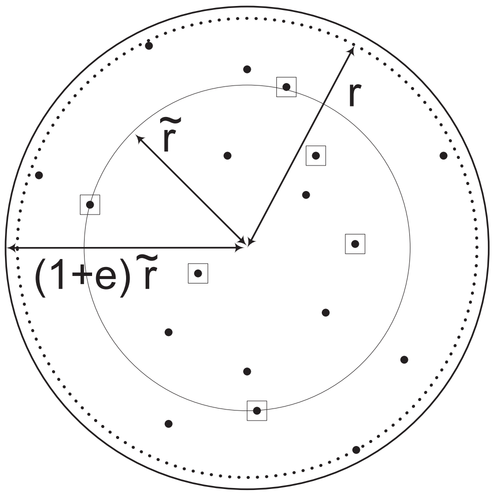


Figure 4.1: An example of exact MEB, Core set MEB, and Core set MEB expansion

Input: Set of points $X \in R^d$; parameter $\epsilon = 2^{-m}$; subset $Q_0 \subset X$
Output: A batch mode MEB learning model Ω

```
1:  for $i \leftarrow 1$ to $m$ do
2:      $\mathbf{Q} \leftarrow Q_{i-1}$
3:      Compute MEB center $\mathbf{c} \leftarrow \sum_{i=1}^{m} \alpha_i \varphi(x_i)$ where $x_i \in \mathbf{Q}$
        radius $r \leftarrow -\sqrt{\alpha' diag(K) - \alpha' K \alpha}$
4:      $includeAll \leftarrow true$, $p \leftarrow (1+\epsilon) * r$
5:      for $j \leftarrow 1$ to $|X|$ do
6:          if $p < ||x_j - \mathbf{c}||$ then
7:              $includeAll \leftarrow false$
8:              $p \leftarrow ||x_j - \mathbf{c}||$
9:              $q \leftarrow x_j$
10:         end if
11:     end for
12:     if $includeAll == true$ then
13:         Return $\Omega \leftarrow \{\mathbf{c}, r, \mathbf{Q}\}$
14:     end if
15:     $Q_i \leftarrow Q_{i-1} \cup \{q\}$
16: end for
17: Return $\Omega \leftarrow \{\mathbf{c}, r, \mathbf{Q}\}$
```

Algorithm 2: Original MEB algorithm

that encloses all data points of S_φ and has the ball center $\mathbf{c}^*$ and radius r^* determined by,

$$(\mathbf{c}^*, r^*) = argmin_{\mathbf{c},r} r^2 : \|\mathbf{c} - \varphi(x_i)\|^2 \leq r^2 \ \forall i. \tag{4.1}$$

By the corresponding Wolfe dual,

$$max_{\alpha_i} \sum_{i=1}^{m} \alpha_i k(x_i, x_i) - \sum_{i,j=1}^{m} \alpha_i \alpha_j k(x_i, x_j), \alpha \geq 0, i = 1, \ldots, m, \sum_{i=1}^{m} \alpha_i = 1, \quad (4.2)$$

where $\alpha = [\alpha_1, \ldots, \alpha_m]$ are the Lagrange multipliers and $K_{m \times m} = [k(x_i, x_j)]$ is the kernel matrix, the MEB centre **c** and radius r can be found from the optimal α as (Tsang et al., 2005)

$$\mathbf{c} = \sum_{i=1}^{m} \alpha_i \varphi(x_i), r = -\sqrt{\alpha' diag(K) - \alpha' K \alpha}. \quad (4.3)$$

In a kernel MEB, a core set instance is called a core vector, and the constructed supervised learning model constructed by core vectors, is called a core vector machine.

4.2.3 *Time and Space Complexities for CVM*

Existing decomposition algorithms cannot guarantee the number of iterations and consequently the overall time complexity (Chang & Lin, 2001). This section shows how this can be obtained for CVM. In the following, we assume that a plain Quadratic Programming (QP) implementation, which takes $O(m3)$ time and $O(m2)$ space for m patterns, is used for the QP sub-problem. The time and space complexities obtained below can be further improved if more efficient QP solvers were used. Moreover, each kernel evaluation is assumed to take constant time.

Consider first the case where probabilistic speedup is not used. As proved in (Bā̄doiu, Har-Peled, & Indyk, 2002), CVM converges in $2/\epsilon$ iterations at the most. In other words, the total number of iterations, and consequently the size of the final core set, are $\tau = O(1/\epsilon)$. In practice, it has often been observed that the size of the core set is much smaller than this worst-case theoretical upper bound (Kumar, Mitchell, & Yildirim, 2003). As only one core vector is added at each iteration, $|S_t| = t + 2$. Initialisation takes $O(m)$ time while distance computations take $O((t + 2)^2 + tm) = O(t^2 + tm)$ time. Finding the MEB takes $O((t + 2)^3) = O(t^3)$ time, and the other operations take constant time. Hence, the t^{th} iteration takes a total of $O(tm + t^3)$ time. The overall time for $\tau = O(1/\epsilon)$ iterations is

$$T = \sum_{t=1}^{\tau} O(tm - t^3) = O(\tau^2 m + \tau^4) = O(\frac{m}{\epsilon^2} + \frac{1}{\epsilon^4}). \tag{4.4}$$

which is linear in m for a fixed ϵ.

Next, we consider its space complexity. As the m training patterns may be stored outside the core memory, the $O(m)$ space required will be ignored in the following. Since only the core vectors are involved in the QP, the space complexity for the t^{th} iteration is $O(|S_t|^2)$. As $\tau = O(1/\epsilon)$, the space complexity for the whole procedure is $O(1/\epsilon^2)$, which is independent of m for a fixed ϵ.

On the other hand, when probabilistic speedup is used, initialization only takes $O(1)$ time while distance computations take $O((t+2)^2) = O(t^2)$ time. Time for the other operations remains the same. Hence, the t^{th} iteration takes $O(t^3)$ time. As probabilistic speed-up may not find the furthest point in each iteration, τ may be greater than $2/\epsilon$ though it can still be bounded by $O(1/\epsilon^2)$ (Bādoiu et al., 2002). Hence, the whole procedure takes

$$T = \sum_{t=1}^{\tau} O(t^3) = O(\tau^4) = O(\frac{1}{\epsilon^8}). \tag{4.5}$$

For a fixed ϵ, it is thus independent of m. The space complexity, which depends only on the number of iterations τ, becomes $O(1/\epsilon^4)$.

When ϵ decreases, the CVM solution becomes closer to the exact optimal solution, but at the expense of higher time and space complexities. Such a tradeoff between efficiency and approximation quality is typical of all approximation schemes. Moreover, be cautioned that the O-notation is used for studying the asymptotic efficiency of algorithms. As we are interested in handling very large data sets, an algorithm that is asymptotically more efficient (in time and space) will be the best choice. However, on smaller problems, this may be outperformed by algorithms that are not as efficient asymptotically.

4.3 THE PROPOSED ON-LINE CORE VECTOR MACHINES

This section introduces the proposed on-line CVM method for classification tasks.

4.3.1 *Motivations for using CVM for Incremental Learning*

We found that the CVM updates for the streaming case can be easily adapted to learn the SVM weight vector in a way similar to using online stochastic gradient updates. D. Wang, Zhang, Zhang, and Qiao (2010) addressed this issue by proposing an online CVM classifier with adaptive minimum-enclosing-ball (MEB) adjustment. This method performs polylogarithmic computation at each example, and requires very small and constant storage. However, the computational cost of Wang's algorithm is same as original CVM. To reduce not only the constant storage requirement but also the learning time, we proposed our on-line CVM learning algorithm that perform each updating using geometry techniques. As the updating processing, which is the most important part of incremental learning, has been changed to a much faster algorithm, our OCVM performs more efficiently. Experimental results show that, even in such restrictive settings, the algorithm can learn efficiently in just one pass and demonstrates accuracies comparable to other methods.

4.3.2 *On-line Mode CVM Learning*

Let $\mathbf{x}'$ be the input point causing an update to the MEB and B' be the resulting ball after the update. From figure 4.2, it is easy to verify that the new centre $\mathbf{c}'$ lies on the line joining the old center $\mathbf{c}$ and the new point $\mathbf{x}'$. The radius r' and the centre $\mathbf{c}'$ of the resulting MEB can be defined by simple update equations as follows:

$$r' = r + \delta, \tag{4.6}$$

$$\delta = ||\mathbf{c}' - \mathbf{c}||. \tag{4.7}$$

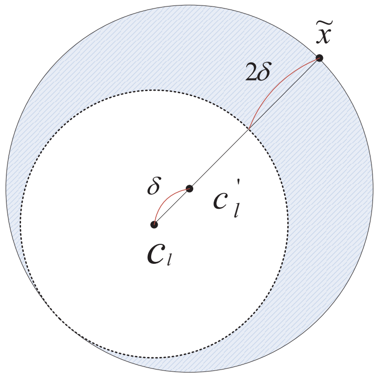


Figure 4.2: MEB expansion

Hence 2δ, which is the closest distance of the new point $\mathbf{x}'$ from the old ball B, can be defined as:

$$2\delta = (||\mathbf{x}' - \mathbf{c}|| - r) \tag{4.8}$$

Using these, we can define a closed-form analytical update equation for the new ball B':

$$\mathbf{c}' = \mathbf{c} + \frac{\delta}{||\mathbf{x}' - \mathbf{c}||}(\mathbf{x}' - \mathbf{c}). \tag{4.9}$$

Thus, the updated MEB learning model Ω' is obtained by conducting the following algorithm.

Kernelised On-line CVM

In order to extend the usability of OCVM from linear kernels to non-linear kernels, we modify Algorithm 3. Instead of storing the weight vector $\mathbf{c} = \frac{\sum_{x_l \in Q}}{|Q|}$, it stores Lagrange coefficients $\mathbf{c} \leftarrow \sum_{i=1}^{m} \alpha_i \varphi(\mathbf{x}_i)$ as the center of MEB. The distance from an incoming data sample $\mathbf{x}_i'$ to the centre is calculated as

$$||\mathbf{x}_i' - \mathbf{c}||^2 = \sum_{\mathbf{x}_j, \mathbf{x}_l \in \mathbf{Q}} \alpha_j \alpha_l k(\mathbf{x}_j, \mathbf{x}_l) + k(\mathbf{x}_i, \mathbf{x}_i) - 2 \sum_{\mathbf{x}_j \in \mathbf{Q}} \alpha_j k(\mathbf{x}_j, \mathbf{x}_j) + \xi^2. \tag{4.10}$$

```
Input: Original CVM learning model $\Omega = \{\mathbf{c}, r, \mathbf{Q}\}$, incoming data matrix
    $X'$, MEB parameter $\epsilon$, and slack parameter $\xi$
Output: A batch mode MEB learning model $\Omega'$
 1: Initial $\mathbf{c}' \leftarrow \mathbf{c}, r' \leftarrow r$, and $\mathbf{Q}' \leftarrow \mathbf{Q}$
 2: for $i \leftarrow 1$ to $|X'|$ do
 3:   Compute distance from $x_i'$ to $\mathbf{c}$
      $d = ||x_i' - \mathbf{c}|| + \sqrt{1/\xi}$
 4:   Compute MEB center $\mathbf{c}$ and radius $r$ on subset $\mathbf{Q}$ using equation 4.3
 5:   if $(1+\epsilon) * r < d$ then
 6:     $\delta \leftarrow \frac{(||x_i' - \mathbf{c}|| - r)}{2}$
 7:     $\mathbf{c}' \leftarrow \mathbf{c} + (\frac{1}{2} - \frac{r}{2||x_i' - \mathbf{c}||})(\mathbf{x}_i' - \mathbf{c})$
 8:     $r' \leftarrow r + \frac{(||\mathbf{x}_i' - \mathbf{c}|| - r)}{2}$
 9:     $\mathbf{Q}' \leftarrow \mathbf{Q} \cup \mathbf{x}_i'$
10:   end if
11: end for
12: Return $\Omega' \leftarrow \{\mathbf{c}', r', \mathbf{Q}'\}$
```

Algorithm 3: Incremental mode On-line CVM algorithm

Once the new sample $\mathbf{x}_i'$ updates the MEB then $\mathbf{Q}' = \mathbf{Q} \cup \varphi(\mathbf{x}_i')$, the Lagrange coefficient for the new core vector is $\alpha_i = \frac{1}{2} - \frac{r}{2||\mathbf{x}_i' - \mathbf{c}||}$.

Contraction of On-line CVM

During the expanding of MEB, spare area is inevitably presented because the size of the MEB depends on the most significant outlier. In order to preserve the efficiency of MEB knowledge encryption, a contraction is conducted once an outlier is found. The approach of MEB contraction is similar to the expansion approach.

Given $q \in \widetilde{\mathbf{Q}}$ where $\widetilde{\mathbf{Q}} \subseteq \mathbf{Q} - Q_{i-1}$, and $\forall \widetilde{q} \in \widetilde{\mathbf{Q}}$ where $\widetilde{q} \neq q$, $\neg\exists ||\widetilde{q} - c_{i-1}|| < ||q - c_{i-1}||$. From equation 4.7 and 4.8, the relationship between δ and δ' is defined as $\delta' = 2\delta$. From figure 4.3, radius r_{i-1} for the original MEB B_{i-1} and radius r_i for the updated MEB B_i are:

$$r_{i-1} = r_i + \delta + ||q - c_{i-1}|| + \delta', \tag{4.11}$$

$$r_i = ||q - c_{i-1}|| + \delta. \tag{4.12}$$

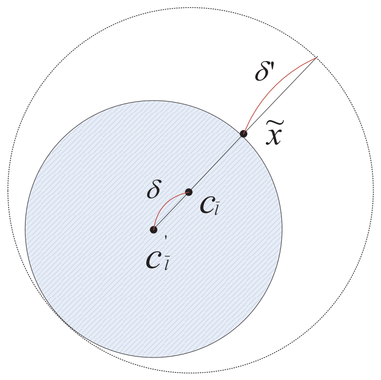


Figure 4.3: MEB contraction

From equation 4.7 and 4.8, the relationship between δ and δ' is defined as $\delta' = 2\delta$. Thus, equation 4.11 is transformed as:

$$r_{i-1} = ||q - c_{i-1}|| + \delta + \delta + ||q - c_{i-1}|| + \delta', \tag{4.13}$$

$$r_{i-1} = 2||q - c_{i-1}|| + 3\delta, \tag{4.14}$$

$$\delta = \frac{r_{i-1} - 2||q - c_{i-1}||}{3}. \tag{4.15}$$

Once we have the value for δ we can easily calculate r_i and updated MEB center c_i as:

$$r_i = \frac{r_{i-1} + ||q - c_{i-1}||}{3}, \tag{4.16}$$

$$c_i = c_{i-1} + \frac{\delta}{||q - c_{i-1}||}(q - c_{i-1}). \tag{4.17}$$

4.4 EXPERIMENTS AND DISCUSSIONS

To evaluate the performance of on-line CVM we compared the classification accuracy and the computational cost with two conventional classification methods on six two-class and six multi-class datasets. Table 4.1 and table 4.2 summarise the characteristics of 12 datasets and five classification methods respectively.

Type	Dataset	No. of Features	No. of Classes	Complexity	Size
two-class	synthetic binary	2	2	0.0743	200
	liver-disorder	6	2	0.5432	345
	breast-cancer	10	2	0.1743	683
	heart	13	2	0.7632	270
	ionosphere	34	2	0.5163	351
	web Spam	57	2	5.5188	4601
multi-class	synthetic multi-class	2	3	0.0743	200
	iris	4	3	0.0636	150
	wine	13	3	0.3007	178
	vehicle	18	4	2.7264	846
	vowel	10	11	5.1335	990
	KDD99(KDD99, 1999)	42	5	0.5163	100000

Table 4.1: *Summarisation of data sets characteristics*

Notation	**Descriptions**	**Parameters**
KNN	k-nearest neighbor	k
Bayes	Naive Bayes	None
MLP	Multi layer perception	α
SVM	Support vector machine	Kernel Method (KM) Kernel Parameter (KP)
OCVM	Hierarchical minimum enclosing ball	Kernel Method (KM) Kernel Parameter (KP) Epsilon (ϵ)

Table 4.2: *Summarisation of classifiers characteristics*

Accuracy of Binary Classification

For each two-class dataset, we conducted K-folds cross validation which randomly partitioned original data into K subsamples. Of the K subsamples, a single subsample was retained as the validation data for testing the model, and the remaining $K-1$ subsamples are used as training data. The cross-validation process was repeated K times, with each of the K subsamples used exactly once as the validation data. Then we can obtain the mean value and standard deviation (std) value of the K results from the folds. In our case, we set $K = 10$ as 10-fold cross-validation is commonly used.

Table 6.3 shows the classification accuracy of the two-class datasets using different classifiers. The accuracy is represented by the mean value and the std value of the 10-fold cross validation. To make sure that the results for the conventional methods are not biased by an inappropriate choice of parameters, we optimised all parameters (see table 4.4) by running cross validation.

On the two-class datasets, MLP and SVM achieved the two highest accuracies among the conventional methods. Compared to the conventional methods, OCVM obtained higher accuracies on three out of six datasets(i.e. breast-cancer, ionosphere and web Spam). It can be found from table 4.1, these three datasets have greater number of features than the other datasets. That indicates the proposed OCVM is effective for classifying higher dimensional datasets.

Dataset	KNN	Bayes	MLP	SVM	OCVM
Syn binary	$93.31\% \pm 1.65\%$	$89.93\% \pm 3.12\%$	$93.78\% \pm 2.51\%$	$\mathbf{94.65\% \pm 2.18\%}$	$88.13\% \pm 2.53\%$
liver-disorder	$64.92\% \pm 6.55\%$	$63.17\% \pm 6.31\%$	$\mathbf{71.28\% \pm 7.71\%}$	$64.92\% \pm 1.09\%$	$59.13\% \pm 4.91\%$
breast-cancer	$96.63\% \pm 1.71\%$	$95.97\% \pm 2.44\%$	$96.65\% \pm 2.38\%$	$\mathbf{97.07\% \pm 3.07\%}$	$\mathbf{97.07\% \pm 2.82\%}$
heart	$82.59\% \pm 8.20\%$	$75.65\% \pm 8.25\%$	$\mathbf{83.33\% \pm 5.25\%}$	$83.19\% \pm 6.06\%$	$82.67\% \pm 6.33\%$
ionosphere	$64.11\% \pm 1.09\%$	$65.32\% \pm 1.74\%$	$89.16\% \pm 5.35\%$	$92.06\% \pm 4.46\%$	$\mathbf{93.73\% \pm 4.57\%}$
web Spam	$93.32\% \pm 0.78\%$	$84.13\% \pm 1.97\%$	$94.62\% \pm 0.12\%$	$94.60\% \pm 0.10\%$	$\mathbf{94.64\% \pm 0.06\%}$

Table 4.3: *Classification accuracy comparison for five methods applied on the six two-class datasets*

classifier	**parameters**	liver-disorder	breast-cancer	heart	ionosphere	web Spam
KNN	k	3	5	5	3	5
MLP	α	100	0.5	10	2	100
SVM	KM	RBF	RBF	RBF	RBF	RBF
	KP	0.001	0.5	0.01	0.01	0.001
OCVM	KM	RBF	RBF	RBF	RBF	RBF
	$KP(\gamma)$	4	3	4	2	4
	ϵ	$1e-6$	$1e-2$	$1e-3$	$1e-6$	$1e-3$

Table 4.4: *Summarisation of classifiers' parameters for six two-class datasets*

Accuracy of Multi-class Classification

Similarly to the experiments with two-class datasets, we conduct 10-fold cross validation with optimised parameters (see table 4.6) on multi-class dataset. The classification accuracy was represented by the 10-folds accuracies mean value and std value in shown in table 4.5.

On the multi-class datasets, except Naive Bayes all selected conventional methods perform similar. Compared to the conventional methods, OCVM which showed the 3 highest accuracies, performs better specially on KDD99 which has a much larger number of samples.

Dataset	KNN	Bayes	MLP	SVM	OCVM
Syn m-class	$94.53\% \pm 1.14\%$	$88.57\% \pm 4.29\%$	$93.86\% \pm 2.68\%$	$\mathbf{94.97\% \pm 1.99\%}$	$90.95\% \pm 3.67\%$
iris	$94.67\% \pm 6.13\%$	$90.35\% \pm 7.11\%$	$95.97\% \pm 4.03\%$	$96.00\% \pm 5.62\%$	$\mathbf{96.66\% \pm 4.24\%}$
wine	$\mathbf{97.16\% \pm 3.00\%}$	$71.31\% \pm 8.53\%$	$93.91\% \pm 8.01\%$	$74.85\% \pm 9.15\%$	$76.40\% \pm 8.22\%$
vehicle	$71.04\% \pm 3.01\%$	$66.24\% \pm 5.57\%$	$\mathbf{77.78 \pm 3.96\%}$	$71.61\% \pm 3.88\%$	$71.61\% \pm 3.88\%$
vowel	$96.89\% \pm 1.00\%$	$91.37\% \pm 3.25\%$	$98.08\% \pm 1.21\%$	$97.47\% \pm 1.73\%$	$\mathbf{98.56\% \pm 1.34\%}$
KDD99	$89.32\% \pm 1.12\%$	$75.21\% \pm 5.32\%$	$90.53\% \pm 2.83\%$	$90.42\% \pm 2.18\%$	$\mathbf{94.95\% \pm 1.47\%}$

Table 4.5: *Classification accuracy comparison for five methods applied on the six multi-class datasets*

classifier	**parameters**	iris	wine	vehicle	vowel	KDD99
KNN	k	3	5	5	3	3
MLP	α	0.001	100	0.5	0.2	5
SVM	KM	RBF	RBF	RBF	RBF	RBF
	$KP(\gamma)$	0.01	0.001	0.001	0.5	0.05
OCVM	KM	RBF	RBF	RBF	RBF	RBF
	$KP(\gamma)$	3	2	4	4	4
	ϵ	$1e-6$	$1e-2$	$1e-6$	$1e-3$	$1e-4$

Table 4.6: *Summarisation of classifiers' parameters for six multi-class datasets*

Computational Costs and Speed Efficiency

We measured the computation cost by analysing the training time and number of the support vectors (for SVM and OCVM). For each dataset, we train the model based on 10% samples up to 100% step by 10%.

Results are shown in Table 4.7. As it can be seen, the CPU time cost by SVM shot up with the growth of the training size, but it is the shortest while the size of training set is smaller than 300. OCVM is much faster and produces far

Dataset	Classifier	$T_{10\%}$	$T_{20\%}$	$T_{30\%}$	$T_{40\%}$	$T_{50\%}$	$T_{60\%}$	$T_{70\%}$	$T_{80\%}$	$T_{90\%}$	$T_{100\%}$
breast-cancer	SVM	**≈ 0**	**0.031**	**0.062**	**0.062**	**0.156**	0.203	0.39	0.499	0.78	0.967
	MLP	0.437	0.343	0.562	0.624	0.811	0.92	0.983	1.326	1.56	1.42
	OCVM	0.156	0.094	0.156	0.203	0.172	**0.14**	**0.218**	**0.078**	**0.125**	**0.125**
liver-disorder	SVM	**≈ 0**	**≈ 0**	**≈ 0**	**0.031**	**0.031**	**0.031**	**0.062**	**0.062**	**0.062**	**0.062**
	MLP	0.811	0.858	0.889	0.998	1.029	1.138	1.232	1.123	1.31	1.357
	OCVM	0.156	0.094	0.094	0.094	0.125	0.109	**0.062**	0.078	**0.062**	0.094
heart	SVM	**≈ 0**	**≈ 0**	**≈ 0**	**0.031**	**≈ 0**	**0.016**	**0.016**	**0.031**	**0.062**	**0.062**
	MLP	0.811	0.858	0.889	0.998	1.03	1.139	1.232	1.123	1.31	1.357
	OCVM	0.25	0.14	0.12	0.187	0.109	0.25	0.187	0.156	0.156	0.156
web Spam	SVM	**0.56**	4.41	14.38	34.36	40.56	79.15	140.63	241.63	419.94	605.61
	MLP	4.58	5.6	5.78	6.41	6.92	11.88	8.47	10.92	9.89	10.29
	OCVM	0.8	**0.936**	**0.858**	**0.889**	**0.842**	**0.75**	**0.905**	**0.905**	**0.874**	**0.8**
ionosphere	SVM	**≈ 0**	**≈ 0**	**≈ 0**	**0.031**	**0.062**	**0.046**	**0.062**	**0.125**	**0.125**	**0.156**
	MLP	1.076	1.294	1.232	1.248	1.638	1.373	1.591	1.638	1.747	1.825
	OCVM	0.234	0.25	0.25	0.187	0.25	0.3432	0.1716	0.312	0.187	0.187
iris	SVM	**≈ 0**	**0.031**	**≈ 0**	**0.031**	**≈ 0**	**0.031**	**0.031**	**0.062**	**0.062**	**0.062**
	MLP	0.234	0.312	0.296	0.437	0.4056	0.499	0.312	0.358	0.421	0.484
	OCVM	0.109	0.078	0.062	0.078	0.078	0.078	0.062	0.078	0.062	0.109
wine	SVM	**≈ 0**	**0.016**	**0.016**	**0.016**	**0.062**	**0.062**	**0.062**	**0.078**	0.171	0.203
	MLP	1.107	0.982	0.858	0.795	0.982	1.014	1.326	1.372	1.310	1.076
	OCVM	0.078	0.125	0.062	0.078	0.093	0.203	0.109	0.125	**0.062**	**0.062**
vehicle	SVM	**0.047**	**0.125**	**0.39**	0.718	1.279	2.714	4.321	6.864	9.329	13.245
	MLP	1.06	1.123	1.326	1.279	1.388	1.622	1.841	1.778	1.762	2.012
	OCVM	0.125	0.172	0.156	**0.14**	**0.125**	**0.187**	**0.187**	**0.218**	**0.218**	**0.14**
vowel	SVM	**0.109**	**0.437**	0.983	2.636	5.834	9.547	14.929	22.48	33.135	48.048
	MLP	0.53	0.686	0.842	1.123	1.341	1.451	1.653	1.95	1.794	2.34
	OCVM	0.468	0.515	**0.546**	**0.53**	**0.53**	**0.593**	**0.546**	**0.64**	**0.64**	**0.499**
KDD99	SVM	1471.2	OM	OM	OM	OM	OM	OM	OM	OM	OM
	MLP	12.948	21.887	27.456	35.334	38.033	53.805	74.116	98.327	109.901	153.957
	OCVM	**7.21**	**7.503**	**7.871**	**7.323**	**7.525**	**7.619**	**7.28**	**7.324**	**7.234**	**7.101**

Table 4.7: *Comparison of training CPU time (in seconds). OM indicates out of memory caused by size of data input that was too large.*

fewer support vectors (which implies faster testing) on large data sets. In particular, for web Spam data 5000 samples can be processed in less than 1 second, which is 750 times faster than SVM. As we developed the evaluation environment based on MATLAB, the largest memory is restricted. Thus, $10,000$ is the maximum number of samples that can be used by SVM and training takes 200 times more CPU time than OCVM. The CPU time cost by MLP is not obviously greater than OCVM on a small training set. However, it is easy to see that only OCVM keeps consistent speed once the size of training data is over 5000.

The comparison result for number of support vectors, which is shown in Table 4.8, is similar to the CPU time comparison. For relatively small training sets, with fewer than 200 samples, the SVM produces less support vectors. However, the number of support vectors obtained using SVM hoicks. Af-

Dataset	Classifier	$T_{10\%}$	$T_{20\%}$	$T_{30\%}$	$T_{40\%}$	$T_{50\%}$	$T_{60\%}$	$T_{70\%}$	$T_{80\%}$	$T_{90\%}$	$T_{100\%}$
breast-cancer	SVM	**34**	**68**	**102**	**136**	**170**	204	238	272	306	340
	OCVM	215	178	**102**	133	223	**196**	**166**	**136**	**118**	**176**
liver-disorder	SVM	**17**	**34**	**51**	**68**	**85**	**102**	**119**	**136**	**153**	**170**
	OCVM	171	171	171	170	170	170	171	170	170	171
heart	SVM	**13**	**27**	**40**	**54**	**67**	**81**	**94**	**108**	121	135
	OCVM	99	103	83	103	100	106	97	116	**78**	**101**
web Spam	SVM	**192**	384	576	768	960	1152	1344	1536	1728	1920
	OCVM	294	**296**	**294**	**295**	**294**	**298**	**294**	**299**	**295**	**294**
ionosphere	SVM	**17**	**35**	**52**	**70**	**87**	**105**	**122**	140	157	175
	OCVM	140	137	139	105	122	123	135	**133**	**130**	**145**
iris	SVM	**7**	**15**	**22**	**30**	**37**	**45**	**52**	**60**	**67**	75
	OCVM	67	70	70	64	67	69	64	67	70	**66**
wine	SVM	**8**	**17**	**25**	**34**	**42**	**51**	**59**	**68**	**76**	**85**
	OCVM	87	86	85	86	86	86	86	86	87	86
vehicle	SVM	**42**	84	126	168	210	252	294	336	378	420
	OCVM	71	**72**	**72**	**70**	**72**	**72**	**70**	**73**	**73**	**72**
vowel	SVM	**49**	**99**	148	198	247	297	346	396	445	495
	OCVM	111	110	**110**	**110**	**112**	**110**	**110**	**112**	**111**	**110**
KDD99	SVM	2500	OM	OM	OM	OM	OM	OM	OM	OM	OM
	OCVM	**122**	**134**	**134**	**142**	**145**	**145**	**146**	**146**	**147**	**148**

Table 4.8: *Number of support vectors/core vectors comparison. OM indicates out of memory caused by too large size of data input.*

ter training $10,000$ samples, SVM created 20 times more support vectors than OCVM.

4.5 SUMMARY

Within the streaming framework for learning, we have presented an efficient, on-line CVM learning algorithm using a streaming algorithm for the MEB problem. OCVM, incrementally learning the stream data by extending the corresponding MEB towards a new sample which located outside of the existing MEB and this sample will be saved by updating the core vectors. In addition, to preserve the efficiency of MEB knowledge encryption, a contraction is conducted once an outlier is found. The experiments on batch mark datasets show the on-line CVM obtained similar or higher classification accuracy than the selected bench learning methods by taking significantly less computational time.

In the next chapter, a novel method for hierarchical structure stream data modelling is presented.

Chapter 5

HIERARCHICAL CORE VECTOR MACHINE - A NOVEL HIGH-SPEED HIERARCHICAL MULTI-LABEL CLASSIFICATION ALGORITHM

This chapter presents a high-speed hierarchical multi-label classification algorithm, called the hierarchical core vector machine (HCVM), for labelling hierarchical multi-label structure stream data.

We model the multi-label hierarchy into a data Hyper-Sphere constructed by numbers of minimum enclosing balls/core vector machines (MEB/CVM). The MEBs/CVMs are separating, encompassing and overlapping with each other formed as a tree structure, representing the multi-label hierarchy encoded. Provided an unlabelled sample, the HCVM seeks a MEB/CVM enclosing the sample, and multi-label the sample according to the MEB's position in the hierarchy. The proposed HCVM has been examined on a Gaussian synthetic data.

This chapter is structured as follows: Section 5.1 describes related researches on machine learning method for network intrusion detection and multi-label classification methods. Section 5.2 presents the proposed HCVM learning. In Section 5.3, we cover experimentation and algorithm evaluation. Lastly, in Section 5.4 we draw our conclusion and state future directions.

5.1 WHY HIERARCHICAL LEARNING? - ANSWER FOR QUESTION 2

Previous supervised learning methods group any network traffic into several major clusters, disregarding the complexity fact that one major category communication includes often several subcategories, and so on for those subcategories. In general, network traffic poses, by its nature, a complex multi-label, rather than a single-label status.

5.1.1 *Review of Multi-label Classification Methods*

The existing multi-label classification methods can be grouped into two main categories: a) problem transformation methods and b) algorithm adaptation methods.

The problem transformation methods were defined in (Tsoumakas & Katakis, 2007) as those methods that transform a multi-label classification problem into a multiple single-label's classification problem (Boutell, 2004; Diplaris, 2005). Given a dataset X with a set of class labels L for multi-label classification training, a common problem transformation method is to train $|L|$ binary classifiers $H_l : X \to \{l, \neg l\}$, one for each individual label l in L. Thus, the original data set is transformed into $|L|$ data sets, for the l^{th} dataset D_l the same instance of the original dataset whose instances are labelled as l or $\neg l$, and a binary classifier is applied for the classification on the dataset. In this way, when an unlabeled instance x is provided for classification, a set of labels are produced simultaneously by $|L|$ classifiers,

$$H(x) = \bigcup_{l \in L} \{l\} : H_l(x) = l. \tag{5.1}$$

Algorithm adaptation methods extend the existing learning algorithms such as AdaBoost to handle multi label data directly. Adaboost.MH and Adaboost.MR (Schapire, 2000) are two extensions of AdaBoost (Freund, 1997) for multi-label classification. Both extensions are used on weak classifiers represented in the form $H : X \times L \to R$. Given an unlabelled instance x, in AdaBoost.MH, x is labelled as $l, l \in L$ if the weak classifier for l is positive. In AdaBoost.MR, the output of the weak classifiers is considered for ranking each of the labels in L and x is labelled as l which is the top rank class label.

5.2 THE PROPOSED HCVM LEARNING METHOD

This section introduces the proposed HCVM method for hierarchical multi-label classification in the context of network security stream data.

5.2.1 *MEB/CVM and Motivations of CVM for HMC*

The minimum enclosing ball computes the ball(s) of minimum radius enclosing a given set of points. Traditional algorithms for finding exact MEBs/CVMs developed by Megiddo (Megiddo, 1983) and Welzl (Welzl, 1991) do not scale well with higher dimensional datasets. Recently, approximation algorithms for finding MEBs/CVMs have been given by Badoiu (Bādoiu et al., 2002) and Kumar (Kumar et al., 2003). Badoiu indicated that a $(1+\epsilon)-approximation$ MEB can be efficiently obtained by using a subset of the input dataset, called the core set. Additionally, Badoiu (Bādoiu et al., 2002) found that the size of the MEB core set is independent of both dataset dimensionality d and the size of the dataset. Kumar et al. (Kumar et al., 2003) developed methods for computing core sets and approximate the smallest enclosing hyper-spheres in high dimension spaces. In this way, the MEB can be implemented in applications with large numbers of numerical attributes.

As mentioned above, a straightforward method is to transfer a HMC problem to a number of single class classification problems using a certain transformation method (Boutell, 2004; Diplaris, 2005), so that a HMC problem can be managed as a typical multi-class classification. The disadvantage of this method is, the original label hierarchy information (i.e. the partial order of hierarchical multi-labels) of the data is changed or lost completely, which eventually leads to an unsatisfied multi-label classification.

Alternatively, we consider here the HMC problem from the viewpoint of MEB (Bādoiu et al., 2002; Kumar et al., 2003), for each single class of the HMC problem, wherever it is located in the label hierarchy, it can be approximated by an MEB. One class is represented as one MEB, meanwhile the MEB may overlap with MEBs/CVMs from other classes, or encompass MEBs/CVMs from its child classes. In this way, the HMC problem is addressed by a set of related MEBs/CVMs, in which the relationship of MEB represents the label hierarchy, so that the original hierarchy information of data get reserved in MEB modelling.

5.2.2 *HMC Problem Transformation*

Hierarchical multi-label classification (HMC) is an extension of binary classification where an instance can be labeled with multiple classes that are organized in a hierarchy (Hendrik, Leander, Jan, & Amanda, 2006).

In the example of network intrusion detection, Fig. 5.1 describes a typical HMC adapted from the KDD'99 dataset, where different network connection types are divided into 4 major categories and 21 subcategories. The relationship between these subcategory connection types and major connection category types are structured as a Markov Tree. The major connection types are set as the branch of the tree, and those subcategory types as the leaves of the tree. In this way, the problem of Internet intrusion detection is interpolated into the learning of hierarchical multi-label classification. Here, the definition of HMC task is briefed as follows (Hendrik et al., 2006):

Given: *(1) data set $S = \{x_1, \dots, x_m\}$, where each $x_i \in \Re^d$, (2) class label set $C = \{c_1, \dots, c_n\}$, and (3) a class hierarchy $(C, \leq_h)$, where $\leq_h$ is a partial order representing the parent class relationship ($\forall c_1, c_2 \in C : c_1 \leq_h c_2$, if and only if c_1 is a superclass of c_2).*
Find: *a function $f : x \rightarrow C_i$ where $x \in S$ and $C_i \subseteq C$, C_i includes an leaf class c and its parent classes c' such that $c \in C_i \Rightarrow \forall c' \leq_h c : c' \in C_i$.*

In HMC, f is able to classify an unlabelled message as normal, or as a subcategory attack type together with all its parent category attack types.

5.2.3 *Hierarchical Minimum Enclosing Ball*

For HMC, the above kernel MEB is extended to the approximation of hierarchical minimum enclosing balls (HCVM) as follows:

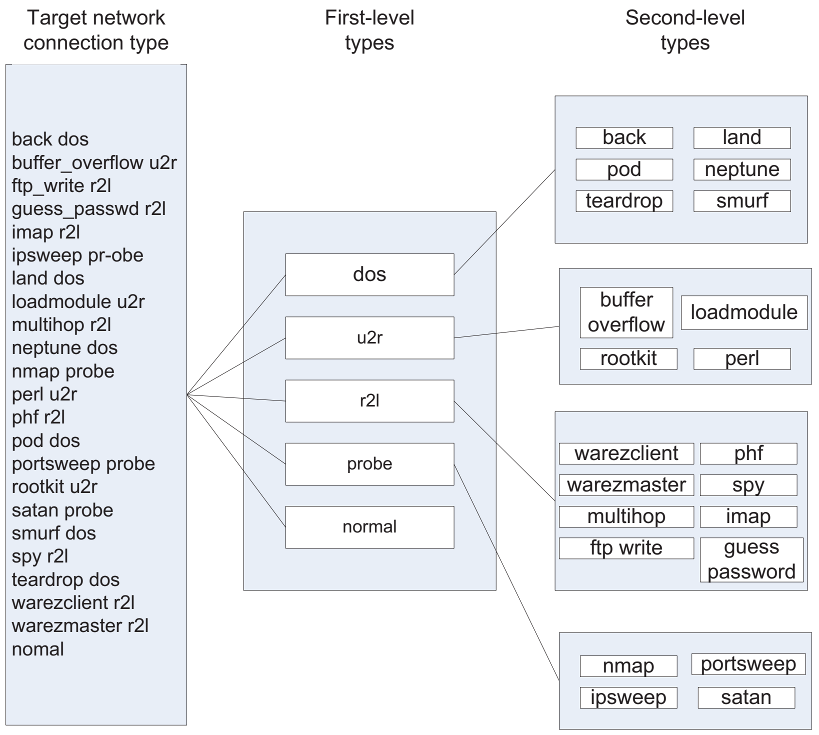


Figure 5.1: The label hierarchy of network connection types adapted from KDD'99 dataset

Given dataset $S = (X \times Y)$ where $X \in \Re^d$, and class label set Y is in such a hierarchy as $Y = \{Y_i | y \leq_h \forall y'\}$ whose $y, y' \in Y_i$, and $y' \neq y$. To transform the HMC problem to a HCVM problem, we seek the root class label set $C_1 \subseteq Y$ where $\forall y^l \in C_1 : \neg\exists y' \leq_h y^l : y' \in Y, y' \notin C_1$. For each root class label $y^l \in C_1$, we create an MEB, B, and mark it with (1) its hierarchical level which is 1, (2) the corresponding class label y^l, and (3) its parent class. (we use 0 to represent the parent class of a root class). Thus, for each class y^l in the root class label set C_1, we obtain the corresponding MEB $B_1^{y^l,0}$. Similarly, we address the next level class set $C_2 \subseteq Y$ where $\forall y^l \in C_2 : \exists p^l \leq_h y^l : p^l \in C_{2-1}$, and create the MEB $B_2^{y^l,p^l}$ for each class label $y^l \in C_2$. This process is continued until no child classes can be found.

As a result, we have a multi-label hierarchy modelled as an HCVM structured as a set of MEBs/CVMs ordered with its class label, level in the hierarchy and its parent class,

$$\{Y_i | y \leq_h \forall y'\} \rightharpoonup \{B_1^{y^l,0}, B_2^{y^l,p^l}, ..., B_i^{y^l,p^l}\}. \tag{5.2}$$

While we transfer a multi-label hierarchy to an HCVM, the knowledge of the relationship between classes at the same level naturally appears. This is because that label represents the class as a signature, but HCVM represents the class in a data distribution space, it captures not only the parenthood relationship (i.e. parent class MEB encompasses the child class MEB) but also the relationship between classes at the same level of the hierarchical structure (i.e. same level classes MEBs/CVMs may separating or overlapping).

For decision making, we expect that $f(B)$ approximates the ground truth multi-label $f(x)$ better than $f(X,Y)$ because each MEB in HCVM approximates accurately its associated class of Y, preserving perfectly the multi-label hierarchy in Y. Thus, the HCVM eventually can be learned by an aggregation of MEBs/CVMs as,

$$f(x) = f(\{B_i^{y^l,p^l}\}) = \bigvee_{i=1}^{m} \bigvee_{l=1}^{t_i} f_{B_{i,l}} \tag{5.3}$$

where $f_{B_{i,l}}$ represents an elementary MEB model on class y_i^l. In Eq. (5.3), $\bigvee$ denotes the union between models, in which $\bigvee_i$ represents a natural parent relationship. As modelling $f_{B_{i,l}}$, class y^l is addressed as a binary problem with class y^l as the positive and the remaining classes as the negative, and a set of MEBs/CVMs are computed according to Eq. (4.1)- Eq. (4.3) to enclose only the positive instances. Then, $f_{B_{i,l}}$ is formed by assigning the corresponding MEB core sets as core vectors.

Given an instance x enclosed in one i^{th}-layer MEB, due to the parent relationship, it must also be enclosed in the $(i-1)^{th}, (i-2)^{th}, ..., 1st, 0$ layer ancestor MEB; and $\bigvee_l$ represents a relationship at the same layer of HCVM. l is often fixed by distances calculation. To compute the distance between x and the center c_t of B_t, x is mapped into the feature space by a kernel function $k(x,x)$. By Eq. (4.3), the distance between x and c_t is calculated as:

$$\|c_t - \varphi(x)\|^2 = \sum_{z_i,z_j \in S_t} \alpha_i \alpha_j k(z_i,z_j) - 2\sum_{z_i \in S_t} \alpha_i k(z_i,x) + k(x,x). \tag{5.4}$$

Fig. 5.2 gives an example of HCVM data approximation over a synthetic dataset with 2 layers, $A1 \leq_h [A2, A3, A4]$. To approximate the data distribution of the dataset, an individual MEB can be used to enclose data in class $A1$.

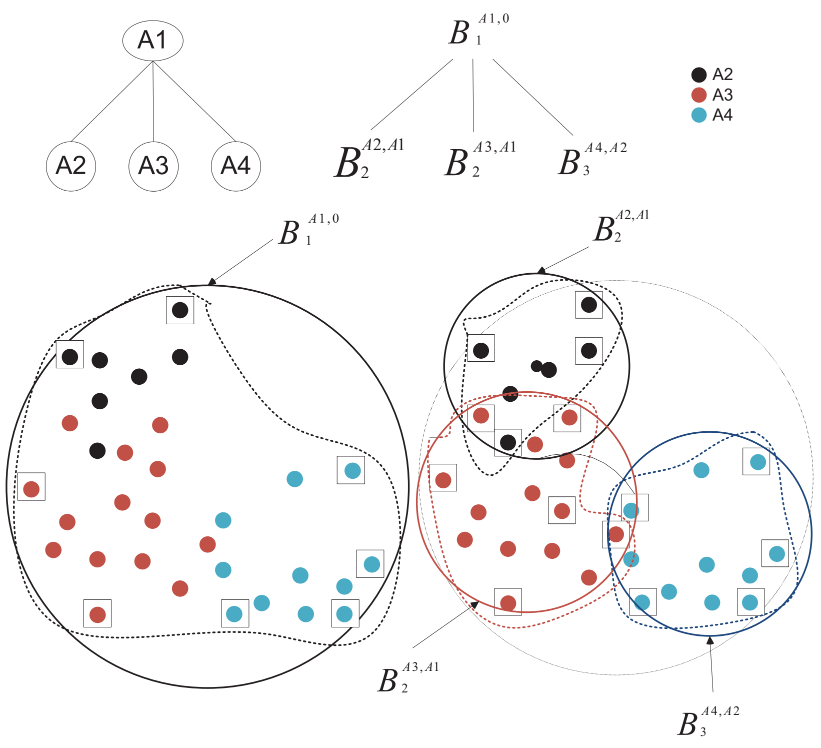


Figure 5.2: Core set data approximation comparison: boundary on the MEB core set from the parent class A1 versus boundary on the proposed HCVM ($A1 \preceq_h [A2, A3, A4]$) core set.

As shown in Fig. 5.2, the resulting boundary (identified as the dashed curve) summarizes the knowledge of MEB (represented as the solid line cycle) and the MEB core set (identified as squared data points). Base on the fact that if an instance is in a parent class, then it must belong to one of its child classes, the proposed HCVM method approximates the dataset by combining core sets from $A1$ MEB and MEBs/CVMs of $A1$'s 3 child classes $A2$, $A3$ and $A4$, and has the boundary for data approximation plotted in Fig. 5.2 right.

It clearly shows that the resulting boundary enclosed almost 2 times the space then the obtained boundary did. However, no data point exists in the extra space. Thus, the HCVM method gives a much smaller enclosing space while discarding the non-essential areas from the MEB for the parent class only.

5.2.4 *The Proposed HCVM Algorithm*

According to the theories discussed above, we have developed the following 3 algorithms for hierarchical multi-label classification. Given dataset $S = (X \times Y)$ where $X \in \Re^d$, and class label set Y is in such a hierarchy as $Y = \{Y_i | y \leq_h \forall y'\}$ whose $y, y' \in Y_i$, and $y' \neq y$. Algorithm 4 computes the MEB and core set for S. Algorithm 5 constructs HCVM where Algorithm 4 is used for modelling kernel MEBs/CVMs on subclasses $S_i = (X_i, y_i) \subset S$. Note that the constructed HCVM B is an m level hierarchical MEB, $B = \{B_1^{y^l,p^l}, B_2^{y^l,p^l}, \ldots, B_t^{y^l,p^l}\}$ where y^l, p^l denote the corresponding class label and the parent-class label respectively.

```
Input: Set of points $S \in R^d$; parameter $\epsilon = 2^{-m}$; subset $Q_0 \subset S$
Output: A $(1+\epsilon)$-approximation MEB(S); $O(1/\epsilon)$-size core set $Q$
 1: for $i \leftarrow 1$ to $m$ do
 2:   loop
 3:     $Q \leftarrow Q_{i-1}$
 4:     Compute $B_{c,r} = MEB(Q)$
 5:     if $S \subset B_{c,(1+\epsilon)r}$ then
 6:       Return $Bc, r$, $Q$
 7:     else
 8:       $p \leftarrow$ point of S maximizing distance, $\|cp\|$, from $c$
 9:     end if
10:     $Q \leftarrow Q_{i-1} \cup \{p\}$
11:   end loop
12:   $Q_i \leftarrow Q$
13: end for
```

Algorithm 4: MEB algorithm

Algorithm 6 tests the constructed HCVM for the multi-label classification of a new instance x. In order to check whether x is located in the range of the nearest MEB's radius an iterative checking process is run until one nearest MEB is found at every level of the HCVM. Associating together as the set of located MEBs/CVMs, it comes up with a complete tree path from the root to the terminal level of the HCVM. Along the tree path, the instance x is branched

```
Input: Set of points $S = (X \times Y)$ where $Y = \{Y_i | y \leq_h \forall y'\}$ whose $y, y' \in Y_i$,
    and $y' \neq y$, $\epsilon$.
Output: HCVM $B$; the $O(1/\epsilon)$-size core set $Q$.
 1: Initial $i \leftarrow 1$.
 2: Find $C_i \subseteq Y$ where $\forall y \in C_i : \neg\exists y' \leq_h y : y' \in Y, y' \notin C_i$.
 3: while $C_i \neq \phi$ do
 4:   $t_i \leftarrow |C_i|$.
 5:   for $l \leftarrow 1$ to $t_i$ do
 6:     $S = \{\varphi(x)\}$, $x = (X, y^l)$ where $y^l \in C_i$ and $Q = 0$.
 7:     $[MEB(b), CoreSet(q)] \leftarrow$ Run Algorithm 4 with input $S$,$Q$, and $\epsilon$.
 8:     if i=1 then
 9:       $p^l \leftarrow 0$.
10:     else
11:       $p^l \leftarrow p$ where $p \leq_h y^l$.
12:     end if
13:     $\boldsymbol{B_i^{p^l,y^l} \leftarrow b, Q_i^{p^l,y^l} \leftarrow q}$.
14:     $i = i + 1$.
15:   end for
16:   Find $C_i \subseteq Y$ where $\forall y \in C_i : \exists p \leq_h y : p \in C_{i-1}$.
17: end while
```

Algorithm 5: Hierarchical core vector machines training algorithm

from the root progressively to the terminal level of HCVM, meanwhile multi-labeling x with the associated labels of MEBs/CVMs at every level.

5.2.5 *Time and Space Complexities for HCVM*

Note that CVM has the computational complexity $O(\sqrt{nd^2(n+d)log(n/\epsilon)})$ (Kumar et al., 2003). Using CVM for HMC problem, the expected computational complexity is $O(\sqrt{nd^2(n+d)log(n/\epsilon)} + \sum_{i=1}^{m} \sqrt{n_i d^2(n_i+d)log(n_i/\epsilon)})$, where m is the number of child classes. This obviously is less expensive than the quadratic programming (QP) methods, such as SVM, whose training time complexity is $O(n^3)$. In this sense, a HMC problem can be modelled by MEB more efficiently, especially for problems involving large size dataset. CVM core set estimates the boundary of a given class in $O(1/\epsilon)$ (Kumar et al., 2003), thus MEB in the nature is independent on dataset size (n) and dimension (d), which indicates that MEB is deterministically robust to any class-

```
Input: Instance $x$; HCVM $B$
Output: labels of $x$ $Label$
1: Transfer $x$ to kennel $k(x,x)$ by putting it in feature map $\varphi$.
2: $Label \leftarrow 0$.
3: $index = \overbrace{1, 1, \ldots, 1}^{m}$.
4: $i \leftarrow 1$.
5: while $i \leq m$ do
6:     for $j \leftarrow index(i)$ to $n$ do
7:         Find MEB $B_i^{y^l,p^l} \in B$ where $p^l == Label(i)$, $B_i^{y^l,p^l}$'s center is $j^{th}$
           nearest to $k(x,x)$.
8:         if $k(x,x)$ inside of boundary formed by core set of $B_k$ then
9:             $Label \leftarrow Lable \cup l$.
10:            Break.
11:        else if $j == n$ then
12:            $Label \leftarrow Label - Label(i)$.
13:            $i \leftarrow i - 1$.
14:        end if
15:    end for
16:    if $Label$ is empty then
17:        Return $Label \leftarrow 0$.
18:    end if
19:    $index(i) = index(i) + 1$.
20:    $i \leftarrow i + 1$.
21: end while
```

Algorithm 6: Hierarchical core vector machines testing algorithm

imbalanced and/or high dimensional datasets. In HCVM, maximum m MEBs/CVMs are used for HMC problem solving where m represents the number of child classes. Thus, the $O((m+1)/\epsilon)$ computational complexity of HCVM enable solving of any HMC problem more efficiently, meanwhile immune to the class-imbalanced and/or high dimensional difficulty.

5.3 VALIDITY EXAMINATION ON SYNTHETIC DATASET

For testing the capability of the proposed HCVM for multi-label, in particular for hierarchical multi-label classification, we examined the developed HCVM on a synthetic dataset.

The synthetic dataset for HCVM examination is a mixture of several 2D Gaussian labelled with a 2-level multi-label hierarchy, $\mathcal{H}$ ={class 1: {class(1, 1),class(1,2),class(1,3) }; class 2:{class(2,1), class(2,2), class(2,3)}}, in which the distribution of class 1 data is shown in Fig 5.3.

5.3.1 *Results and Discussion*

Using MEB, class 1 can be approximated by constructing a single independent MEB as Fig. 5.4a, and the obtained core set, as support vectors for building class 1 de scripting plane, is presented in Fig. 5.4c. Alternatively using HCVM, the MEB can be constructed instead as Fig. 5.4b for both the parent and the child classes as class 1 in fact has 3 child classes. Thus, a 2-level HMC problem is modelled by HCVM since a set of correlated MEBs/CVMs (i.e a hierarchical MEB). Fig. 5.4d gives accordingly the obtained core sets from 4 correlated MEBs/CVMs. As seen, it is apparently that the core set in Fig. 5.4d approximates class 1 data more accurate than the core set in Fig. 5.4c. This demonstrates that the developed HCVM derives important additional discriminant information from the 2-level label hierarchy.

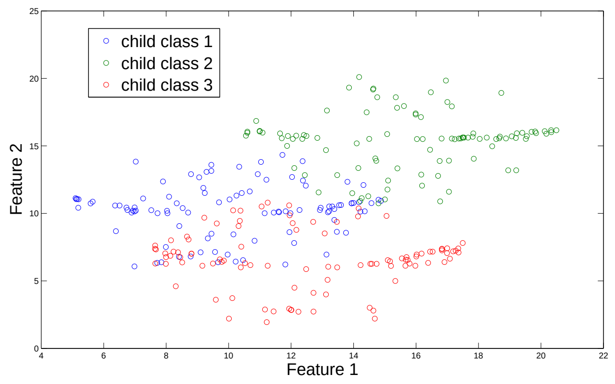
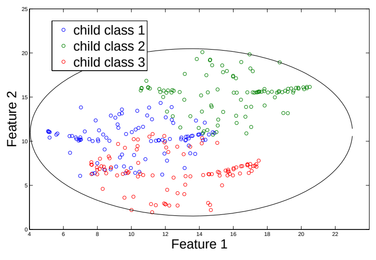
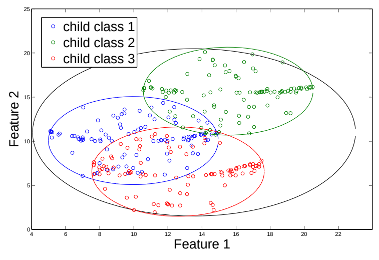


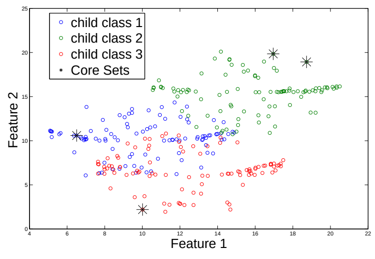
Figure 5.3: Synthetic data for HMC, where data points of different classes are represented as different colors.



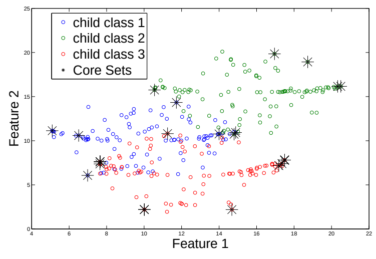
(a) a MEB for $class1$



(b) 4 MEBs/CVMs for $class\,1$, $class(1,1)$, $class(1,2)$ and $class(1,3)$



(c) the distribution of core set from (a)



(d) the distribution of core sets from (b)

Figure 5.4: An example of using MEB for HMC, where * represents a MEB core set point. Note that this figure is best seen in color.

The results also shows that, HCVM is a local modelling method which learns the local information from sub-classes. And local modelling beat global modelling by obtaining a much more accurate class boundary.

5.4 SUMMARY

In this chapter, we studied HMC problem aiming to multiply labelling network intrusions for Internet security. The HMC differs from typical classification in two aspects: (1) an instance belongs often to more than one class simultaneously; (2) every instance from any class belongs automatically to its ancestor classes, as classes in HMC are in hierarchical tree structure.

By aggregating MEBs/CVMs, we proposed a novel HCVM model in which MEBs/CVMs separate, encompass and overlap with each other representing an organized multi-label tree structure. Moreover, we have modified the original MEB algorithm using the Gaussian kernel method, enabling MEBs/CVMs to be applicable for kernel computing in a higher dimensional space.

In general, the proposed HCVM method has the following desirable properties. First, HCVM models a difficult hierarchical multi-label problem as a simple MEB association analysis. Second, MEB excludes the existing sparseness of data, which enables HCVM to approximate data more accurately. However, the proposed method inhibits a limitation that the classification performance of the terminal classes might be inferior to that of non-terminal classes since the boundaries for the terminal classes in a multi-label hierarchy are not optimized.

The next chapter introduces a on-line local modelling method which is capable of not only hierarchical structured stream datasets but normal datasets.

Chapter 6

DYNAMIC EVOLVING CORE VECTOR MACHINES - A FAST CONNECTIONIST-BASED KERNEL LEARNING SYSTEMS

This chapter introduces a new type of kernel-based learning algorithm, denoted as dynamic evolving core vector machines (DE-CVM). This method is developed for adaptive on-line stream data learning. Similar to DENFIS (Kasabov & Song, 2002), DE-CVM evolve through incremental, hybrid (supervised/unsupervised) learning, and accommodate new input data, including new features, new classes, etc., through local element tuning.

This chapter is structured as follows: Section 6.1 explains why DE-CVM is essential for stream data learning. Section 6.2 gives the motivations for the proposed DE-CVM. The algorithm is presented in section 6.3. Section 6.4 covers experimentation and algorithm evaluation. Lastly, in section 6.5 we draw our conclusion.

6.1 WHY EVOLVING SYSTEMS ARE ESSENTIAL FOR STREAM DATA MINING? - ANSWER FOR QUESTIONS 3

Kasabov and Song (2002) maintained that, the complexity and dynamics of real-world problems, especially in engineering and manufacturing, require sophisticated methods and tools for building on-line, adaptive intelligent systems (ISs). Such systems should be able to grow as they operate, to update their knowledge and refine the model through interaction with the environment.

Stream data mining focuses on real-world applications, such as network intrusion detection, texture categorisation and image recognition. These applications need machine learning methods that have the following characteristics: fast learning, on-line incremental adaptive learning, open structure organisation, memorising information, active interaction, knowledge acquisition and self-improvement, and spatial and temporal learning (Kasabov & Song, 2002).

We adopt evolving connectionist systems (ECOSs) (Kasabov, 1998) for stream data mining because ECOSs evolve their structure and functionality from a continuous input data stream in an adaptive, life-long, modular way. Another important reason is that ECOSs employ local learning as they create connectionist-based modules and connect them, if that is required according to the input data distribution and the system's performance at a certain time moment.

6.2 MOTIVATION OF DE-CVM

The minimum enclosing ball (MEB) problem is to compute a ball of minimum radius enclosing a given set of objects (points, balls, etc.) in $\mathcal{R}^d$. It has been widely implemented for clustering applications, such as spatial hierarchies (Hubbard, 1996), and support vector clustering (Ben-Hur, Horn, Siegelmann, & Vapnik, 2002); classification applications, such as area gap tolerant classifiers (Burges, 1998b), and core vector machine (CVM) (Tsang et al., 2005); as well as approximation applications, such as fast farthest neighbour query approximation(Goel, Indyk, & Varadarajan, 2001), and 1-cylinder problem approximation (Bādoiu et al., 2002).

Classic CVM for classification computes a $(1+\varepsilon)$-approximation (Bādoiu et al., 2002) for a minimum radius ball learning, and extracts those data points located at the outer area of a CVM for classification modelling. The set of those extracted data points characterise the entire given dataset, and thus are called core vector set or core set. For classification modelling, CVM can be used to approximate each class data distribution, so that one class can be distinguished from another by core set computing.

However, in practice such classic MEB has the following difficulties: (1) CVM often encloses sparseness together with data. To enclose an isolated outlier point, a huge MEB is required, which makes the MEB include actually more sparseness than the data occupation; (2) MEB is keen on enclosing data, and thus disables the detection of any outliers despite the outliers producing the sparseness of MEB.

To mitigate the above problems, we propose a novel dynamic evolving CVM (DE-CVM) approach to learning the core set in a group manner. DE-CVM sets MEB/CVM in different data distribution area, reducing the sparseness in CVM by decomposing data space based on data distribution density, discriminating core vectors on class interaction hyperplanes, and enabling outlier detection to decrease the effects of noise.

To have a better and clear presentation of the proposed one-pass MEB, Table 6.1 presents the symbols that will be used in this chapter.

Notation	**Descriptions**
$\mathbf{X}$	data matrix
$\mathbf{X_i}$	data matrix for i-th class
n	number of training data instances
n_i	number of instances for i-th class
l	number of classes
$B_{\mathbf{X}}$	minimum enclosing ball/balls (MEB) based on data set $\mathbf{X}$
$\mathbf{c}_{i,j}$	i-th class j-th MEB center/centers
$r_{i,j}$	i-th class j-th MEB radius
k_i	total number of MEBs for i-th class
$\mathbf{Q}_{i,j}$	i-th class j-th MEB core set
$\mathbf{\Omega}_{MEB}$	classic MEB learning model
$\mathbf{\Omega}_{cMEBs}$	cross-MEBs learning model
ϵ	MEB approximation value

Table 6.1: NOTATION

6.3 THE PROPOSED DE-CVM LEARNING

In terms of complex data distribution, a large MEB/CVM follows the sparseness of the ball, thus it is often unable to approximate a data distribution accurately. A single MEB/CVM for one class data is likely to enclose wrongly almost all data points from another class, especially when data is zonally distributed. As a solution, a number of smaller MEBs/CVMs are able to drill into the details of any data distribution, apparently allowing a more accurate approximation.

Motivated by this, the above *kernel* MEB is renovated for group manner MEB/CVM computing (DE-CVM). Instead of addressing a whole class data $\mathbf{X}_i$ with one MEB/CVM $B_{\mathbf{Q}_i}$, DE-CVM models class data using with a set of MEBs $B_{\mathbf{Q}_i} = \cup_{j=1}^{k} B_{\mathbf{Q}_{i,j}^u}$, where k is the number of MEBs/CVMs. Consider that MEB/CVM learning is an iterative learning process, we represent an individual *kernel* MEB/CVM here as $B_{\mathbf{Q}_{i,j}^u} = \{\mathbf{c}_{i,j}^u, (1+\varepsilon)r_{i,j}^u, \varphi(\mathbf{Q}_{i,j}^u)\}$ with j as the index of MEB/CVM, and u as the iteration number of MEB/CVM updating. In this way, given $\mathbf{X} = \cup_{i=1}^{l} \mathbf{X}_i$ as the training dataset, the proposed DE-CVM learning is described below.

Similar to CVM, we initialise one MEB/CVM (i.e. $B_{\mathbf{Q}_{i,k}^0}, \ k = 1$) on one class data $\mathbf{X}_i$ at very beginning. Here, we initialise the core set as $\mathbf{Q}_{i,1}^0 = \{\varphi(\mathbf{x}_a), \varphi(\mathbf{x}_b)\}$, where $\mathbf{x}_a$ is the furthest data point to a random $\mathbf{x} \in \mathbf{X}_i$ and $\mathbf{x}_b$ is calculated as:

$$\mathbf{x}_b = \mathbf{x}_a + \frac{\mathbf{x}_a - \arg\max_{\mathbf{x}\in\mathbf{X}} ||\mathbf{x}_a - \mathbf{x}||}{\lambda}, \quad \lambda > 1. \tag{6.1}$$

Then, we obtain $B_{\mathbf{Q}_{i,1}^0} = \{\mathbf{c}_{i,1}^0, (1+\varepsilon)r_{i,1}^0, \mathbf{Q}_{i,1}^0\}$ using CVM optimisation function.

Theorem 6.3.1. *The first initialised MEB/CVM's radius* $r_{i,1}^0 \approx \frac{\Delta}{2\lambda}$, *where* Δ *denotes the diameter of* $\mathbf{X}_i$.

Proof. Science $\mathbf{Q}_{i,1}^0 = \varphi(\mathbf{x}_a, \mathbf{x}_b)$, $\mathbf{x}_a$ is the furthest point to $\mathbf{x}_1$. Clearly, the distance from $\mathbf{x}_a$ to its furthest data point approximates to δ. Then, $\mathbf{x}_b$ locates about Δ/λ away from $\mathbf{x}_a$. According to the definition, $r_{i,1}^0$ is the radius of $B_{\mathbf{Q}_{i,1}^0}$, we have $r_{i,1}^0 \approx \frac{\Delta}{2\lambda}$. □

If there is $\mathbf{x} \in \mathbf{X}_i$ not contained in any MEB $B_{\mathbf{Q}_{i,j}^{u-1}}$, $(j = \{1, 2, \ldots, k\})$. It results in one of the following 3 DE-CVM updating cases.

In the first case, if MEB $B_{\mathbf{Q}_{i,j}^{u}}$ over the existed core set $\mathbf{Q}_{i,j}^{u-1}$ with $\mathbf{x}$ (i.e. $\mathbf{Q}_{i,j}^{u} = \mathbf{Q}_{i,j}^{u-1} \cup \varphi(\mathbf{x})$) has radius $r_{i,j}^{u}$, which is less or equal to the upper bound value as $1 + \frac{\varepsilon^2 \times \eta}{16}$, $(\eta > 1)$ times of the existed radius $r_{i,j}^{u-1}$. The existing MEB $B_{\mathbf{Q}_{i,j}^{u-1}}$ is updated by replacing with $B_{\mathbf{Q}_{i,j}^{u}}$. Note that, $\mathbf{c}_{i,j}^{u-1}$ is the closest MEB centre to $\mathbf{x}$.

Theorem 6.3.2. *Given an upper bound of radius increment as $r_{i,j}^{u} \le (1 + \frac{\varepsilon^2 \times \eta}{16}) r_{i,j}^{u-1}$, the DE-CVM expansion is between $32\lambda/\varepsilon^2$ to $128\lambda\delta/\varepsilon^2$ times.*

Proof. Since $r_{i,j}^{0} \ge \Delta/2\lambda$, and each step we increase the radius by at least $(\Delta/4)\varepsilon^2/16 = \Delta\varepsilon/64$, it follows that we cannot encounter this case more than $64/\varepsilon$ times, as δ is an upper bound of the radius of the minimum enclosing ball of $\mathbf{X}_i$. $\square$

In the second case, if radius $r_{i,j}^{u}$ is greater than the upper bound value $(1 + \frac{\varepsilon^2 \times \eta}{16}) r_{i,j}^{u-1}$, a new fragmentary core set $\mathbf{Q}_{i,k+1}^{0} = \{\varphi(\mathbf{x})\}$ is created as a completed core set which has at least 2 vectors.

In the third case, if the distance from $\mathbf{x}$ to the closest fragmentary core set $\mathbf{Q}_{i,j}^{0}$ is less than $\frac{\Delta}{\lambda}$, we add $\varphi(\mathbf{x})$ into the fragmentary core set $\mathbf{Q}_{i,j}^{0}$ as $\mathbf{Q}_{i,j}^{0} = \{\mathbf{Q}_{i,j}^{0}, \varphi(\mathbf{x})\}$, in this way, the fragmentary core set $\mathbf{Q}_{i,j}^{0}$ becomes completed. In addition, a new MEB $B_{\mathbf{Q}_{i,k}^{0}} = \{\mathbf{c}_{i,k}^{0}, (1+\varepsilon) r_{i,k}^{0}, \mathbf{Q}_{i,k}^{0}\}$, $(k = k+1)$ is created using CVM optimisation function (Y. Chen, Pang, Kasabov, Ban, & Kadobayashi, 2009). The threshold $\frac{\Delta}{\lambda}$ is also considered as the outliers threshold, as if a single data point away from the rest data farther than the threshold, this data point is treated as an outlier by DE-CVM.

Theorem 6.3.3. *The total number of DE-CVM k is equal to approximately $\lambda/2$.*

Proof. Science we guarantee that the radius of each firstborn DE-CVM $r_{i,j}^{0}$, $(j = \{1, 2, \ldots, k\})$ is less than $\frac{1}{2\lambda}$ diameter of the given data $\mathbf{X}_i$, the total number of the DE-CVM k approximates to $\lambda/2$. $\square$

The DE-CVM updating is terminated once $\mathbf{X}_i \subset \cup_{j=1}^{k} B_{\mathbf{Q}_{i,j}^{k}}$. For the overall dataset $\mathbf{X} = \cup_{i=1}^{l} \mathbf{X}_i$, let k_i be the number of MEBs/CVMs of i-th class, then we factorise the core sets by abandoning these core vectors contained in just one MEB/CVM. As a result, we have the DE-CVM model Ω_{DE-CVM} as the set of MEBs/CVMs that constructed by the above 3 DE-CVM updating cases.

$$\Omega_{DE-CVM} = \begin{pmatrix} \cup_{j=1}^{k_1}\mathbf{c}_{1,j} & \cup_{j=1}^{k_1}(1+\varepsilon)r_{1,j} & \cup_{j=1}^{k_1}\mathbf{Q}_{1,j} \\ \cup_{j=1}^{k_2}\mathbf{c}_{2,j} & \cup_{j=1}^{k_2}(1+\varepsilon)r_{2,j} & \cup_{j=1}^{k_2}\mathbf{Q}_{2,j} \\ \vdots & \vdots & \vdots \\ \cup_{j=1}^{k_l}\mathbf{c}_{l,j} & \cup_{j=1}^{k_l}(1+\varepsilon)r_{l,j} & \cup_{j=1}^{k_l}\mathbf{Q}_{l,j} \end{pmatrix}, \tag{6.2}$$

and summarise the computation of DE-CVM learning as Algorithm 7.

6.4 EXPERIMENTS AND DISCUSSIONS

In this section, we describe two experiments where we used DE-CVM for banana data verification and for benchmark UCI data classification.

6.4.1 *Class Factorisation Ability Test*

To highlight the class factorisation ability of the proposed DE-CVM, we cluster an artificial two-dimensional banana dataset (see Fig. 6.1a) containing 200 data points in a banana shaped distribution using both DE-CVM and classic CVM. Fig. 6.1b illustrates the cluster boundary formed by 70 classic CVM core vectors over the banana data set, where star symbols represent the core vectors and a solid line represents the boundaries. From 6.1b, it can be seen that all the MEB/CVM core vectors are positioned around the outer area of data distribution. As a result, data points located around the inner curvature of the banana distribution are not encompassed by the cluster boundary.

To overcome the difficulty of forming the inner curvature area of the banana data cluster, we employ the proposed DE-CVM method by inputting a new cluster dataset which contains 80 data points located close to the inner curvature area of the banana dataset (see Fig. 6.1c, the new cluster is represented by squares). Then we perform DE-CVM on both datasets by configuring the banana dataset as a positive class while the new cluster is configured as negative class. Fig. 6.1c illustrates the six DE-CVM core vectors refining the cluster boundary calculated by the classic CVM, thereby accurately depicting the in-

```
Input: Set of points $\mathbf{X} \in \mathbb{R}^d$; parameter $\varepsilon$, $\lambda$, and $\eta$
Output: A DE-CVM learning model $\Omega_{DE-CVM}$
 1: for each $\mathbf{X}_i \subset \mathbf{X}$ do
 2:     Initialise $k \leftarrow 1,\ T = \emptyset$, and $\Delta$ which is the diameter of $\mathbf{x}$
 3:     Initialise $\mathbf{Q}^0_{i,j}$ by equation 6.1
 4:     Computer the initial MEB $B^0_{i,1}$ and its radius $r^0_{i,1}$ and center $\mathbf{c}^0_{i,1}$ using
        equation ?? on $\mathbf{Q}^0_{i,1}$
 5:     for each $\mathbf{x} \in \mathbf{X}_i$ do
 6:         if $\mathbf{x} \notin \cup^k_{j=1} B_{i,j}$ then
 7:             Find $\mathbf{t} \leftarrow \arg\min_{\mathbf{t}\in\mathbf{T}} ||\mathbf{x} - \mathbf{t}||$
 8:             if $||\mathbf{x} - \mathbf{t}|| < \frac{\Delta}{2\lambda}$ then
 9:                 Remove $\mathbf{t}$ from $\mathbf{T}$
10:                 $k \leftarrow k + 1$
11:                 $\mathbf{Q}^0_{i,k} = \{\varphi(\mathbf{t}), \varphi(\mathbf{x})\}$
12:                 Computer a new MEB $B^0_{i,k}$ and its radius $r^0_{i,k}$ and center $\mathbf{c}^0_{i,k}$
                    using optimisation function on $\mathbf{Q}^0_{i,k}$
13:             else
14:                 Find $\mathbf{c}^u_{i,j} \leftarrow \arg\min_{j\in\{1:k\}} ||\mathbf{x} - \mathbf{c}^u_{i,j}||$
15:                 $\mathbf{Q}^{u+1}_{i,j} \leftarrow \{\mathbf{Q}^u_{i,j}, \varphi(\mathbf{x})\}$
16:                 Update MEB $B^{u+1}_{i,j}$ and its radius $r^{u+1}_{i,j}$ and center $\mathbf{c}^{u+1}_{i,j}$ using
                    equation ?? on $\mathbf{Q}^{u+1}_{i,j}$
17:                 if $r^{u+1}_{i,j} > r^u_{i,j} \times (1 + \frac{\varepsilon^2\times\eta}{16})$ then
18:                     $\mathbf{T} \leftarrow \mathbf{T} \cup \mathbf{x}$
19:                     Undo this update
20:                 end if
21:             end if
22:         end if
23:     end for
24:     $\Omega_{DE-CVM} \leftarrow \Omega_{DE-CVM} \bigcup \{\cup^k_{j=1}\mathbf{c}_{i,j}, \cup^k_{j=1}(1+\varepsilon)r_{i,j}, \cup^k_{j=1}\mathbf{Q}_{i,j}\}$
25: end for
26: Return $\Omega_{DE-CVM}$
```

Algorithm 7: DE-CVM Algorithm

ner curvature of banana dataset. Furthermore, we increase the new cluster data by adding another 50 data points that surround the left hand side of the banana dataset, then implement DE-CVM on these two datasets. Fig. 6.1d illustrates an approximate hyperplane, which is formed by 22 DE-CVM core vectors, in order to discriminate the banana data from the new cluster data. From Fig. 6.1d, it can be inferred that DE-CVM is able to factorize the class characteristics of the banana dataset by creating a more accurate boundary using fewer core vectors than MEB/CVM. Similarly, by adding new cluster data around the banana cluster, using the same process, we are able to reduce the core vector size due to class factorization.

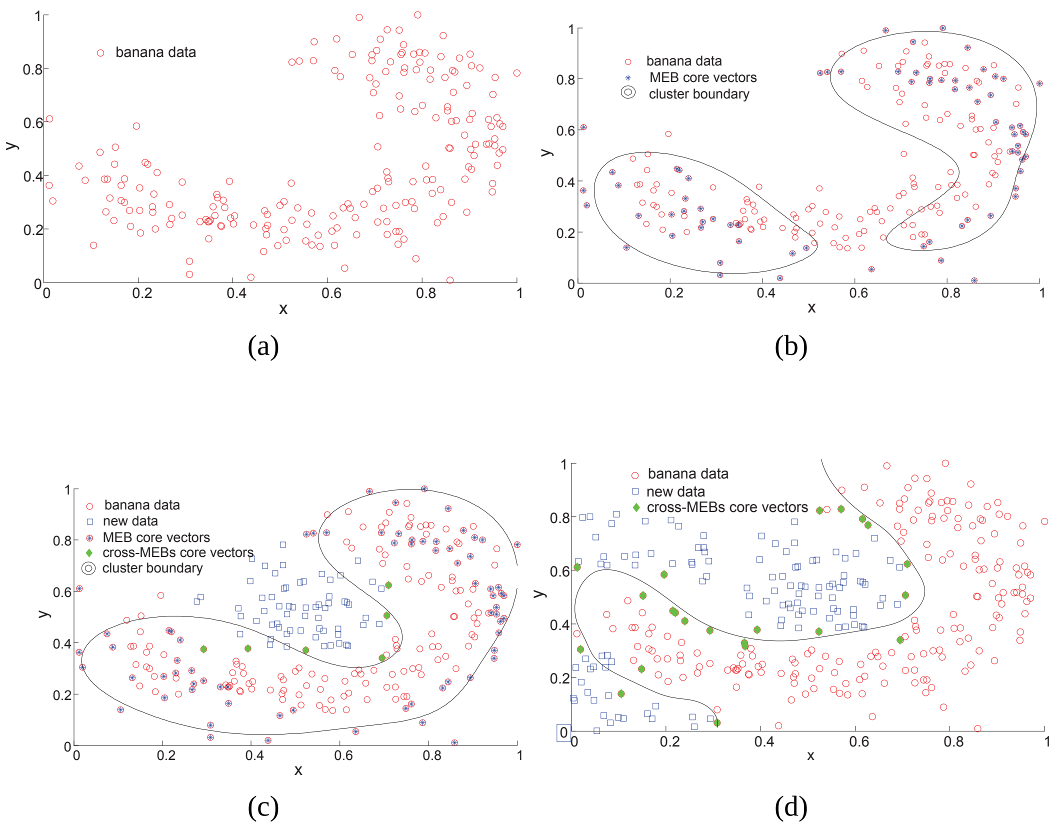


Figure 6.1: Banana data set clustering (a) plot 200 banana shaped data points, (b) plot the cluster boundary in black lines formed by 70 classic MEB core vectors, (c) plot the new cluster data in the bend area of banana data and banana data cluster boundary formed by the classic MEB core vectors and 6 DE-CVM core vectors, (d) plot the new data surround the left hand side of the banana data and the DE-CVM core vectors.

6.4.2 *Classification Accuracy Comparison*

To evaluate the classification performance of DE-CVM, we compared DE-CVM with five other classification methods (i.e. KNN, Bayes, MLP, SVM and

CVM) on four two-class and four multi-class benchmark datasets which are summarised in Table 6.2.

Type	Dataset	No. of Features	No. of Classes	Size
two-class	heart	13	2	270
	liver-disorder	6	2	345
	ionosphere	34	2	351
	breast-cancer	10	2	683
multi-class	iris	4	3	150
	wine	13	3	178
	vehicle	18	4	846
	vowel	10	11	990

Table 6.2: Summarisation of data sets characteristics

For each dataset, we conducted K-fold cross-validation which randomly partitioned the original data into K subsamples. Of the K subsamples, a single subsample was retained as the validation data for testing the model, and the remaining $K-1$ subsamples are used as training data. The cross-validation process was repeated K times, with each of the K subsamples used exactly once as validation data. Then we can obtained the mean value of the K results from the folds. In our case, we set $K = 10$ as 10-fold cross-validation which is commonly used.

We also compared the class factorization ability of DE-CVM, CVM, and SVM as these three kernel-based methods extract support/core vectors which are the discriminative data points for each class. We consider a method to have strong class factorization ability, if it is able to learn most discriminative class characteristics. In other words, this is a method that is capable of extracting fewest support/core vectors and giving highest classification accuracy along with strongest class factorization ability.

Table 6.3 presents the comparison results of the classification accuracies and the number of support/core vectors among six classifiers. The three columns next to the name of datasets record classification accuracies for three conventional methods, such as KNN, Bayes, and MLP where KNN and MLP achieve the highest accuracy on one dataset only. The remaining columns in Table 6.3 summarised the classification accuracies and number of support/core vectors extracted by those three kernel-based classification methods mentioned above. It can be observed that the three selected kernel-based methods achieve higher classification accuracy than those three conventional methods in general. DE-CVM undoubtedly shows that it has the strongest class factorisation ability among these three kernel base methods by achieving the highest classification accuracy but extracting fewest core vectors on five out of eight datasets. The results indicate that DE-CVM can be considered the perfect method for most of the selected datasets.

Classifier	KNN	Bayes	MLP	SVM		CVM		DE-CVM	
Dataset	acc	acc	acc	acc	SVs	acc	CVs	acc	CVs
heart	82.59%	75.65%	83.33%	83.19%	135	83.67%	101	**84.11%**	**84**
liver-disorder	64.92%	63.17%	**71.28%**	64.92%	170	59.13%	171	65.13%	**138**
ionosphere	64.11%	65.32%	89.16%	92.06%	135	**92.73%**	101	92.72%	**77**
breast-cancer	96.63%	95.97%	96.65%	**97.07%**	340	95.43%	176	**97.07%**	**136**
iris	94.67%	90.35%	95.97%	96.00%	75	**96.66%**	66	**96.66%**	**63**
wine	**97.16%**	71.31%	93.91%	74.85%	85	73.76%	86	83.12%	**82**
vehicle	71.04%	66.24%	77.78%	71.61%	420	71.61%	**72**	**78.13%**	92
vowel	96.89%	91.37%	98.08%	97.47%	495	97.45%	**110**	**98.56%**	124

Table 6.3: Classification accuracy(acc) and number of support vectors(SVs)/core vectors(CVs) comparison on eight selected data sets

6.5 SUMMARY

In this chapter, a novel DE-CVM is proposed that learns the characteristics of a class (*i.e.* core vectors) in a group manner. DE-CVM factorizes class characteristics by reducing the sparseness area, discriminating core vectors on class interaction hyperplanes, and enabling outliers detection. DE-CVM is evaluated by conducting a comparison of classification accuracy and computational cost on eight data sets with five conventional classification methods including original CVM. DE-CVM obtains the highest classification accuracy for five out of eight datasets. Compared to CVM, DE-CVM wins on all the data sets except ionosphere by utilising fewer core vectors hence rendering it computationally more efficient than CVM.

In the next chapter, we will introduce a novel string kernels classification method for string format stream data mining.

Chapter 7

STRING FORMAT STREAM DATA MODELLING TECHNIQUES

In this chapter, we investigate string format stream data modeling techniques. String classification is becoming a major area of stream data learning. This is because of the explosive growth of Internet users. Network intrusions such as SPAM emails, malicious software which formed as sequences of string data are also increasing (Shawe-Taylor & Cristianini, 2004).

Recently, string kernels-based support vector machines have shown competitive performance in tasks such as text classification and protein homology detection (Lodhi, Saunders, Shawe-Taylor, Cristianini, & Watkins, 2002). We proposed two novel string kernels learning methods: Meta Learning String Kernel SVMs and String Kernel MEBs to improve the effectiveness of the traditional string kernels SVMs.

Section 7.1 gives a brief review of string kernels and string kernels SVMs. Meta Learning String Kernel SVMs and String Kernel MEBs will be introduced in Section 7.2 and 7.3, respectively. Section 7.4 gives the conclusion for this chapter.

7.1 STRING KERNEL SVMS

Due to SVMs success in numerical pattern recognition with the help of kernel functionality, research focused on using SVMs with string kernels for string classification tasks. As Shawe-Taylor and Cristianini (2004) and Cortes, Haffner, and Mohri (2004) explain, apart from popularly used string kernels like Bag-of-Words and Edit Distance, one can also use kernels like n-gram or subsequence, which sum up substrings or subsequence in a document, as string kernels for a string classification task. Results from previous experiments (Shawe-Taylor & Cristianini, 2004; Cortes et al., 2004) have shown that SVM with string kernel functionality is able to recognise string patterns much more efficiently than other methods due to its ability to handle high dimensional data like string data without a decrease in performance decrease. This has made SVMs with string kernel functionality an ideal solution for DNA prediction, document classification, language recognition, image recognition and network anomaly detection scenarios. Even though string kernels are a subset of the kernel domain, they have their own properties and computational complexities associated with them. The following subsections explain about some widely used string kernels.

7.1.1 *String Kernel methods*

Further below we review the definition of mismatch kernels (Leslie, Eskin, Weston, & Noble, 2002) and present three new families of string kernels: Bag-of-Words kernel, Gap-Weighted Subsequences kernel, n-gram kernel and Levenshtein (or edit) distance kernel. In each case, the kernel is defined via an explicit feature map from the space of all finite sequences from an alphabet S to a vector space indexed by the set of k-length subsequences from S. These models have been used in the computational biology literature in other contexts, in particular for sequence pattern discovery in DNA and protein sequences (Sagot, 1998) and probabilistic models for sequence evolution (Henikoff, 1992; Altschul, Gish, Miller, Myers, & Lipman, 1990).

Bag-of-Words Kernel

In document categorisation, a collection of *documents* is called a ‘corpus’, which consists of a set of predefined *terms* and is identified as a *dictionary*. A term or synonymously a *word* in the dictionary is any sequence of letters separated by punctuation or spaces. On the other hand, a *bag,* is defined as a set that allows repeated items. This definition of bag helps one to view a document as a *bag of terms* or *bag of words* (BOW). This allows a document to be presented as a vector where each dimension is associated with a term in the dictionary. This representation (ϕ) is given as:

$$\phi(d) = (tf(t_1, d), tf(t_2, d), ..., tf(t_n, d)) \in \mathbb{R}^n,$$

here $tf(t_i, d)$ is the frequency of the t_i^{th} *term* in document d. Also, n is the space dimensionality and the size of the dictionary (Shawe-Taylor & Cristianini, 2004). Now, one can define a function k in this document space to compare the similarity between two documents d_1 and d_2:

$$\kappa_{bow}(d_1, d_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f(t_i, t_j)^j, \tag{7.1}$$

where, d_1 has n_1 *terms* and d_2 has n_2 *terms*. Also, f is defined as:

$$f(t_x, t_y) = \begin{cases} \lambda_B^2 & \textit{if } t_x = t_y, \lambda_B \in (0, 1) \\ 0 & \textit{otherwise} \end{cases} \tag{7.2}$$

where t_x and t_y are two *terms*.

N-gram Kernel

In n-gram kernel, a string s is defined from alphabet Σ of $|\Sigma|$ symbols, and is presented in a feature space F, where each dimension is a string (Shawe-Taylor & Cristianini, 2004; Lodhi et al., 2002). Also, Σ^* represents the set of all strings and Σ^n represents the string set of length n. Furthermore, ss' represents the concatenation of strings s and s'. Now, the *substrings*: u, v_1, v_2 of string s, are defined such that:

$$s = v_1 u v_2,$$

where, if $v_1 = \varepsilon$ (ε is the empty string of 0 length) then, u is called to be the *prefix* of s and if $v_2 = \varepsilon$, then u is called to be the *suffix* of s. Now, a feature map φ is defined in feature space F, with below embedding,

$$\phi_u^n(s) = |\{(v_1, v_2) : s = v_1 u v_2\}|, u \in \Sigma^n. \tag{7.3}$$

The associated kernel is defined as:

$$\kappa_{ngram}(s,t) = \langle \phi^n(s), \phi^n(t) \rangle = \sum_{u \in \Sigma^n} \phi^n_u(s)\phi^n_u(t). \tag{7.4}$$

Also, the computational complexity of ***n-gram*** kernel is written as $O(n|s|t|)$ (Shawe-Taylor & Cristianini, 2004).

Gap-Weighted Subsequences Kernel

Let $s, t \in \sum^*$ be two strings from a finite alphabet $\sum$. Denote the lengths of the strings by $m = |s|$, $n = |t|$ and assume without loss of generality that $n \leq m$. Given a gap penalty λ, and an integer p, the gap-weighted subsequences (GWS) kernel is

$$\kappa_{GWS}(s,t) = \sum_{u \in \sum^p} \phi^p_u(s)\phi^p_u(t), \tag{7.5}$$

where $\phi^p_u(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{l(\mathbf{i})}, u \in \sum^p$ is the embedding to the feature space of subsequences of length p.

Levenshtein (or edit) distance kernel

The levenshtein (or edit) distance associates to the difference between two strings. The difference refers to, number of insertions, substitutions and deletions required to transform string s to string t. Assume that string s is of length n and string t is of length m. For the string s let $s(i)$ be its i^{th} character. Also for two characters a and b, we define r as follows:

$$r(a,b) = \begin{cases} 0 & \text{if a=b} \\ 1 & \text{otherwise} \end{cases} \tag{7.6}$$

An $(n+1)(m+1)$ array d, furnishes the levenshtein distance $L(s,t)$ between s and t via its $(n+1),(m+1)^{th}$ item. The calculation of $d(i,j)$ is done in a recursive manner by initially setting up $d(i,0)$ and $d(0,j)$ as

$$d(i,j) = \min(d(i-1,j)+1, d(i,j-1)+1, d(i-1,j-1)+r(s(i),t(j))). \tag{7.7}$$

In this work, we used $e^{\lambda d(i,j)}$, rather than $d(i,j)$ for improved results.

7.1.2 *Parameter Optimisation for String Kernels*

String kernels also have different parameters like λ_L in edit-distance kernel, λ_B in bag-of-words kernel, sub-string size in n-gram kernel and subsequence size in fixed length subsequence kernel, which derives different kernel matrices. Hence, the parameter optimisation problem applies to string kernels as well. Adding to the complexity, the computation cost for strings can be quite expensive compared to numeric data, where most of the string kernels require an internal function to map strings to numbers (i.e. function r in edit-distance and function f in bag-of-words kernel that is explained in section 7.1.1). Also, if one observes the experimental results mentioned in (Rieck & Laskov, 2007), (Lodhi et al., 2002), (Sharma, Girolami, & Sventek, 2007) and (S. Sonnenburg, 2006) even the same string kernel requires different parameter combinations on different string datasets to yield good classification accuracies. This brings about the point that string dataset characteristics also need to be included in a string kernel SVM optimisation method.

7.2 A NOVEL METHOD OF META LEARNING FOR STRING CATEGORISATION

7.2.1 *Meta Learning for String Categorisation - Answer for Question 4*

This section discusses the motivation to use meta learning for string classification. It defines and explains meta learning, then elucidates the process of meta learning and its applications, followed by an explanation on meta-features. Then this chapter discusses the application of meta learning for text categorisation using a set of text meta-features, which are calculated from text data. After that it clearly demarcates the difference between text and string data, clarifying their ability to use some of the text meta-features for string classification at the end.

Meta Learning

As Giraud-Carrier, Vilalta, and Brazdil (2004) explain, meta learning is the process of acquiring and exploiting meta-knowledge through *re-learning* from meta-features. Re-learning, which is to maintain the learning algorithm unchanged or to modify it, helps the learning system to profit from repetitive use of similar tasks. It can be applied on a single learning system to optimise parameters, or on a set of algorithms to select the best algorithm for a given classification task (Vilalta, Giraud-Carrier, Brazdil, & Soares, 2004; Furdík, Paralič, & Tutoky, 2008; Brazdil, Soares, & Da Costa, 2003; Giraud-Carrier et al., 2004). The process of re-learning requires a set of domain specific characteristics so called 'meta-features' to evaluate the performance of an algorithm or algorithms (Brazdil et al., 2003). In practice, meta learning is used to select the best algorithm for a text classification (Lam & Lai, 2001; Furdík et al., 2008), predict optimum parameters for kernels in SVMs (Soares, Brazdil, & Kuba, 2004), and to optimise neural networks (Kord et al., 2010).

Apart from parameter optimisation, meta learning on a single learning system is used to evolve the architecture of the learning system via experience, such as evolving a decision tree using past experience (Brazdil, Giraud-Carrier, Soares, & Vilalta, 2008) or to evolve a neural network considering past topology parameters (Kord et al., 2010). On the other hand, meta learning on a set of algorithms is used in situations such as algorithm ranking (Brazdil et al., 2003) and algorithm identification in text categorisation (Furdík et al., 2008). Sound meta features that effectively describe domain characteristics are required in both these systems (single learning and systems with multiple algorithms) (Brazdil et al., 2008; Giraud-Carrier et al., 2004).

Generally, there are three types of meta-features. Firstly, *simple statistical and information-theoretic meta-features* are calculated from the dataset, such as number of classes, number of features, degree of correlation between features, and average class entropy (Brazdil et al., 2008). Secondly, there are *model based meta-features*: which describe certain characteristics of the learning system, such as maximum number of nodes per feature in a decision tree, kernel width of the gaussian kernel, or maximum depth of a decision tree.

Thirdly, *landmark meta-features* describe the performance (*i.e.* accuracy, mean squared error) of a learning algorithm (Kord et al., 2010; Brazdil et al., 2008).

Review of Meta Learning for Text Categorisation

Lam and Lai (2001) explain nine text meta-features for text categorisation, later expanded by Furdík et al. (2008) in their work. Both studies use the text meta-features to build a meta model, which selects the best algorithm for a given document category in document categorisation.

Text Meta-features

The text meta-features elucidated by Lam and Lai (2001) and Furdík et al. (2008) for document categorisation are:

1. **TraningInstancesPerCategory**: Number of positive training instances per category.

2. **TestingInstancesPerCategory**: Number of positive testing instances per category.

3. **AvgDocLenPerCategory**: The average document length of a category. The document length refers to the number of index terms in a document. The average is taken across all the positive documents within a category.

4. **AvgTermValPerCategory**: The average term weight of a document within a category. The average index term weight is taken for single document and the average is then computed for all the documents in a category.

5. **AvgMaxTermValPerCategory**: The average maximum term weight of a document within a category. The maximum index term weights for individual documents are summed and the average is taken for a category.

6. **AvgMinTermValPerCategory**: The average minimum term weight of a document within a category. The minimum index term weights for individual documents are summed and average is taken for a category.

7. **AvgTermThrePerCategory**:The average number of terms above a term weight threshold for a given category. The ‘term weight threshold’ is set

globally. The number of index terms above the term weight threshold are summed for category, and the average is computed for all instances in the category.

8. **AvgTopInfoGainPerCategory**:The average information gain of the top n index terms of a category. The information gain of each individual index term is computed for each category and ranked. The average is taken across top n index terms with the highest information gain within a category.

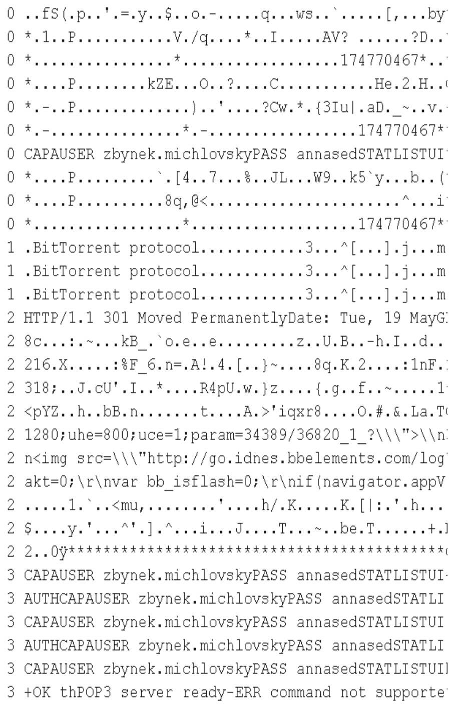
9. **NumInfoGainThresPerCategory**:The number of index terms in a category, where the information gain value exceeds a globally specified threshold.

The above text meta-features explained by Lam and Lai (2001) and Furdík et al. (2008) extracts statistical and information-theoretic information from the dataset, and are later used to train a meta model, which identifies the most suitable algorithm for a given document category.

Text Data versus String Data

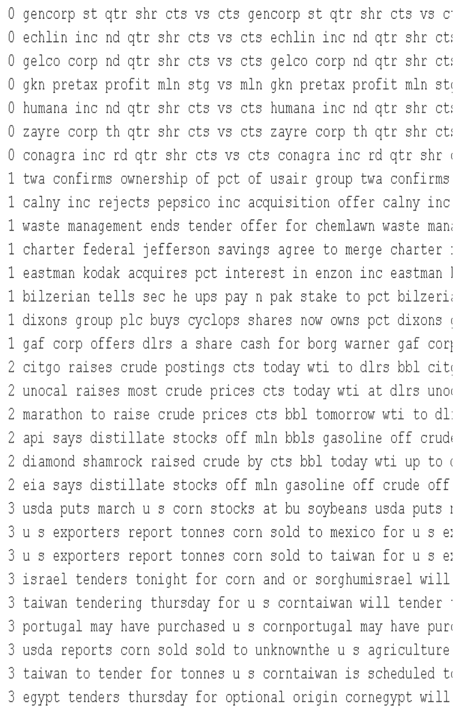
As the main focus of our research is to optimise string kernel SVMs, this section attempts to clearly define the difference between text and string data. According to Singhal (2001); De-Bie and Cristianini (2004); Shawe-Taylor and Cristianini (2004), a text dataset is a collection of *words*, where word or synonymously term is any sequence of letters separated by punctuation or spaces. On the other hand, a *string* is a finite sequence of *symbols* from an *alphabet* (Shawe-Taylor & Cristianini, 2004). This means even the word demarcation symbols like space and punctuation can be in a string.

If one is given a string and a text dataset, it is easy to categorise them both into their respective data types, considering their *term* separability (De-Bie & Cristianini, 2004). This can be illustrated more clearly using Figure 7.1 where there are three types of data: network data (Figure 7.1a), Reuters-21578 news data (Figure 7.1b), and spam data which consists of spam and non-spam emails (Figure 7.1c). While it is difficult to separate *terms* in both network data and spam data, *terms* are easily separable in Reuters-21578 data. This helps one



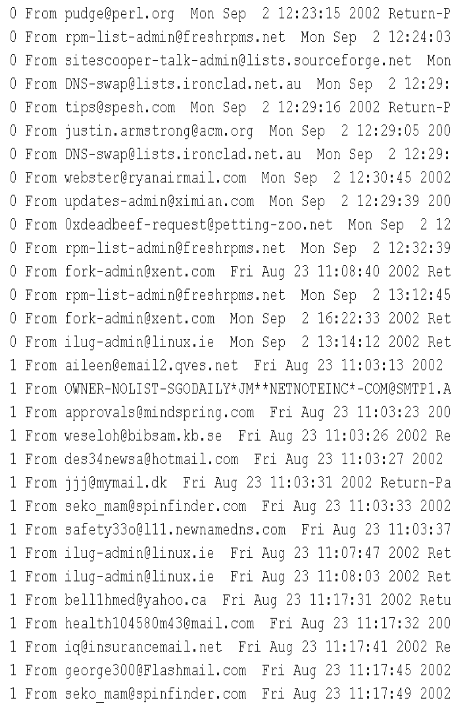
```
0 ..fS(.p..'.=.y..$..o.-.....q...ws..`.....[,...by
0 *.1..P...........V./q....*..I.....AV? ......?D..
0 *................*..................174770467*..
0 *....P........kZE...O..?....C...........He.2.H..
0 *.-..P.............)..'....?Cw.*.{3Iu|.aD._~..v.
0 *.-..............*.-.................174770467*
0 CAPAUSER zbynek.michlovskyPASS annasedSTATLISTUI
0 *....P.........`.[4..7...%..JL...W9..k5`y...b..(
0 *....P..........8q,@<......................^...i
0 *.................*..................174770467*
1 .BitTorrent protocol............3...^[...].j...m
1 .BitTorrent protocol............3...^[...].j...m
1 .BitTorrent protocol............3...^[...].j...m
2 HTTP/1.1 301 Moved PermanentlyDate: Tue, 19 MayG
2 8c...:.~...kB_.`o.e..e.........z..U.B..-h.I..d..
2 216.X.....:%F_6.n=.A!.4.[..}~....8q.K.2....:1nF.
2 318;..J.cU'.I..*....R4pU.w.}z....{.g..f..~.....1
2 <pYZ..h..bB.n.......t....A.>'iqxr8....O.#.&.La.T
2 1280;uhe=800;uce=1;param=34389/36820_1_?\\\">\\n
2 n<img src=\\\"http://go.idnes.bbelements.com/log
2 akt=0;\r\nvar bb_isflash=0;\r\nif(navigator.appV
2 .....1.`..<mu,........'....h/.K.....K.[|:.'.h...
2 $....y.'...^'.].^...i...J....T...~..be.T......+.
2 2..0ÿ*******************************************
3 CAPAUSER zbynek.michlovskyPASS annasedSTATLISTUI
3 AUTHCAPAUSER zbynek.michlovskyPASS annasedSTATLI
3 CAPAUSER zbynek.michlovskyPASS annasedSTATLISTUI
3 AUTHCAPAUSER zbynek.michlovskyPASS annasedSTATLI
3 CAPAUSER zbynek.michlovskyPASS annasedSTATLISTUI
3 +OK thPOP3 server ready-ERR command not supporte
```

(a) Network Data



```
0 gencorp st qtr shr cts vs cts gencorp st qtr shr cts vs c
0 echlin inc nd qtr shr cts vs cts echlin inc nd qtr shr ct
0 gelco corp nd qtr shr cts vs cts gelco corp nd qtr shr ct
0 gkn pretax profit mln stg vs mln gkn pretax profit mln st
0 humana inc nd qtr shr cts vs cts humana inc nd qtr shr ct
0 zayre corp th qtr shr cts vs cts zayre corp th qtr shr ct
0 conagra inc rd qtr shr cts vs cts conagra inc rd qtr shr
1 twa confirms ownership of pct of usair group twa confirms
1 calny inc rejects pepsico inc acquisition offer calny inc
1 waste management ends tender offer for chemlawn waste man
1 charter federal jefferson savings agree to merge charter
1 eastman kodak acquires pct interest in enzon inc eastman
1 bilzerian tells sec he ups pay n pak stake to pct bilzeri
1 dixons group plc buys cyclops shares now owns pct dixons
1 gaf corp offers dlrs a share cash for borg warner gaf cor
2 citgo raises crude postings cts today wti to dlrs bbl cit
2 unocal raises most crude prices cts today wti at dlrs uno
2 marathon to raise crude prices cts bbl tomorrow wti to dl
2 api says distillate stocks off mln bbls gasoline off crud
2 diamond shamrock raised crude by cts bbl today wti up to
2 eia says distillate stocks off mln gasoline off crude off
3 usda puts march u s corn stocks at bu soybeans usda puts
3 u s exporters report tonnes corn sold to mexico for u s e
3 u s exporters report tonnes corn sold to taiwan for u s e
3 israel tenders tonight for corn and or sorghumisrael will
3 taiwan tendering thursday for u s corntaiwan will tender
3 portugal may have purchased u s cornportugal may have pur
3 usda reports corn sold sold to unknownthe u s agriculture
3 taiwan to tender for tonnes u s corntaiwan is scheduled t
3 egypt tenders thursday for optional origin cornegypt will
```

(b) Reuters-21578 Data



```
0 From pudge@perl.org  Mon Sep  2 12:23:15 2002 Return-P
0 From rpm-list-admin@freshrpms.net  Mon Sep  2 12:24:03
0 From sitescooper-talk-admin@lists.sourceforge.net  Mon
0 From DNS-swap@lists.ironclad.net.au  Mon Sep  2 12:29:
0 From tips@spesh.com  Mon Sep  2 12:29:16 2002 Return-P
0 From justin.armstrong@acm.org  Mon Sep  2 12:29:05 200
0 From DNS-swap@lists.ironclad.net.au  Mon Sep  2 12:29:
0 From webster@ryanairmail.com  Mon Sep  2 12:30:45 2002
0 From updates-admin@ximian.com  Mon Sep  2 12:29:39 200
0 From 0xdeadbeef-request@petting-zoo.net  Mon Sep  2 12
0 From rpm-list-admin@freshrpms.net  Mon Sep  2 12:32:39
0 From fork-admin@xent.com  Fri Aug 23 11:08:40 2002 Ret
0 From rpm-list-admin@freshrpms.net  Mon Sep  2 13:12:45
0 From fork-admin@xent.com  Mon Sep  2 16:22:33 2002 Ret
0 From ilug-admin@linux.ie  Mon Sep  2 13:14:12 2002 Ret
1 From aileen@email2.qves.net  Fri Aug 23 11:03:13 2002
1 From OWNER-NOLIST-SGODAILY*JM**NETNOTEINC*-COM@SMTP1.A
1 From approvals@mindspring.com  Fri Aug 23 11:03:23 200
1 From weseloh@bibsam.kb.se  Fri Aug 23 11:03:26 2002 Re
1 From des34newsa@hotmail.com  Fri Aug 23 11:03:27 2002
1 From jjj@mymail.dk  Fri Aug 23 11:03:31 2002 Return-Pa
1 From seko_mam@spinfinder.com  Fri Aug 23 11:03:33 2002
1 From safety33o@l11.newnamedns.com  Fri Aug 23 11:03:37
1 From ilug-admin@linux.ie  Fri Aug 23 11:07:47 2002 Ret
1 From ilug-admin@linux.ie  Fri Aug 23 11:08:03 2002 Ret
1 From bell1hmed@yahoo.ca  Fri Aug 23 11:17:31 2002 Retu
1 From health104580m43@mail.com  Fri Aug 23 11:17:32 200
1 From iq@insurancemail.net  Fri Aug 23 11:17:41 2002 Re
1 From george300@Flashmail.com  Fri Aug 23 11:17:45 2002
1 From seko_mam@spinfinder.com  Fri Aug 23 11:17:49 2002
```

(c) Spam Data

Figure 7.1: String and Text Data: (a) Network traffic data produced by network applications. (b) Data from Reuters-21578 dataset. (c) Spam data which consists of ham and Spam E-mail messages (in every instance, the first character represents the class label followed by actual data)

to categorise both network data and spam as string data and Reuters-21578 as text data.

With this difference between string data and text data in mind, the next subsections explain our motivation to use text meta-features explained for string classification.

Motivation for using Meta Learning for String Categorisation

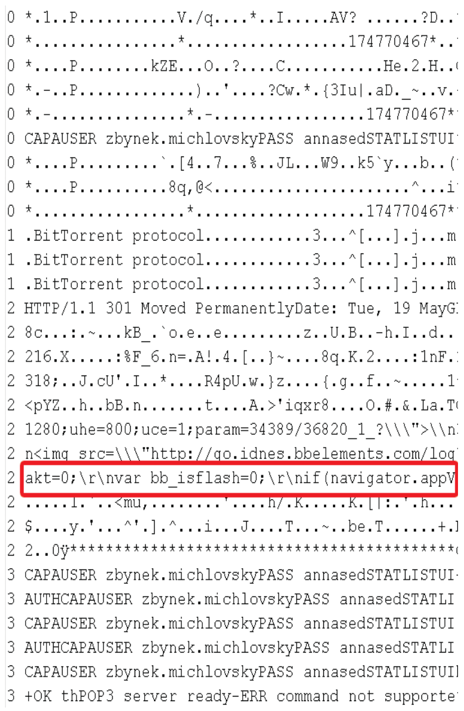
All text meta features described before (apart from *TraningInstancesPerCategory* and *TestingInstancesPerCategory*) are computed using terms and their frequencies. In this way it is possible to represent a string dataset as a collection of terms and their frequencies, which helps to derive some string meta-features that are computed by terms and their frequencies. Now these string meta-features help to employ meta learning on string classification.

7.2.2 *Meta Learning for String Kernel SVM Optimisation*

String Meta-features

In order to use the meta-features as discussed in section 7.2.1 for string classification, the string dataset needs to be presented as terms and term frequencies. We accomplish this in a string dataset by using splitting characters: ”+’:(){}[]. ,-\” to split a string into set of terms or synonymously *tokens*. This approach is referred as ‘tokenisation’ in the literature (Shawe-Taylor & Cristianini, 2004). To explain tokenisation, consider the highlighted string in Figure 7.2, which is from a network application that uses *http* protocol. Using specified splitting characters, one can split the string into tokens: “$akt = 0;$”, “r”, “$nvarbb_isflash = 0;$”, “r”, “nif”, “$navigator$”, and “$appV$”. Now a *token frequency table* is generated, as shown in Table 7.1. This token-frequency information is used to compute the string meta-features explained in this section.

The main difference between the text meta features discussed in section 7.2.1 and string meta-features explained here, is that the *text meta-features* are calculated for a text category in a text dataset, but, the *string meta-features* are calculated for the entire string dataset. Out of nine text meta-features discussed in the section 7.2.1, seven are considered in deriving these string meta-features.



```
0 *.1..P...........V./q....*..I.....AV? ......?D..
0 *................*.................174770467*..
0 *....P........kZE...O..?....C...........He.2.H..
0 *.-..P.............)..'....?Cw.*.{3Iu|.aD._~..v.
0 *.-...............*.-.................174770467*
0 CAPAUSER zbynek.michlovskyPASS annasedSTATLISTUI
0 *....P.........`.[4..7...%..JL...W9..k5`y...b..(
0 *....P..........8q,@<......................^...i
0 *................*..................174770467*
1 .BitTorrent protocol...........3...^[...].j...m
1 .BitTorrent protocol...........3...^[...].j...m
1 .BitTorrent protocol...........3...^[...].j...m
2 HTTP/1.1 301 Moved PermanentlyDate: Tue, 19 MayG
2 8c...:.~...kB_.`o.e..e.........z..U.B..-h.I..d..
2 216.X.....:%F_6.n=.A!.4.[..}~....8q.K.2....:1nF.
2 318;..J.cU'.I..*....R4pU.w.}z....{.g..f..~.....1
2 <pYZ..h..bB.n.......t....A.>'iqxr8....O.#.&.La.T
2 1280;uhe=800;uce=1;param=34389/36820_1_?\\\">\\n
2 n<img src=\\\"http://go.idnes.bbelements.com/log
2 akt=0;\r\nvar bb_isflash=0;\r\nif(navigator.appV
2 .....I.`..<mu,........'....h/.K.....K.[|:.'.h...
2 $....y.'...^'.].^...i...J....T...~..be.T......+.
2 2..0ÿ*******************************************
3 CAPAUSER zbynek.michlovskyPASS annasedSTATLISTUI
3 AUTHCAPAUSER zbynek.michlovskyPASS annasedSTATLI
3 CAPAUSER zbynek.michlovskyPASS annasedSTATLISTUI
3 AUTHCAPAUSER zbynek.michlovskyPASS annasedSTATLI
3 CAPAUSER zbynek.michlovskyPASS annasedSTATLISTUI
3 +OK thPOP3 server ready-ERR command not supporte
```

Figure 7.2: String Data: Network traffic data produced by network applications (in every instance, the first character represents the class label of the network application, followed by actual network traffic data)

Assume a string dataset that has n number of instances. The seven string meta-features are:

1. **AvgInstanceLen**: The average instance length of the dataset. The instance length refers to number of tokens in an instance. The average is taken across all the instances. If i^{th} instance has N_i tokens, then the average instance length for that dataset is $\frac{\sum_{i=1}^{n} N_i}{n}$.

2. **AvgTokenVal**: The average token weight of an instance across a string dataset. Initially, the token weight is calculated for each token and the average is computed for single instance. Then, the average token weights for each instance are summed and the average is computed for all the instances.

   If there are m unique tokens in i^{th} instance, the average token weight for a string dataset is written as:

$$\textit{Average token weight of the string dataset} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} TW(j,i)}{mn},$$

Token	Token Frequency
$akt = 0;$	1
r	2
$nvarbb_isflash = 0;$	1
nif	1
$navigator$	1
$appV$	1

Table 7.1: Token Frequency Table: Tokens and their frequencies for highlighted string in Figure 7.2

$$ \tag{7.8}$$

where $TW(j,i)$ is the token weight of j^{th} token in i^{th} instance. According to the interpretation given by Hersh (2008) of the *term weight*, the $TW(j,i)$ can be written as:

$$TW(j,i) = \textit{TF}(j,i) \times \textit{IDF}(j), \tag{7.9}$$

where *IDF(*j*)* is the inverse document frequency of j^{th} token, and *TF(*j,i*)* is the frequency of j^{th} token in instance i. Furthermore, according to (Hersh, 2008), the *IDF(*j*)* is computed as:

$$IDF(j) = \log \frac{n}{TF(j)} + 1, \tag{7.10}$$

where $TF(j)$ is the frequency of the j^{th} token in the dataset. Now, considering (7.9) and (7.10), equation (7.8) is rewritten as:

$$\textit{Average token weight of the string dataset} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} TF(j)\left(\log\frac{n}{TF(j)}+1\right)}{mn} \tag{7.11}$$

3. **AvgMaxTokenVal**: The average maximum token weight of an instance across a string dataset. Maximum token weights of an instance are summed and the average is taken across all instances.

4. **AvgMinTokenVal**: The average minimum token weight of an instance across a given string dataset. Minimum token weights of an instance are summed and the average is taken across all instances.

5. **AvgTokenThre**: The average number of tokens above a token weight threshold for a given string dataset. The token weight threshold is set globally. The number of tokens where the token weight is above the threshold are summed and the average is taken across all instances.

6. **AvgTopInfoGain**: The average information gain of the top r tokens in the string dataset. The information gain of each individual token is computed for each instance and ranked. Then, the average is taken across top r terms with highest information gain.

7. **NumInfoGainThres**:The average number of tokens in an instance where the information gain value exceeds a globally specified threshold.

Meta Learning for String Classification

The mentioned string meta-features help to employ meta learning on a string classification. The principle of using meta learning for string classification is discussed in section 7.2.2. A novel string kernel SVM optimisation method is elucidated in section 7.2.2.

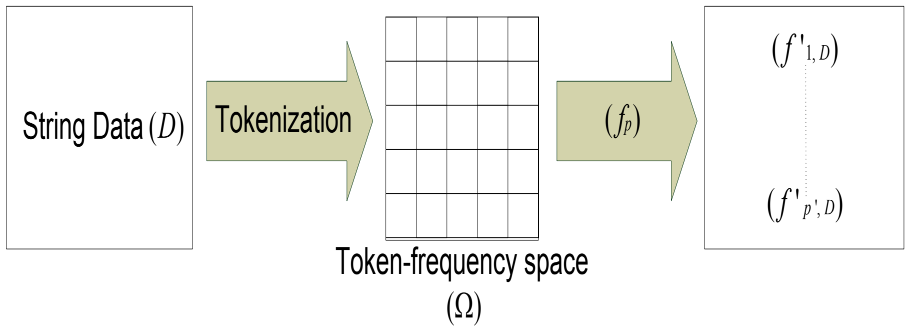


Figure 7.3: String meta-feature generation process

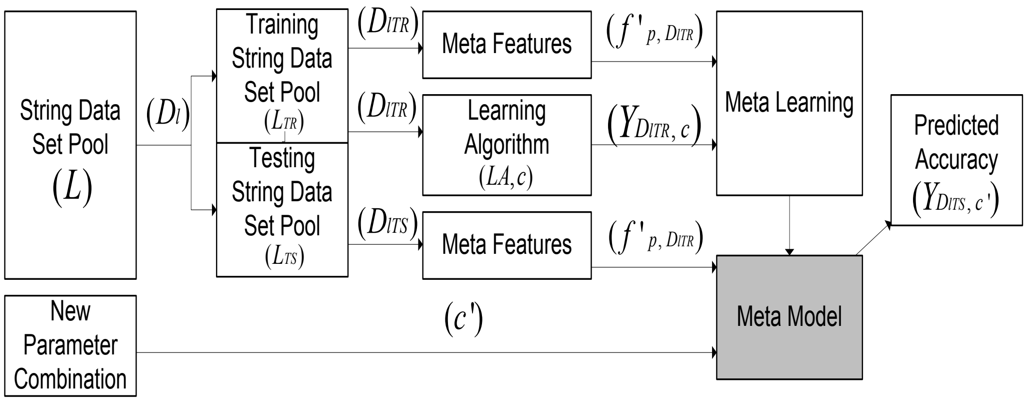

String Data Set Pool (L)
(Dl)
Training String Data Set Pool (LTR)
Testing String Data Set Pool (LTS)
(DlTR)
(DlTR)
(DlTS)
Meta Features
Learning Algorithm (LA,c)
Meta Features
(f'p,DlTR)
(YDlTR,c)
(f'p,DlTR)
Meta Learning
Predicted Accuracy (YDlTS,c')
Meta Model
New Parameter Combination
(c')

Figure 7.4: The procedure to employ meta learning for string classification

Meta Learning for String Classification: Principle

Consider a string dataset D, which is represented as a vector in *token-frequency space* Ω, where each dimension in Ω is associated with one *token*. Now, the dataset D is represented via function ω in this new Ω token-frequency space:

$$\omega(D) = (TF(t_1, D), TF(t_2, D), ..., TF(t_N, D)) \in \Omega, \tag{7.12}$$

where $TF(t_j, D)$ is the token frequency of j^{th} token in the string dataset D, and N is the number of unique tokens in the dataset. Now one can derive a function $f_p : \Omega \to \mathbb{R}$:

$$f_p(D) = f'_{p,D} \tag{7.13}$$

where $f'_{p,D}$ represents the value for the p^{th} string-meta feature for D. For the string dataset D, there are p' finite meta-features, where all string meta-features $f_p(D)$ ($p = 1, 2, 3, ..., p'$) are well defined. This sting meta-feature generation process is shown in Figure 7.3.

Using the above discussed sting meta-features, Figure 7.4 explains the principle of meta learning for string classification. Assume there is a string dataset pool L with l' datasets, where, each string dataset D_l ($D_l \in L, l = 1, ..., l'$) is again subdivided into unique D_{lTR} (training) and D_{lTS} (testing) datasets, which creates training (L_{TR}) and testing (L_{TS}) dataset pools. The string meta-feature $f'_{p,D_{lTR}}$ is computed for dataset D_{lTR}. Also, for D_{lTR}, the machine learning algorithm LA with parameter combination c, generates $Y_{D_{lTR},c}$ classification accuracy. These computed string meta-features ($f'_{p,D_{lTR}}$), parameter combinations (c) and accuracy information ($Y_{D_{lTR},c}$) generate a meta model via regression, which is able to predict the classification accuracy for a new string dataset, given the computed sting meta-features and the parameter combination. Hence, for a new string dataset D_{lTS}, the meta model predicts the accuracy $Y_{D_{lTS},c'}$ for parameter combination c' by computing string-meta features $f'_{p,D_{lTS}}$.

Meta Learning for String Kernel SVM Optimisation: Algorithm

According to the principle introduced in this section the built meta model is able to predict the string classification accuracy for a machine learning algorithm on a novel string dataset, using computed string meta-features. This section explains the procedure to use this principle (meta learning for string classification) to optimise string kernel SVMs, which is shown in Algorithm 8.

Algorithm 8 explains the procedure of using meta learning to optimise string kernel SK with SVM. The proposed algorithm uses training string dataset pool L_{TR}, testing string dataset pool L_{TS}, training parameter pool C and testing parameter pool C' as inputs. Also, SVM with string kernel SK is set as the learning algorithm (LA). Initially, a meta model is built using meta features calculated for each dataset in L_{TR} with accuracy information obtained for each parameter combination in C via n-fold cross validation. Then, for each new string dataset in L_{TS}, the built meta model predicts the classification accuracy for each combination in C'. The combination ($\hat{c}_l$) which yields the highest accuracy is presented as the optimum parameter combination for dataset D_{lTS}.

7.3 A NOVEL FAST STRING CLASSIFICATION METHOD

7.3.1 *Motivation*

Despite of the high performance of meta learning string kernels SVMs on classification accuracy, the computational cost of the learning process is huge. Alternatively, we developed another novel method *string kernels MEB* which

```
input :
        $L_{TR}$ =Training String Dataset Pool
        $L_{TS}$ =Testing String Dataset Pool
        $C$ =Parameter Combination Pool for Training ($c \in C$)
        $C'$ =Parameter Combination Pool for Testing ($c' \in C'$)
        $LA$ =SVM with String Kernel $SK$
output: Parameter combination $\hat{c}_l$ which yields the best accuracy for
        sting dataset $D_{lTS}$

for $l \leftarrow 1$ to $l'$ do
    Pick $D_{lTR}$ from $L_{TR}$
    for $p \leftarrow 1$ to $p'$ do
        Compute $f'_{p,D_{lTR}}$
    end
    repeat
        Pick a parameter combination $c$ from $C$
        Do 10-fold cross validation on $D_{lTR}$, using $LA$ with parameter
        combination $c$ which yields $Y_{D_{lTR},c}$ accuracy
    until no more parameter combinations in $C$;
end

Build a regression model (meta model) using $f'_{p,D_{lTR}}$, $c$, and $Y_{D_{lTR},c}$
for $l \leftarrow 1$ to $l'$ do
    Pick $D_{lTS}$ from $L_{TS}$
    for $p \leftarrow 1$ to $p'$ do
        Compute $f'_{p,D_{lTS}}$
    end
    repeat
        Pick a parameter combination $c'$ from $C'$
        Predict accuracy $Y_{D_{lTS},c'}$ for $LA$ with parameter combination $c'$
        using build meta model
        if $Y_{D_{lTS},c'}$ is maximum then
            $\hat{c}_l = c'$
        end
    until no more parameter combinations in $C$;
end
```

Algorithm 8: The proposed meta learning algorithm for string kernel SVM optimisation

extends original MEB/CVM algorithm with string kernels methods. As we discussed in chapter 4 the computational complexity of MEB is much smaller than SVMs, the string kernel learning time for string kernels MEB is shorter.

7.3.2 *String Kernels MEB Algorithm Description*

Given a data matrix $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$, MEB over $\mathbf{X}$ is modelled as the smallest hypersphere $B_{\mathbf{X}} = \{\mathbf{c}, r, \mathbf{X}\}$, where $\mathbf{c}, r$ are the centre and radius of $B_{\mathbf{X}}$, respectively. The $B_{\mathbf{X}}$ is calculated by solving the optimisation problem:

$$\begin{array}{ll} \min_{\mathbf{c},r} & r^2 \\ subject\ to & ||\mathbf{c} - \phi(\mathbf{x}_i)||^2 = (\phi(\mathbf{x}_i) - \mathbf{c})'(\phi(\mathbf{x}_i) - \mathbf{c}) \leq r^2 \\ & i = 1, 2, \ldots, l \end{array} . \tag{7.14}$$

which can be solved by introducing a Lagrange multiplier $\alpha_i \geq 0$ for each constraint

$$\begin{array}{ll} L(\mathbf{c}, r, \alpha) & = r^2 + \sum_{i=1}^{l} \alpha_i \left[||\phi(\mathbf{x}_i) - \mathbf{c}||^2 - r^2 \right] \\ & = \ldots \\ & = \sum_{i=1}^{l} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{l} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \end{array} . \tag{7.15}$$

Given a set of string $S = s_1, s_2, \ldots, s_l$ while is defined from alphabet Σ of $|\Sigma|$ symbols. MEB B_S is able to be obtained by transferring string kernels (e.g. bag-of-words) of S into equation 7.15 as:

$$L(\mathbf{c}, r, \alpha) = \sum_{i=1}^{l} \alpha_i \kappa_{bow}(s_i, s_i) - \sum_{i,j=1}^{l} \alpha_i \alpha_j \kappa_{bow}(s_i, s_j). \tag{7.16}$$

The optimised Lagrange parameter α^* is calculated by solving the problem as:

$$\begin{array}{ll} \min_{\alpha} & \sum_{i=1}^{l} \alpha_i \kappa_{bow}(s_i, s_i) - \sum_{i,j=1}^{l} \alpha_i \alpha_j \kappa_{bow}(s_i, s_j) \\ subject\ to & \sum_{i=1}^{l} \alpha_i = 1,\ and\ \alpha_i \geq 0,\ i = 1, 2, \ldots, l \end{array} . \tag{7.17}$$

According to the Karush.Kuhn.Tucker (KKT) theorem that the Lagrange parameters can be non-zero only if the corresponding inequality constraint is an equality at the solution, only these $S_{sv} \subset S$ that lie on the surface of the optimal hypersphere have their corresponding α_i^* non-zero. Thus, we treat S_{sv} as the support vectors of S.

7.3.3 *Online String Kernels*

In this stage, Meta Learning String Kernel SVMs and String Kernel MEBs can only work in batch learning mode. However, the combination of string kernel with MEB gives a new direction of string kernel online learning mode. In the future, we can implement the string kernel online learning by taking a suitable online kernel learning method and replace the kernel matrices with the string kernels matrices.

7.4 SUMMARY

This chapter gives a brief introduction to string kernels SVM and presents two novel string kernels learning methods: Meta Learning String Kernel SVMs and String Kernel MEBs. Although these two methods are for batch learning only, the combination of batch learning methods and online learning methods is also proper for stream data learning. These two methods are able to outperform traditional string kernels methods on both accuracy and efficiency. The evaluation of these two methods is presented in chapter 9

Chapter 8

THE APPLICATION OF HIERARCHICAL CORE VECTOR MACHINES TO SOLVE NETWORK INTRUSION DETECTION

This chapter presents a case study of network intrusion detection. We use RCV1-V2 (Lewis, Yang, Rose, & Li, 2004) for a simulation of network strings examination as RCV1-V2 gives a one level multi-label (*i.e.* not hierarchical multi-label) text classification problem. KDD'99 (KDD99, 1999) poses a hierarchical multi-label Internet security problem, we thus use KDD'99 as a comprehensive network security real application case study.

8.1 BACKGROUND

Upon the explosive growth of number of Internet users, malicious activity such as denial of service attacks, port scans or even attempts to crack into computers by monitoring network traffic are also increasing. These malicious activities on the Internet, also known as network intrusions, trouble the Internet users and cause Internet security threats. Network intrusion detection systems (NIDS) are developed for safeguarding the users and their systems from any threats of Internet intrusion. As a preventive measure, existing NIDS mostly employ two popular intrusion detection methods: (1) the white and black list methods filter spam through network ID/address validity verification (Chirita, Diederich, & Nejdl, 2005); (2) the Intrusion signature recognition methods detect any intrusions by comparing network data with a predetermined attack signature (Ye & Chen, 2001). However, both methods have proved brittle to any small alteration of attack (Kofcz, Chowdhury, & Alspector, n.d.). This requires ordinary NIDS (e.g. a firewall system for a personal computer) to be updated in a regular time interval.

As an alternative solution, machine learning methods solve the above difficulty confronted by existing NIDS, because the attack signature/filter are learned dynamically from the streaming network data. In this field, Mukkamala and Sung (2003) implemented support vector machine (SVM) method on the KDD Cup dataset and obtained an overall 99% classification accuracy. Frank and Mda-c (1994) took decision trees as the most suitable classification method for intrusion categorisation. Panda and Patra (2007) compared the performance of Naive Bayes with the neural network approach, and authenticated the suitability of Naive Bayes for intrusion detection modelling. Despite that, machine learning for network intrusion detection still have underlying issues related to accurate network threats authentication. With the vulnerability of present-day software and protocols combined with the increasing sophistication of attacks, network-based attacks are on the rise (Staff, 2005), which has made an astonishing revenue loss every year. The 2005 annual computer crime and security survey (CSI & FBI, 2005) reported that the financial losses incurred by the respondent companies due to network attacks/intrusions were US $\$130$ million.

In terms of using machine learning for intrusion detection, existing methods perform intrusion detection by grouping any network traffic into several major clusters. This in practice often reduces the detection rate/efficiency because the detailed information of network traffic, including attack messages, is ignored. For example, spam is simply divided into several major categories, such as junk mail, IM spam, TXT spam etc. It could be obviously advantageous to spam detection, if the junk mail could be dividable into unsolicited bulk e-mail (UBE) and unsolicited commercial e-mail (UCE). Thus, it is important for network intrusion detection that the whole picture of network traffic/intrusion is presented as a hierarchy as seen in Fig.5.1, and it is necessary to do multipl labelling of the network traffic/intrusions.

8.1.1 *Multi-label Intrusions*

Intrusion detection is formatted into a hierarchical multi-label classification (HMC) problem. A high-speed hierarchal minimum enclosing balls (HCVM) algorithm is proposed for multi labelling network intrusions in a MEB/CVM structure where MEBs/CVMs are separating, encompassing, and overlaping with each other. A minimum enclosing ball (MEB/CVM) computes a hypersphere of minimum radius which encloses a specific type of network traffic/intrusion. Given an unlabelled network message, the proposed HCVM seeks a MEB/CVM enclosing the message, and multi-labels it by the MEB/CVM's position and its ancestor relationship in the HCVM. In the experiments, we have authenticated the proposed HCVMs classification proficiency and computational efficiency. For the HMC from KDD'99 dataset, the proposed HCVM exhibits an outstanding classification accuracy for U2R and R2L attack types. Furthermore, the capability of HCVM for handling single-level multi-label data is also demonstrated in another experiment with the RCV1-V2 dataset.

8.2 EXPERIMENTS AND DISCUSSION

For testing the capability of the proposed HCVM for multi-label, in particular for hierarchical multi-label classification, we examined the developed HCVM on the RCV1-V2 text dataset (Lewis et al., 2004), and the benchmark KDD'99 dataset (KDD99, 1999), respectively.

8.2.1 *Experimental setup*

The proposed HCVM is implemented in MATLAB version (7.6.0), on a $1.86Hz$ Intel Core 2 machine with $2GB$ RAM. In our experiments, we use a non-linear Gaussian kernel $k(x, y) = exp(-\left\| x - y \right\|^2 / \beta)$ with $\beta = \frac{1}{m^2} \sum_{i,j=1}^{m} \left\| x_i - y_j^2 \right\|$. We set the ϵ as $1e - 6$ for $(1 + \epsilon)$-approximation of MEB/CVM, and the Gaussian kernel parameter β as 1000.

Parameter n in Algorithm 6 is a positive integer, representing the number of the nearest MEBs/CVMs to the input instance. Thus, n is greater than or equal to the maximum number of multi-labels for a single instance. For example, in the case of the RCV1-V2 dataset, n takes a value of 11. It is worth noting that n in practice is more often determined by cross-validation tests over the training data. In the case of KDD'99 dataset, n is set to 3 because the highest 93.6% accuracy is achieved when n equals 3 in cross-validation tests, as shown in Fig. 8.1.

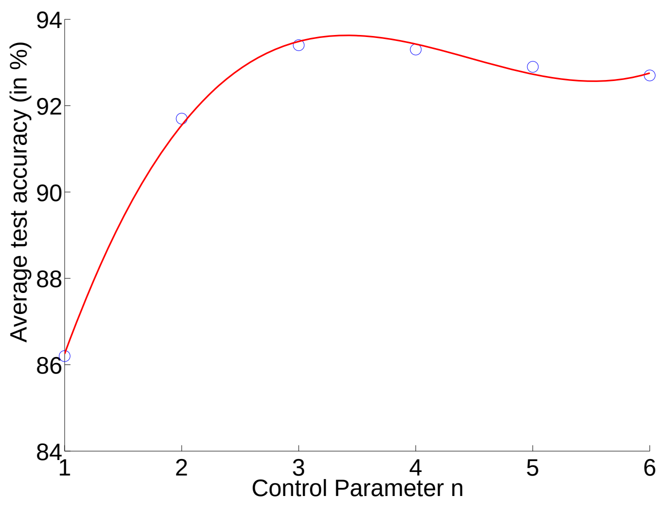


Figure 8.1: Cross validation tests of Algorithm 6 for parameter n determination on KDD'99 dataset

8.2.2 *Simulation of Network Strings Examination: RCV1-V2 Dataset*

The RCV1-V2 data set collected by Lewis et al. (2004) contains newswire stories from Reuters Ltd. RCV1-V2 and was pre-processed by several schemes, including removing stopping words, stemming, transforming the documents to vectors with TF-IDF format and cosine normalisation, etc. Among the three category sets (Topics, Industries, and Regions) of RCV1-V2, we consider, analogous to the previous works done by Lewis et al. (2004), only the Topics category set, with the statistics of training and testing dataset is given in table 8.1.

As a simulating network string examination, we implement Algorithm 5 for HCVM training, and Algorithm 6 for testing the generated HCVM model. Algorithm 6 checks the n nearest MEBs/CVMs of the HCVM, and labels each new instance as one of the classes of the MEBs/CVMs enclosing the instance.

Data set	#inst.	#feat.	#label	Label size frequency (%)				
				1	2	3	4	≥ 5
RCV1-V2 train	$23,149$	$47,236$	101	12.3%	29.5%	35.7%	10.8%	11.7%
RCV1-V2 test	$781,265$	$47,236$	103	12.3%	25.4%	45.3%	10.9%	15.3%

Table 8.1: *Statistics of RCV1-V2 dataset*

For performance evaluation, we compare HCVM with SVM (Chang & Lin, 2001) on one of the most used performance measures for information retrieval (Hripcsak & Rothschild, 2005), F-measure, which is calculated as the harmonic mean of precision (P) and recall (R):

$$F - measure = \frac{2PR}{P+R}. \tag{8.1}$$

where

$$P = \frac{\sum_{i=1}^{l} \widehat{y}_j^i y_j^i}{\sum_{i=1}^{l} \widehat{y}_j^i}\ ,\ R = \frac{\sum_{i=1}^{l} \widehat{y}_j^i y_j^i}{\sum_{i=1}^{l} y_j^i}, \tag{8.2}$$

y_j^i is the true label of i^{th} instance for class j from total l testing instances and $\widehat{y}_j^i$ is the predicted label. In our experiments, we use two approaches: macro-average F-measure and micro-average F-measure (Tague-Sutcliffe, 1997) which extended the F-measure from single-label to multi-label. The macro-average F-measure for a total of d labels is obtained from the unweighted mean of the label F-measures and can be calculated as,

$$F = \frac{1}{d} \sum_{j=1}^{d} \frac{2 \sum_{i=1}^{l} \widehat{y}_j^i y_j^i}{\sum_{i=1}^{l} \widehat{y}_j^i + \sum_{i=1}^{l} y_j^i}. \tag{8.3}$$

The micro-average F-measure, which considers predictions from all instances together and calculate the F-measure across all labels,

$$micro - F = \frac{2\sum_{i=1}^{d}\sum_{i=1}^{l}\widehat{y}_j^i y_j^i}{\sum_{i=1}^{d}\sum_{i=1}^{l}\widehat{y}_j^i + \sum_{i=1}^{d}\sum_{i=1}^{l} y_j^i}. \tag{8.4}$$

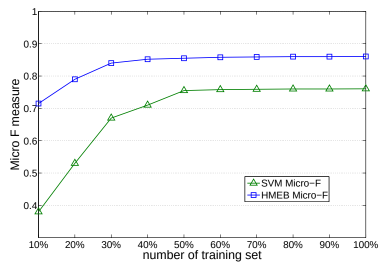
The additional measurement we used for performance evaluation is AUC (the area under the Receiver Operating Characteristic (ROC) curve). AUC reflects mainly the ranking quality of predictions. A perfect classifier has an expected AUC score equal to 1. A detailed description of the AUC and ROC graphs can be found in (Fawcett, 2003).

Methods	F-Measure	Macro-Average F-Measure	AUC
SVM	0.76	0.51	0.85
HCVM	**0.86**	**0.71**	**0.91**

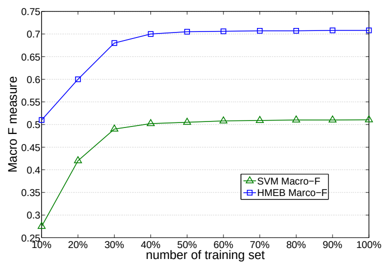
Table 8.2: *AUC, F-Measure and Macro F-Measure comparison*

Table 8.2 presents the comparison result. The proposed HCVM performs 10% better than SVM on Macro F-measure, 20% better on Micro F-measure, and 6% better on AUC under the condition that the complete dataset is used for training. Fig. 8.2 gives further comparison on Macro F-measure and Micro F-measure, when the sizes of training sets vary from 10% to 100%. As seen, the proposed HCVM consistently and significantly outperforms the SVM by $10\% - 30\%$ on both Macro F-measure and Micro F-measure despite the change in the size of training set. Fig. 8.3 gives the ROC comparison, where the AUC area of HCVM is seen compassing the AUC area of SVM, which demonstrates again that the proposed HCVM provides crisply better quality of predictions than SVM at any size of training set.

The poor performance of SVM can be explained in part by the fact that the SVM hinge loss is not well suited for unbalanced data (Zhang & Oles, 2000). However, HCVM is the aggregation of MEBs/CVMs on multi-label hierarchy. The advantages of HCVM on class-imbalanced problem are: (1) single MEB/CVM is capable of modelling whatever skewed class (i.e. a class

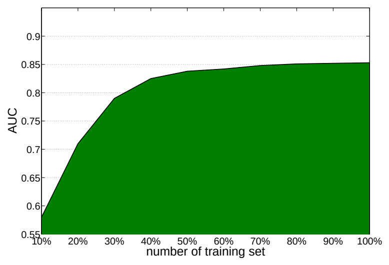


(a) Micro-Average F-measure

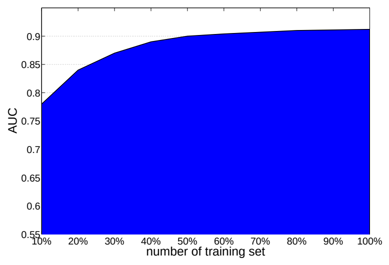


(b) Macro-Average F-measure

Figure 8.2: (a) and (b) plot the Micro-Average F-measure and Macro-Average F-measure of different number of training dataset for both SVM and HCVM respectively.



(a) ROC graph for SVM



(b) ROC graph for HCVM

Figure 8.3: ROC graphs with AUC results of 10 chunks training dataset with same size. The filled areas in (a) and (b) correspond to the AUC for SVM and HCVM respectively.

with fewer samples than other classes) more accurately; and (2) HCVM is convenient to allocate more computing power (i.e. more MEBs/CVMs) for the classification of the skewed-class.

8.2.3 *Internet Security Application: KDD'99 Datasets*

KDD'99

KDD Cup 1999 dataset (KDD99, 1999) contains 5 million Internet connection records. Each record encodes 41 connection features including 1 class label, 34 continuous features, and 7 symbolic features. The class label identifies one of 22 network connection types including normal, buffer_overflow, guess_password etc. All connection features are assumed to be conditionally independent. Apart from the normal connection type, the rest of 21 attack types are associated with 4 major categories of attack, they are:

1. DOS, denial of service, e.g. back;
2. R2L, unauthorised access from a remote machine, e.g. guessing password;
3. U2R, unauthorised access to local superuser (root) privileges, e.g. buffer overflow;
4. PROBE, information gathering, e.g. port sweep.

Data Preprocessing

For data preprocessing, we replace the original label for each instance with a set of numerical labels by the following rules: the major attack categories stay at the head followed by its subcategories. For example, 'back', as the first subcategory of DOS attack, is represented as (1,1), and the second subcategory 'land', as (1,2). In this way, all labels can be transformed into a numerical hierarchical structure as shown in Fig. 8.4. Additionally, we normalise every continuous feature into $[0, 1]$, and encode every symbolic feature to binary digit.

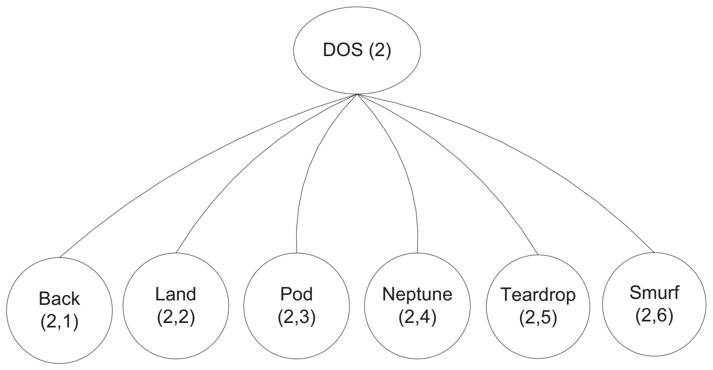


Figure 8.4: KDD99 data classification label replacement

Comparison Results

For constructing HCVM, we compute MEBs/CVMs and core sets for 21 subcategories and 4 major categories of attacks, respectively. For each category and subcategory, we use the instances of the core set as the core vectors. As a result, we obtain 25 core vector sets in total corresponding to 4 major categories plus 21 subcategories of attacks. Table 8.3 and Figure 8.5 present the comparison of our classification results to the results achieved by Bernhard who is the winner of KDD'99 cup (Elkan, 2000).

Actual VS Predicted	Actual Normal	Actual DOS	Actual U2R	Actual R2L	Actual Probe
Bernhard predicted Normal	60262	5299	168	14527	511
HCVM predicted Normal	**920431**	36818	9845	2603	2084
Bernhard predicted DOS	78	223226	0	0	184
HCVM predicted DOS	153064	**3619301**	10341	14732	85932
Bernhard predicted U2R	4	0	30	8	0
HCVM predicted U2R	6	3	**43**	0	0
Bernhard predicted R2L	6	0	10	1360	0
HCVM predicted R2L	321	193	44	**517**	51
Bernhard predicted Probe	243	1328	20	294	3471
HCVM predicted Probe	2843	2104	412	506	**35237**
Bernhard total accuracy	99.5%	97.1%	13.2%	8.4%	83.3%
HCVM total accuracy	94.6%	93.2%	**82.7%**	**45.9%**	**85.7%**

Table 8.3: *Compare the classification accuracy with (Bernhard 1999) who is the winner of KDD'99 cup*

As seen from the table, Bernhard (1999) achieved an extremely high classification accuracy of 99.5% on normal connect type, however U2R and R2L showed poor classification performance, with none of them exceeding 15%, because of their class size being smaller than the other classes. Although the overall classification accuracy of the proposed HCVM is slightly lower than that of the Bernhard's method for normal connection type and DOS type attacks, the proposed HCVM outperforms Bernhard's method on the classification of 3 most important classes. HCVM particularly increases the classification accuracy of U2R and R2L by 70% and 35%, respectively. This demonstrates the advantage of the proposed HCVM, which enables a more accurate approximation of class information, even for a very small dataset.

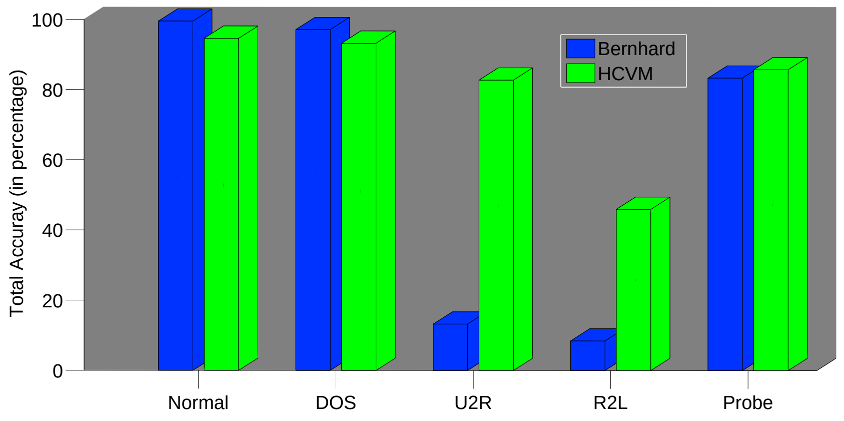


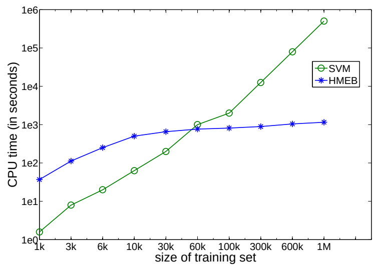
Figure 8.5: Compare the classification accuracy with the winner of KDD'99 cup on each attack type

To evaluate the efficiency of the proposed methods we computed the number of CPU time, support vectors and test error rate of HCVM and SVM with different sizes of training datasets. As seen in Fig. 8.6a, the time cost of HCVM increases more slowly than the SVM. In addition, Fig. 8.6b shows the number of support vectors found by HCVM stays constant regardless of the training dataset size. For SVM, the number of support vectors increases proportionally to the dataset size. On comparison of test error rate, Fig. 8.6c indicates that HCVM constantly provides lower error rate than SVM, specially when the training dataset size is small.

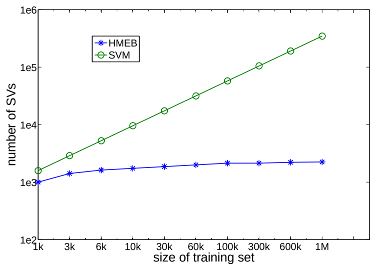
8.3 SUMMARY

We have evaluated the proposed HCVM method for network intrusion detection applications In our experiments, we have implemented the HCVM on benchmark datasets from UCI archives such as KDD'99 and RCV1-V2. The experimental results show that the proposed HCVM is clearly more efficient and accurate than the traditional methods, especially for the U2R and R2L problem of the KDD'99 dataset.

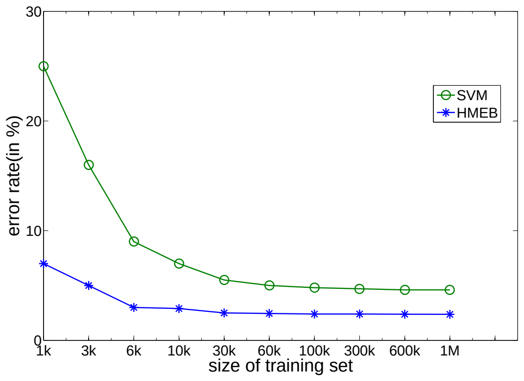
Next chapter presents a case study for face membership authentication.



(a) CPU time



(b) number of support vectors



(c) classification error rate

Figure 8.6: (a) CPU time required by HCVM and SVM, (b) number of support vectors created by HCVM and SVM with different size of training dataset, respectively. (c) the error rate variation against the size of training dataset.

Chapter 9

THE APPLICATION OF DYNAMIC EVOLVING CORE VECTOR MACHINES TO SOLVE FACE MEMBERSHIP AUTHENTICATION PROBLEM

This chapter presents a case study of face membership authentication (FMA). To evaluate the class factorisation ability of DE-CVM (Y. Chen, Pang, & Kasabov, 2010), we study the FMA problem (Pang, Kim, & Bang, 2005) which is to distinguish the membership class from the non-membership in a total group through a binary class classification. FMA involves different levels of class overlapping which include the most discriminative class characteristics (Garcia, Alejo, Sinchez, Sotoca, & Mollineda, 2008) because class overlapping increases while the size of the membership group is close to the size of non-membership group. The size of the membership group can be dynamically changed which makes class characteristic of membership and non-membership manually adjustable. The smaller the size of membership group, the less discriminative class characteristics are involved.

Section 9.1 explains the background of face membership authentication, Section 9.2 presents dataset description and data pre-processing. The experiment results for DE-CVM are presented in Section 9.3. Section 9.4 gives the conclusion.

9.1 BACKGROUND

Membership authentication is a typical problem in digital security schemes (Pang, Kim, & Bang, 2003). The problem can be generally depicted as follows. Consider a certain group of people G with N members, which is the universal set. If there exists an arbitrary subgroup M such that $M \subset G$ and $|M| < N/2$, then it is a membership group, and the remaining people $\bar{M} = G - M$ make a non-membership group. Thus, the objective of membership authentication is to distinguish the membership M class from the non-membership class $\bar{M}$ in the group. Since anonymity is an essential feature of digital security schemes (M.-H. Yang, Ahuja, & Kriegman, 2000), this would require membership authentication to allow changing dynamically the size of the membership group and the members in that group. Therefore, unlike all previous types of works on face recognition for security, where identification of a given face image is needed, dynamic membership authentication requires to authenticate an individuals membership without revealing the individuals identity and without restricting the group size and/or the members of the group. For example, for security or access control, the permission such as the right to enter an important building is often assigned to many individuals. To get the permission, it is required that members of the group be distinguished from nonmembers, while the members need not be distinguished from one another due to privacy concerns (Schechter, Parnell, & Hartemink, 1999).

For dynamic face membership authentication, Xie, Xu, and Hundt (2001) introduced a verification system, in which they authenticated face memberships by combining a face recognition method using template matching and a face verification using single support vector machine (SVM) classifier. Since the system is not entirely independent from face identification, it is not a real dynamic membership authentication system according to our definition. Pang et al. (2003) introduced an SVM ensemble method for membership authentication in dynamic face groups. It was a novel membership authentication method for two main reasons. First, it performed the membership authentication in terms of binary classification without revealing the individuals identity, *i.e.* it was only concerned with whether a member was included in a membership group or not. A powerful SVM ensemble combining several binary SVM classifiers was introduced for supporting this property. Second, it performed dynamic membership authentication without restricting the group size and/or the members of the group, *i.e.* the membership authentication environments could be changing dynamically. An effective face representation using an eigenfeature fusing technique was introduced to support this requirement.

However, Pang et al. (2003) also found that the SVM ensemble method could only remain stable for a membership group whose size is less than 45 persons (16.6% of total group). As the membership group size increases, its membership authentication performance becomes poorer and very unstable. Furthermore, when the size of the membership group becomes similar to the size of the non-membership group, it is almost impossible to obtain a satisfactory authentication performance. This is due to a complicated mixed data distribution among the membership and non-membership face images, as it is very difficult to discriminate such data in terms of only one classifier even if its classification performance is powerful.

9.2 IMAGE DATA PRE-PROCESSING

In pre-processing, a data splitting procedure divides the data iteratively, and a number of CVM classifiers follow each step of the data partition. Here, by using locally linear embedding (LLE) (Pang et al., 2005) dimensionality reduction theory, we compressed member faces and nonmember faces as a set of LLE eigenfaces which together characterise the variation between all the member face images and nonmember face images, respectively. we partitioned the training set by a membership-based clustering with the membership eigenfaces and non-membership eigenfaces as two cluster centres.

9.2.1 *LLE Eigenface*

Compared with the linear dimensionality reduction method PCA (Pang et al., 2003), LLE is a method for nonlinear dimensionality reduction introduced by Roweis and Saul (2000). This method recovers global nonlinear structure from locally linear fits. It attempts to preserve as much as possible the local neighbourhood of each object, while preserving the global distances to the rest of the objects. These properties of the method might not be of benefit for data classification, but they definitely imply a better clustering of the data. Here, we represent the whole set of face images in the membership group (or non-membership group) as a set of membership (or non-membership) eigenfaces, which we obtained from the LLE eigenface technique as explained below.

A given face feature dataset consists of N real-valued vectors $\mathbf{x}$. Each vector $\mathbf{x}$ is a high-dimensional vector with dimensionality D, and $\mathbf{y}_i$ is the low-dimensional vector embedded in $\mathbf{x}_i$ with embedding dimensionality d, where $D >> d$. The computation of LLE eigenface involves an optimal embedding procedure that reduces a vector from high-dimensional data $\mathbf{x}$ to low-dimensional data $\mathbf{y}_i$ by minimising the following cost function:

$$\Phi(y) = \sum_i \left| \mathbf{y}_i - \sum_j W_{ij}\mathbf{y}_j \right|^2 . \tag{9.1}$$

This procedure consists of three steps:

Step 1. Compute the neighbours of each data point $\mathbf{x}_i$ by computing pairwise distances and finding neighbours. In the simplest formulation of LLE, one can identify K nearest neighbours per data point by measuring the Euclidean distance.

Step 2. Compute the weights W_{ij} that best reconstruct each data point $\mathbf{x}_i$ from its neighbours, minimising the following cost function by constrained linear fits:

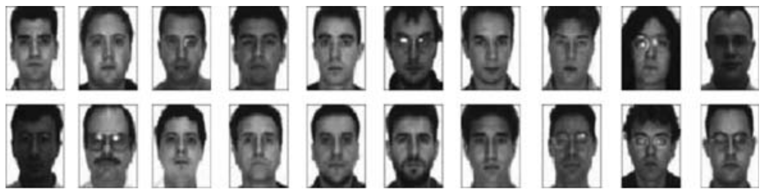
$$\epsilon(W) = \sum_i \left| \mathbf{x}_i - \sum_j W_{ij}\mathbf{x}_j \right|^2 . \tag{9.2}$$

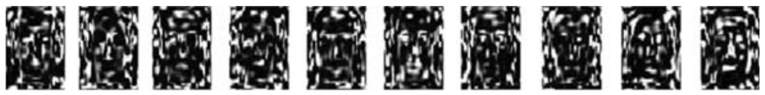
Step 3. Compute the matrix M in terms of the previous weights computation

$$M = (I - W)^T(I - W). \tag{9.3}$$

where I is the $N \times N$ identity matrix.

Note that the bottom $d + 1$ eigenvectors of the matrix M are corresponding to its smallest $d + 1$ eigenvalues. Thus, for fixed weights vectors W_{ij}, the embedding vectors $\mathbf{y}_i$ are found by minimising the cost function in equation 9.1. That is, the optimal embedding can be found by computing the bottom


(a) Original face images.


(b) LLE eigenfaces.

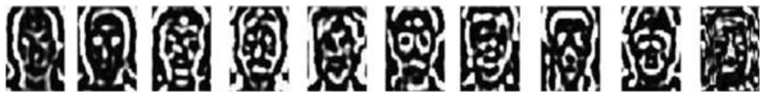

(c) PCA eigenfaces.

Figure 9.1: Comparison of membership LLE eigenfaces and PCA eigenfaces when $K = 10$ and $M = 20$.

$d + 1$ eigenvector of the matrix and M. d/D identifies the compression ratio of the embedded data space to the original data space. A bigger ratio means that more local points of data variations in the original space are kept in the embedded eigenspace, and a smaller ratio means that more global information are reserved by each LLE eigenvector of the embedded space. To reserve the local face variations, d/D is constantly set as $1/3$ in our experiments.

Therefore, for face image data, LLE eigenfaces are a subset of eigenvectors of matrix M, which assumes that a facial image x from training set $\{\mathbf{x}_i\}_{i=1}^N$ can be reconstructed from its neighbours with the lowest reconstruction error in equation 9.2.

Figure 9.1b shows 10 LLE eigenfaces derived from the 20 face images of Figure 9.1a, and which are included in the membership group for authentication. Figure 9.1c shows 10 PCA eigenfaces derived from the same 20 face images of Figure 9.1a. As we can see, each PCA eigenface contains the global components from all 20 faces, which look obviously very different from one another; while the differences between LLE eigenfaces are not very distinct, because each LLE eigenface is obtained from a locally embedding computation, that contains only the local information of 20 face images.

9.2.2 *Data Partition*

In membership authentication, the group members can be divided into membership and non-membership group members as $G = M \cup \bar{M}$. Applying the previous LLE eigenface technique to M and $\bar{M}$ respectively, we obtain two representative eigenface sets such that the membership eignefaces $\mathbf{U}_M = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K]$ and the non-membership eigenfaces $\mathbf{U}_{\bar{M}} = [\bar{\mathbf{u_1}}, \bar{\mathbf{u_2}}, \ldots, \bar{\mathbf{u_K}}]$. They characterise the "membership-face" and "non-membership-face" respectively. Figure 9.1 is an example of membership LLE eigenfaces with membership size equal to 20.

On partitioning, we identify the two partitioned groups as a $2 \times n$ binary matrix **V**, where the element is v_{ij} if the j^th data point $\mathbf{x}_j$ belongs to group i, otherwise it is 0. Once the two cluster centers $\mathbf{U}_M$ and $\bar{\mathbf{U}}_M$ are fixed, then the clustering based on membership can be performed as follows:

$$v_{ij} = \begin{cases} 1 & \text{if } \min_{l=1}^{K} \mathbf{x}_j \cdot \mathbf{u}_l \leq \min_{k=1}^{K} \mathbf{x}_j \cdot \bar{\mathbf{u}}_k \\ 0 & \text{otherwise} \end{cases}$$

Where $\min_{l=1}^{K} \mathbf{x} \cdot \mathbf{u}_l$ is the minimum distance projected onto the membership eigenfaces ($\mathbf{U}_M$), and $\min_{k=1}^{K} \mathbf{x} \cdot \bar{\mathbf{u}}_k$ is the minimum distance projected onto the non-membership eigenfaces $(\bar{\mathbf{U}}_M)$.

Figure 9.2 illustrates an example of binary LLE data partition, where two dotted lines represent the two principal membership LLE eigenvectors (or eigenfaces) and the two solid lines represent two principal non-membership LLE eigenvectors (or eigenfaces). Each group member $\mathbf{x}_i$ in the space is projected to those eigenvectors, and is assigned to a class whose projected distance sum is the smallest. After completing the assignment of all group members, we obtain two disjoint subgroups that correspond to the membership group and the non-membership group, respectively. Each subgroup will be used as two labeled class datasets to train the CVM classifier.

We modified the LLE technique to take into consideration the labeling (membership and non-membership) of the data. Thus, as the evolving clusters are updating, members and nonmembers are forced to gather in numbers of separated sub-clusters by a recursive membership-based LLE clustering.

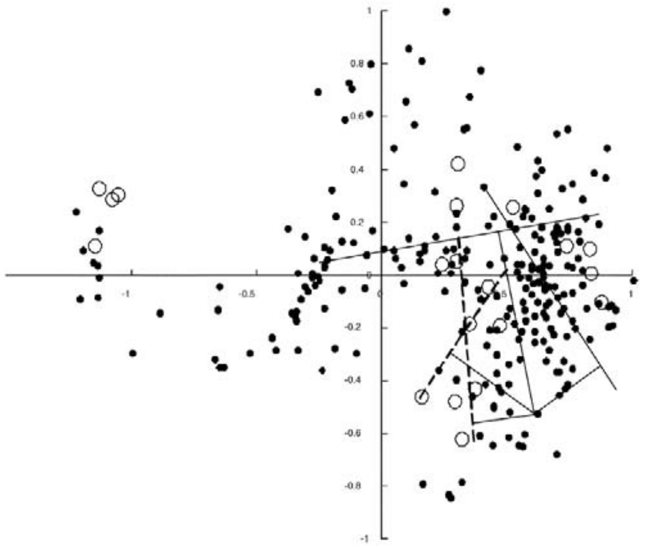


Figure 9.2: Typical example of the binary LLE data partition.

9.3 EXPERIMENT RESULTS AND DISCUSSION

FMA is performed on MPEG-7 face dataset, which consists of 1355 face images of 271 persons (5 different face images per person are taken), where each image has the size of 56×46 pixels. The images have been selected from AR(Purdue), AT&T, Yale, UMIST, University of Berne, and some face images obtained from MPEG-7 news videos (M. S. Kim, Kim, & Lee, 2003). We set the membership size ranging from 35 to 230 with a 10 persons interval to achieve datasets with dynamic class characteristics, and compared the proposed DE-CVM with CVM under the condition of dynamic distinctive class characteristics in FMA.

Methods	CVM		DE-CVM	
Group Size	True-positive	Membership CVs	True-positive	Membership CVs
35	$\mathbf{88.57\%(=\frac{31}{35})}$	36	$\mathbf{88.57\%(=\frac{31}{35})}$	$\mathbf{14}$
55	$85.78\%(=\frac{47}{55})$	37	$\mathbf{89.09\%(=\frac{49}{55})}$	$\mathbf{17}$
75	$84.00\%(=\frac{63}{75})$	40	$\mathbf{90.66\%(=\frac{68}{75})}$	$\mathbf{20}$
95	$83.15\%(=\frac{79}{95})$	41	$\mathbf{85.26\%(=\frac{81}{95})}$	$\mathbf{26}$
115	$81.74\%(=\frac{94}{115})$	39	$\mathbf{91.30\%(=\frac{105}{115})}$	$\mathbf{30}$
135	$79.25\%(=\frac{107}{135})$	38	$\mathbf{88.88\%(=\frac{120}{135})}$	$\mathbf{33}$

Table 9.1: *Classification results of the membership authentication*

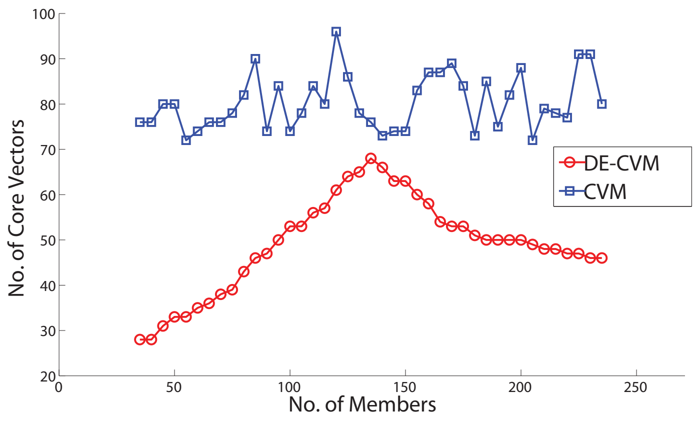


Figure 9.3: A comparison of CVM and DE-CVM on classification accuracy under the condition of different membership group size

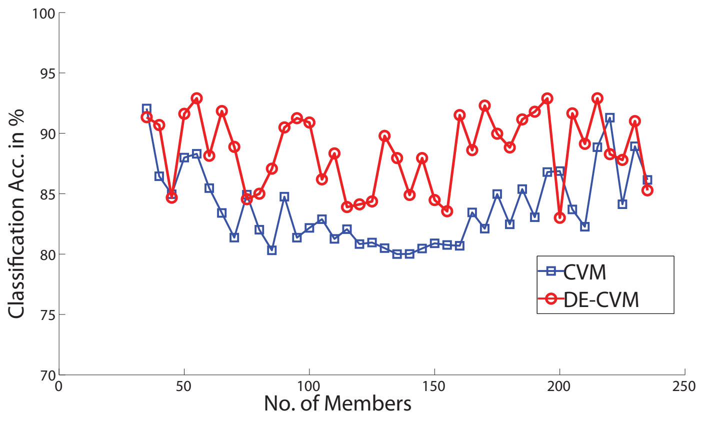


Figure 9.4: A comparison of CVM and DE-CVM on number of core vectors under the condition of different membership group size

Table 9.1 summarises the comparison result of membership authentication true-positive rate, and the number of core vectors between CVM and DE-CVM when the group size varies from 35 to 135 with a 20 interval, respectively. From Table 9.1, it can be observed that DE-CVM provides a better performance on membership authentication by obtaining higher true positive rates using fewer core vectors than CVM with all different group size. In addition, when the group size increases the true-positive rate of CVM decreases remarkably, while that of DE-CVM is always around 88% regardless of the group size.

Fig. 9.3 and Fig. 9.4 illustrate the number of core vectors and classification accuracy for CVM and DE-CVM under the condition of different membership group size. As seen in Fig. 9.3, the number of core vectors from CVM stays constantly around 80, while the membership group size changes between 35 and 135 (equals to 50% of total group size). However, although the number of core vectors from DE-CVM has a spiking increase, it never goes above the number of core vectors from CVM. On the other hand, Fig. 9.4 shows that generally DE-CVM achieves higher FMA accuracy than CVM, and the difference becomes as significant as 8% when the member group size ranges between 50 and 135. Recall that the number of DE-CVM core vectors is always smaller than the number of CVM core vectors, which indicates that DE-CVM core vectors are more discriminative than CVM for FMA. In other words, DE-CVM is more capable than CVM of factorising the class characteristics of membership and non-membership as fewer DE-CVM core vectors often deliver better FMA.

9.4 SUMMARY

This chapter presents a case study for dynamic face membership authentication. The results show that compared to the global modelling method CVM, local modelling method DE-CVM factorises class characteristics by reducing the sparseness area, discriminating core vectors on class interaction hyperplanes, and enabling outliers detection.

Next chapter presents a case study for string format stream data mining.

Chapter 10

THE APPLICATION OF META LEARNING STRING KERNEL MEBS SYSTEM TO SOLVE REAL WORLD STRING FORMAT STREAM DATA MODELING

The chapter presents a case study of modelling real world string format stream data that is carried out to demonstrate the effectiveness of the proposed meta learning algorithm for string kernel MEB optimisation. Three string kernel MEBs were experimented namely: edit-distance, bag-of-words and n-gram on four string datasets: Spam, Reuters-21578, Network Application Detection and e-News categorisation.

Section 10.1 explains the experimental setup and the evaluation criteria, section 10.2 describes the datasets description and the data pre-processing. The experimental results for the three string kernel SVMs are presented in section 10.3. Section 10.4 indicates the limitation of the method found from the case study and how to improve it in the future. Finally, section 10.5 gives the conclusion.

10.1 EXPERIMENT SETUP

The proposed algorithm was experimented on three string kernel MEBs (*edit-distance* MEB, *bag-of-words* MEB and *n-gram* MEB). As shown in Table 10.1, the algorithm was trained using training string dataset pool L_{TR}, and was tested on testing string dataset pool L_{TS}, for each string kernel MEB. In the experiments, the MEB cost parameter (c) was selected as $2^0, 2^1, ..., 2^{16}$ for all string kernel MEBs. The string kernel parameters λ_L and λ_B for edit-distance and bag-of-words string kernels, were selected as $0.001, 0.1, 0.25, 0.5$. The substring length in n-gram string kernel was selected as $1, 2, ..., 8$ in the experiments. The string meta-feature, $AvgMinTokenVal$ was not considered in the training stage, as it was having the value 0 for all datasets. Also, the global threshold for the $AvgTokenThr$ was set to 2 in all the experiments. Support Vector Regression (SVR) was used to build the meta model. In the training stage, the parameters which yield lowest cross validation RMSE for SVR, were considered in regression (in building the meta model), for each string kernel MEB. 10 fold- cross validation was done for the top 10 predicted parameter combinations, on each string dataset. The performance evaluation was done considering Root Mean Squared Error (RMSE) for the top 10 predicted parameters on each dataset.

Dataset Pool	String Dataset (Dataset Label)	Sting Kernel MEB		
		Edit-Distance	Bag-of-Words	N-gram
L_{TR}	Spam	✓	✓	✓
	Reuters-21578	✓	✓	✓
	Network Application Detection	✓	✓	✓
	e-News Categorisation	-	-	✓
L_{TS}	Spam(1)	✓	✓	✓
	Reuters-21578(2)	✓	✓	✓
	Network Application Detection(3)	✓	✓	✓
	e-News Categorisation(4)	-	-	✓

Table 10.1: *The string datasets used in training and testing the proposed algorithm*

In the experiments, edit-distance and bag-of-words string kernels were implemented using LibMEB-2.9 (Chang & Lin, 2001) and n-gram string kernel was implemented using shogun octave interface (S. Sonnenburg, 2006). The string meta-feature computation program was coded using C++ language. All the experiments were run on a PC having Intel Core2 Duo 3GHz processor and 2.96 Gb RAM.

10.2 DATASETS AND DATA PRE-PROCESSING

Four string datasets were used in the string dataset pool $L = \{$*Spam*, *Reuters-21578*, *Network Application Detection*, *e-News Categorisation*$\}$. It was again subdivided into training dataset pool (L_{TR}) and testing dataset pool (L_{TS}), where each consisted of unique string datasets. String dataset pool L_{TR} was used to train the meta model in the proposed algorithm and string datasets pool L_{TS} was used to test the proposed algorithm. For edit-distance and bag-of-words string kernel MEBs, training dataset pool L_{TR} = {*Spam*, *Reuters-21578*, *Network Application Detection*} was used, and for n-gram MEB, L_{TR} = { *Spam, Reuters-21578, Network Application Detection, e-News Categorisation* } was used. In testing, for edit-distance and bag-of-words string kernel MEBs, testing pool L_{TS} = {*Spam*, *Reuters-21578*, *Network Application Detection*} was used, and for n-gram MEB, the testing pool L_{TR} = { *Spam*, *Reuters-21578*, *Network Application Detection*, *e-News Categorisation* } was used. Table 10.1 summarises algorithm training and testing information on each string dataset. A detailed description about each string dataset is given below.

10.2.1 *Spam Dataset*

This dataset consists of 696 ham messages and 384 Spam messages from (*Spam Assassin public mail corpus*, 2002). There are two types of ham e-mails: easy ham (646) and hard ham (50). Easy ham e-mails are non-Spam messages without any Spam signatures and hard ham are non-Spam messages similar in many aspects to Spam messages which use unusual HTML markup, coloured text, Spam-sounding phrases, etc. Each e-mail message has a header, a body and some potential attachments. The training dataset consists with 810 messages (484 easy ham, 38 hard ham and 288 Spam) and testing dataset has 270 messages (162 easy ham, 12 hard ham and 96 Spam).

10.2.2 *Reuters-21578 Dataset*

The Reuters dataset used in the experiments has the exact split to (Lodhi et al., 2002). It consists of 470 documents: 380 for training and 90 for testing. Four document categories, those of earn, acquisition, crude and corn are available in the dataset. Table 10.2 shows the document distribution among the different categories.

Class Label	Document Category	Training	Testing
0	Earn	152	40
1	Acquisition	114	25
2	Crude	76	15
3	Corn	38	10
		370	90

Table 10.2: *Data distribution-reuters-21578 dataset*

10.2.3 *Network Application Detection Data*

The dataset consists of network traffic data produced by network applications, such as http, https, imap, pop3, ssh, ftp and bittorent. All network data were captured, and sorted according to their protocols using "Wireshark" (Combs, 2008) and split into individual connections using *tcpflow* (Elson, 2003). Only TCP traffic was taken into account in the data capturing stage. The option '-s' in "tcpflow" (Elson, 2003) was used to remove all non printable characters in a connection. Also, only the first 50 bytes of a connection were considered in preparing the dataset. Every connection was labelled according to the application type. Table 10.3 shows the label number for every application type and the number of instances in training and testing datasets.

Class Label	Application/protocol	Training	Testing
0	AIM	18	7
1	Bittorrent	140	59
2	http	583	249
3	pop	17	7
		770	329

Table 10.3: *Data distribution-application detection dataset*

10.2.4 *e-News Categorisation Data*

The dataset is collected from four electronic newspapers: (*New Zealand Herald*, 2010), (*The Australian*, 2010), (*The Independent*, 2010) and (*The Times*, 2010), on five news topics (business, education, entertainments, sport and travel). Each document is labelled manually by skimming over the text to identify the category. Punctuations and stop words were removed from the dataset in advance. Table 10.4 shows detailed information about the dataset.

Class Lable	News Category	Training	Testing
0	Business	227	97
1	Education	93	40
2	Entertainments	99	42
3	Sport	118	50
4	Travel	131	56
		668	285

Table 10.4: *Data distribution e-Newsgroup dataset*

10.3 RESULTS

The result of the experiment for three string kernel MEBs are discussed in this section. Section 10.3.1 explains the results for *edit-distance* MEB optimisation, using the proposed algorithm. The experimental results for *bag-of-words* MEB optimisation are presented in section 10.3.2. Section 10.3.3 explains the results for *n-gram* MEB optimisation, using the proposed algorithm.

10.3.1 *Meta Learning for Edit-Distance MEB Optimisation*

Here, the proposed algorithm was used to optimise edit-distance MEB. The algorithm attempts to find the optimum parameter combination (λ_L and MEB cost) for *edit-distance* MEB on three string datasets: Spam, Reuters-21578 and network application detection, in test dataset pool L_{TS} (refer Table 10.1). In the experiments, the SVR parameters: $\gamma = 0.084$ and *SVR Cost=5400* were used in regression. The actual accuracy and the predicted accuracy for the top 10 predicted parameter combinations are shown in Table 10.5. Also, the table shows the RMSE for top 10 predicted parameter combinations on each dataset.

According to Table 10.5a and Table 10.5b, the optimum parameters produced by the proposed algorithm yield very low predicted and actual classification accuracies, for *edit-distance* MEB, on Spam and Reuters-21578 string datasets. This shows that the *edit-distance* MEB is not suitable for string classification on Spam and reuters-21578 datasets. However, according to Table 10.5c, the algorithm produces optimum parameters which yield good string classification accuracies on network application detection dataset, for *edit-distance* MEB (with a low RMSE).

cost	λ_L	rank	predicted%	actual%
65536	0.000488	1	2.36387	0.37
2	0.000488	2	2.36381	5.19
4	0.000488	3	2.36377	4.44
8	0.000488	4	2.36371	3.33
16	0.000488	5	2.36357	1.11
32	0.000488	6	2.36331	0.37
2	0.000976	7	2.36302	4.44
4	0.000976	8	2.36299	3.33
8	0.000976	8	2.36292	1.11
65536	0.000976	10	2.36286	0.37
root mean squared error				1.831455

(a) Spam Data

cost	λ_L	rank	predicted%	actual%
65536	0.000488	1	3.30463	23.33
2	0.000488	2	3.30454	24.44
4	0.000488	3	3.30451	26.67
8	0.000488	4	3.30445	23.33
16	0.000488	5	3.30432	23.33
32	0.000488	6	3.30407	23.33
2	0.000976	7	3.3038	26.67
65536	0.000976	8	3.30379	23.33
4	0.000976	9	3.30377	23.33
8	0.000976	10	3.30371	23.33
root mean squared error				20.846794

(b) Reuters-21578

cost	λ_L	rank	predicted%	actual%
32768	0.0625	1	73.1729	75.99
32768	0.03125	2	73.1653	75.99
32768	0.125	3	73.1602	75.99
32768	0.015625	4	73.1556	75.99
8192	0.125	5	73.1526	75.99
16384	0.125	6	73.1526	75.99
8192	0.25	7	73.1518	75.99
32768	0.5	8	73.1515	75.99
4096	0.25	9	73.1514	75.99
65536	0.25	10	73.1512	75.99
root mean squared error				2.833499

(c) Network Application Detection

Table 10.5: *Experimental results for Edit-Distance MEB optimisation: (the top 10 predicted parameter combinations using the proposed algorithm on each string dataset)*

10.3.2 *Meta Learning for Bag-of-Words MEB Optimisation*

Here, the the proposed algorithm attempts to find the optimum parameter combination (λ_B and MEB cost) for *bag-of-words* MEB. Initially, the algorithm was trained on a training dataset pool L_{TR} (refer Table 10.1). Then, the algorithm predicted the string classification accuracies for *bag-of-words* MEB on three different string datasets in test dataset pool L_{TS} (refer Table 10.1). The SVR parameters: $\gamma = 0.88$ and *SVR Cost=450* were used in regression. 10-fold cross validation was done for the top 10 predicted parameter combinations, and RMSE was calculated for the same. Table 10.6 shows the top 10 predicted parameter combinations on three string datasets for *bag-of-words* MEB.

According to Table 10.6b and Table 10.6c, the proposed algorithm produces parameter combinations which yield high classification accuracies on reuters-21578 and network application detection datasets with a very low RMSE. However, on Spam dataset, the optimised parameter combinations produced by the proposed algorithm yield average string classification accuracies (see Table 10.6a). Considering the overall high string classification accuracy (with a very low average RMSE) shown in Table 10.6 and Table 10.8, on all three datasets, one can say that the proposed algorithm produces optimised parameter combinations which yield good string classification accuracies, for *bag-of-words* MEB.

10.3.3 *Meta Learning for N-gram MEB Optimisation*

In this experiment, the algorithm attempts to find optimised parameters (substring length and MEB cost) for *n-gram* MEB. The algorithm was trained on string dataset pool L_{TR} and tested on testing string dataset pool L_{TS} (refer Table 10.1). The SVR parameters: $\gamma = 0.95$ and *SVR Cost=500* were used in regression. 10-fold cross validation was done for the top 10 predicted parameter combinations on each string dataset. Table 10.7 summarises the experiment results for the top 10 predicted combinations on each dataset.

cost	λ_B	rank	predicted%	actual%
2	0.000488	1	61.6244	64.44
4	0.000488	2	61.6207	64.44
8	0.000488	3	61.6135	64.44
16	0.000488	4	61.5989	64.44
32	0.000488	5	61.5699	64.44
64	0.000488	6	61.5117	64.44
2	0.000976	7	61.4788	64.44
4	0.000976	8	61.4752	64.44
8	0.000976	9	61.4679	64.44
16	0.000976	10	61.4534	64.44
root mean squared error				2.899333

(a) Spam Data

cost	λ_B	rank	predicted%	actual%
4096	0.5	1	87.9839	86.67
2048	0.5	2	87.9734	86.67
1024	0.5	3	87.9551	86.67
512	0.5	4	87.9422	86.67
256	0.5	5	87.9347	86.67
8192	0.5	6	87.9307	86.67
128	0.5	7	87.9307	86.67
32768	0.5	8	87.9304	86.67
64	0.5	9	87.9287	86.67
32	0.5	10	87.9276	86.67
root mean squared error				1.273886

(b) Reuters-21578

cost	λ_B	rank	predicted%	actual%
4	0.000488	1	75.3987	75.99
2	0.000488	2	75.3987	75.99
8	0.000488	3	75.3986	75.99
16	0.000488	4	75.3985	75.99
32	0.000488	5	75.3983	75.99
64	0.000488	6	75.3978	75.99
128	0.000488	7	75.3968	75.99
4	0.000976	8	75.3967	75.99
2	0.000976	9	75.3967	75.99
8	0.000976	10	75.3966	75.99
root mean squared error				0.592261

(c) Network Application Detection

Table 10.6: *Experimental results for Bag-of-Words MEB optimisation: (the top 10 predicted parameter combinations using the proposed algorithm on each string dataset)*

According to Table 10.7, the proposed algorithm produces optimised parameters, which yield good string classification accuracies for *n-gram* MEB, on all four string datasets. The algorithm has a very low RMSE for top 10 predicted on Spam, Reuters-21578 and network application detection datasets (see Table 10.7a, Table 10.7b, and Table 10.7c). Even though, the algorithm has quite a high RMSE on the e-News categorisation dataset, the top 10 predicted parameter combinations yield good string classification accuracies on the dataset (see Table 10.7d).

10.4 LIMITATIONS

The limitations that identified in the research are discussed below. Firstly, for n-gram kernel, the proposed algorithm yields classification accuracies over 100% on Spam and Reuters-21578 string datasets. This can be resolved if one is able to set upper and lower bounds in the algorithm. Secondly, the string meta-features could be more descriptive, since they can explain the string dataset more clearly than just numbers representing the string dataset. Thirdly, while

cost	substring length	rank	predicted%	actual%
16384	8	1	99.3074	98.33333
4096	8	2	99.2982	98.33333
32768	7	3	99.2683	98.33333
16384	7	4	99.2652	98.33333
4096	7	5	99.2628	98.33333
4096	6	6	99.2524	98.33333
4096	5	7	99.2066	98.33333
2048	8	8	99.1935	98.33333
32768	6	9	99.1862	98.33333
4096	2	10	99.1803	97.81251
root mean squared error				0.971167898

(a) Spam Data

cost	substring length	rank	predicted%	actual%
4096	6	1	92.6961	95.36587
2048	6	2	92.6842	95.36587
8192	6	3	92.6771	95.36587
1024	6	4	92.6636	95.36587
512	6	5	92.6474	95.36587
16384	6	6	92.6405	95.36587
256	6	7	92.6376	95.36587
128	6	8	92.6322	95.36587
64	6	9	92.6294	95.36587
32	6	10	92.6279	95.36587
root mean squared error				2.712372587

(b) Reuters-21578

cost	substring length	rank	predicted%	actual%
16384	2	1	99.6656	98.22917
2048	2	2	99.5895	98.22917
1024	2	3	99.5776	98.22917
32768	2	4	99.5685	98.22917
4096	2	5	99.5634	98.22917
512	2	6	99.5607	98.22917
256	2	7	99.5489	98.22917
128	2	8	99.542	98.22917
64	2	9	99.5384	98.22917
32	2	10	99.5365	97.81252
root mean squared error				1.386738588

(c) Network Application Detection

cost	substring length	rank	predicted%	actual%
4096	5	1	90.8189	74.35295
4096	6	2	90.6225	75.05881
4096	4	3	90.5267	73.76471
8192	5	4	90.484	73.88235
32768	5	5	90.3336	73.64706
16384	5	6	90.3317	73.64706
8192	6	7	90.3088	75.17647
4096	3	8	90.2765	74.70588
8192	4	9	90.275	73.41176
32768	4	10	90.1778	73.17646
root mean squared error				16.34502652

(d) e-News Categorisation

Table 10.7: *Experimental results for N-gram MEB optimisation: (the top 10 predicted parameter combinations using the proposed algorithm on each string dataset)*

String Kernel	Dataset	RMSE	Avg RMSE
Edit-Distance	Spam	1.831455	8.503916
	Reuters-21578	20.846794	
	Network Application Detection	2.833499	
Bag-of-Words	Spam	2.899333	1.588493
	Reuters-21578	1.273886	
	Network Application Detection	0.592261	
N-gram	Spam	0.971168	5.353826
	Reuters-21578	2.712373	
	Network Application Detection	1.386739	
	e-News Categorisation	16.345027	

Table 10.8: *Root Mean Squared Error (RMSE) for string dernel MEB optimisation on each string dataset (for top 10 predicted parameter combinations)*

a meta learning algorithm is usually tested on a large number of datasets, the high computational cost of string data and the unavailability of bench-mark string datasets, have forced this researcher to use only four string datasets in the experiments.

These limitations could be also caused by complex distribution of data samples. The proposed meta learning string kernels MEBs algorithm only learns the global model so far. In order to have a more accurate model for string data, a local modelling string kernel method needs to be developed in future.

10.5 SUMMARY

This case study shows that the proposed algorithm produces parameter combinations that yield good string classification accuracies on most of the datasets. They also reveal that some string kernel MEBs may not be suitable for carrying out the string classification required on certain string datasets. Specifically, *edit-distance* MEB yields poor string classification results on both Spam and Reuters-21578 string datasets.

The proposed method has three main contributions to the field of machine learning:

1. **String Meta-features**: The defined *string meta-features* can be used for extracting meta knowledge from any string dataset;

2. **Meta-Learning for String Classification Principle**: explains the procedure for applying meta-learning on string classification but using extracted meta-knowledge via *string meta-features*;

3. **Meta Learning Algorithm for String Kernel MEB Optimisation**: using the *Meta-Learning for String Classification Principle*, a novel string kernel optimisation method is derived, which is able to predict optimum string kernel MEB parameters for a given string kernel MEB on a string dataset by calculating relevant *string meta-features*.

Directions for future work and conclusion of the PhD thesis are given in the next chapter.

Chapter 11

CONCLUSION AND FUTURE DIRECTIONS

In this final chapter, the main achievements of the study are summarised. Several directions for future work are also discussed that would help in further improving the efficacy of the entire system.

The chapter is organised as follows: first, the main achievements of this thesis are summarised from three perspectives: in terms of basic on-line learning algorithm, evolving connectionist systems adopting, and string kernels learning. The section closes by stressing the original contributions of this work and giving a summary of the experimental results. The dissertation closes with a brief discussion of future directions in five possible areas.

11.1 SUMMARY OF ACHIEVEMENTS

This PhD study shows how evolving connection systems (i.e. local modelling) are adapted to stream data mining based on a fast, on-line kernel based learning algorithm. By means of local modelling, we can solve many real world stream data modelling issues such as dealing with large size, high dimensionality, skewed class distribution. The idea of focusing on the unique problem subspaces appears to be beneficial to the stream data modelling problems. It allows identification of sub-problems and allows their further study. In addition, this research developed novel string kernels method to overcome the difficulty of mining string format stream data. At the same time, I have also published 3 conference papers, submitted 1 journal paper, delivered 3 technical reports and developed 2 prototype systems.

11.1.1 *Fast, On-line Kernel Based Learning Algorithm*

We have developed a novel on-line kernel based learning method with low computational cost called on-line core vector machines (OCVM). This method inherits the characteristic of original CVM whose computational complexity is independent from the size of training samples. In addition, this method allows us to on-line analyse the stream data in high dimensional space to be analysed on-line which discriminates skewed class distribution.

The effectiveness of OCVM is evaluated by comparing its classification accuracy and train/testing time to those of other traditional methods on 10 benchmark datasets.

11.1.2 *Evolving Connectionist Systems on Stream Data Modeling*

Evolving connectionist systems (ECOS) are modular connectionist-based systems that evolve their structure and functionality in continuous, self-organised, on-line, adaptive, interactive way from incoming information; they can process both data and knowledge in supervised learning and/or unsupervised way (Kasabov & Song, 2002; Kasabov, 2002).

The proposed method HCVM learns local models from data through clustering of child-classes data and associating a set of local core vectors for each parent-class. DE-CVM is proposed and works similarly to DENFIS (see chapter 3) but in kernel spaces.

The advantages of adopting evolving connectionist systems are confirmed in two case studies: network intrusion detection and face membership authentication where small number of core vectors can provide higher classification accuracy.

11.1.3 *String Format Stream Data Learning*

A novel string classification method called Meta Learning String Kernels CVMs for string format stream data mining is proposed. Since one major task for stream data mining is to categorise texture, this method is an important part of an integrated stream data modelling system.

The capability of this method is verified in the experiments on several string classification tasks.

11.2 FUTURE DIRECTIONS

This section suggests some promising future directions for the development of the methods and systems in stream data modeling.

11.2.1 *Evolving Clustering Optimisation*

In our further work, we would like to address the optimisation problem of the total number of CVMs required for optimal solution. It can only be determined using cross validation method and by investigating on measuring the density of the MEB, and further exploiting new models based on ‘evolving clusters’.

11.2.2 *On-line methods for personalised modeling*

One of the issues with local and distance-based models is that they require good definition of the problem space in order to perform properly, which is often difficult and lacking in many problems (e.g. biological modeling) as they often involve a large number of noisy variables. In order to apply local or personalised models to this type of problem, an in-depth analysis of the features is necessary.

A personalised regression model with incremental feature selection will be proposed during next stage. This method applies incremental feature selection on variables that were ranked using univariate analysis and results from previous studies. This set of variables was then used to define the problem space and identify the relevant subset of data for each prediction. The global regression model is then optimised with this subset of training data to put a focus on the test input vectors residing in problem subspace.

11.2.3 *Spatial and Temporal Processing*

Compared to traditional processing, temporal contexts are particularly important in stream processing. Giannotti, Nanni, Pinelli, and Pedreschi (2007), presents a study on clustering trajectories of mobile objects (e.g., mobile phones). Problems faced in this spatio-temporal data mining task concern the identification of the proper spatial granularity level, the selection of the significant temporal sub-domains, the choice of the most promising clustering method, and the formalisation of the notion of (dis)similarity among trajectories. The authors recommend a density-based approach to trajectory clustering, and stress the importance of temporal focusing to isolate the clusters of higher quality. Both aspects have been tested on a data set automatically generated by a synthesiser of trajectory data.

Next stage we will use micro-clustering phase which is the on-line statistical data collection algorithm. This process is not dependent on any user input such as the time horizon or the required granularity of the clustering process. The aim is to maintain statistics at a sufficiently high level of (temporal and spatial) granularity so that it can be effectively used by both off-line and on-line analysis.

11.2.4 *Local modeling for String Kernels*

So far the proposed meta-learning string kernel CVMs only works on global modeling. Next stage we will aim on local modeling string kernels methods.

Appendix A

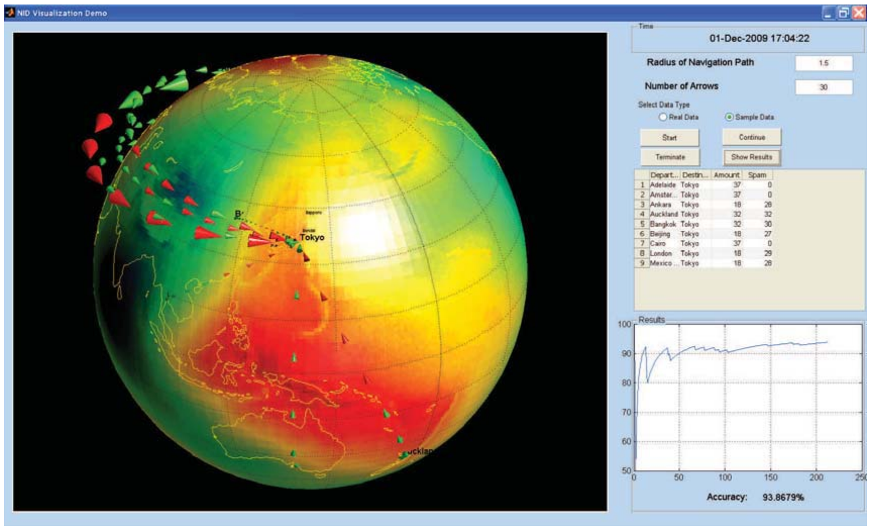
INTRUSION DETECTION VISUALISATION SYSTEM

Appendix A and B present two stream data visualisation systems developed based on our proposed HCVM algorithm (Y. Chen et al., 2009), respectively.

Network Intrusion Detection visualisation System (NIDVS) is developed to simulate the process of world-wide network intrusion detection. This visualisation system can monitor the Internet data stream. Moreover, this system is adaptive to various learning methods thus it provides the opportunity to compare the effectiveness of different methods when used on the same data.

A.1 DESIGN OVERVIEW

NIDVS simulates the process of world-wide network intrusion detection, where network traffic is visualized as data streaming from different source to the destinations, and intrusion filtered by the HCVM algorithm (Y. Chen et al., 2009). This NIDVS has two main functions: to monitor real-time network traffic and to demonstrate the effectiveness of the HCVM algorithm in terms of detection accuracy. NIDVS visualises the streaming data on a 3D world map, where network traffic is traced from the source to destination city (identified by latitude and longitude) for each data packet in transfer. A sequence of moving arrows represent network traffic as a set of data packets where data is indicated by green arrows, and spam/intrusion data is represented as red arrows. In NIDVS, network traffic data is associated with a certain speed to reflect the speed variation of real network traffic. For the convenience of observation, the height of traffic path can be modified to suit users sight.



Time
01-Dec-2009 17:04:22

Radius of Navigation Path: 1.5

Number of Arrows: 30

Select Data Type: ( ) Real Data [✓ Sample Data]

	Depart...	Destin...	Amount	Spam
1	Adelaide	Tokyo	37	0
2	Amster...	Tokyo	37	0
3	Ankara	Tokyo	18	28
4	Auckland	Tokyo	32	32
5	Bangkok	Tokyo	32	30
6	Beijing	Tokyo	18	27
7	Cairo	Tokyo	37	0
8	London	Tokyo	18	29
9	Mexico ...	Tokyo	18	28

Accuracy: 93.8679%

Figure A.1: The NIDVS visualisation demonstration shows the global data flow in the left, and the panels and result tables are displayed on the right side of the demo interface.

The NIDVS is shown in Figure A.1 Users can view the data flow and detection process on the left and control the view of data flow on the right side. The statistical results (*i.e.* total accuracy, number of intrusions from each source) are also shown on the right.

A.2 HARDWARE AND SOFTWARE REQUIREMENTS

The NIDVS requires Intel E8400@3.00 GHz or faster with at least 1.97 GB RAM. The video card should have over 256 MB RAM with support on Pixel Shader 3.0. The required software environment for NIDVS is MATLAB R2009a with Mapping toolbox 2.7.1, on the platform of Microsoft Windows XP Professional with Service Pack 3 or advanced. The best screen resolution is 1280 by 1024 pixels.

A.2.1 *System Architectural Design*

The system is structured as shown by the diagram in Figure A.2. It consists of five basic modules, namely ‘Data Acquiring’,‘Data visualisation’, ‘Intrusion Detection’, ‘Spam Collection’ and ‘Spam labelling’. The input data can be a set of example data or real Internet data. ‘Data Acquiring’ inputs data into the system, and ‘Data visualisation’ displays data as a set of arrows on the screen. The data can be labelled as ‘spam/intrusion’ or ‘normal’ by the ‘HCVM Intrusion Detection’ module. If the data is classified as spam/intrusion, then a detailed description of the spam will be recorded by the ‘Spam Collection’ module. The ‘Spam labelling’ module is in charge of presenting data as coloured arrows. Spam/intrusion data is presented with red colour, normal data with green colour. As seen in Figure A.2, the system runs in a loop that terminates when no data is input.

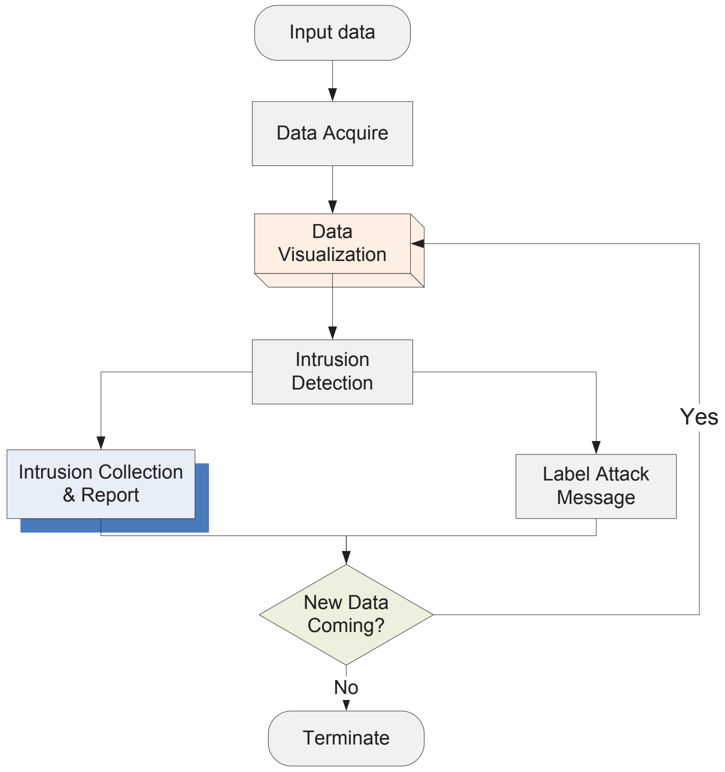


Figure A.2: The NIDVS system design diagram.

A.3 DETAILED DESCRIPTION OF COMPONENTS

The NIDVS consists of two components. The first component performs classification using the HCVM method. The second component visualizes data flow and spam detection on screen.

A.3.1 *HCVM in the NIDVS*

For labelling network intrusions as they state hierarchical multi-label structure, a high-speed hierarchical multi-label classification (HMC) algorithm, called hierarchical minimum enclosing balls (HCVM), is developed. We model the multi-label hierarchy into a data Hyper-Sphere constructed by numbers of MEBs. The MEBs separate, encompass and overlap with each other and form a tree structure representing the encoded multi-label hierarchy. When provided with an unlabelled sample, the HCVM seeks an MEB enclosing the sample, and multi-labels the sample according to the MEBs position in the hierarchy. The HCVM has been tested on a Gaussian synthetic data, the RCV1-V2 text data, and the KDD99 intrusion detection dataset for multi-label classification.

A.3.2 *visualisation*

The visualisation system provides a user interface for both novice and advanced users. It shows the flow of data between cities on the globe. During the data transfer, there is a spam/intrusion detection system processing data on each path. The spam/intrusion data will be detected and labelled with different colours. This visualises NID in real time, at the same time it tests the performance of the NID algorithm (*i.e.* the HCVM).

Users can replace the testing data with their own data. They simply need to replace the data files stored in 'root/Temp Data/or_data' with their own data files. The system will automatically allocate IPs, cities and network traffic speed to each data item. The allocation also selects data randomly from those data files and will never repeat them. Note that the input data is required to be formatted as a data matrix where the last column contains the class label.

The default NID method is the HCVM, although users can also change the detection algorithm. For example, users can choose to use SVM instead of HCVM for NID.

A.4 USER INTERFACE DESIGN

A.4.1 *Description of the User Interface*

The user interface consists of four components: Network Traffic visualisation, The NIDVS control panel, Network Intrusion Detection Report, NID Accuracy Tracking and they are described below.

Objects and Actions

1. Network Traffic visualisation

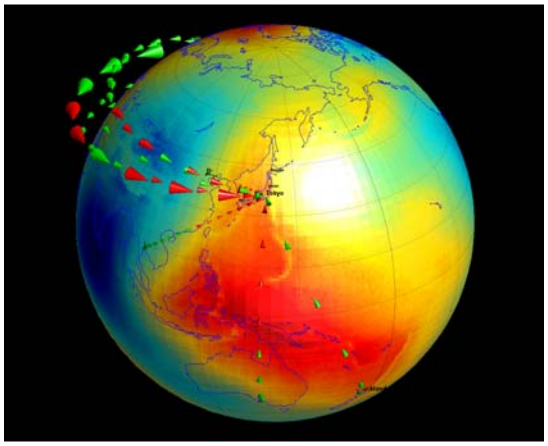


Figure A.3: The network traffic visualisation of NIDVS, in which the green arrows represent normal data and red arrows indicate spam/instrusion.

Figure A.3 depicts a network traffic visualisation of the NIDVS, where a sequence of arrows in colours represent a network data stream flowing from the source to target city. The green or red arrows represent normal or intrusion/spam data packet, respectively.

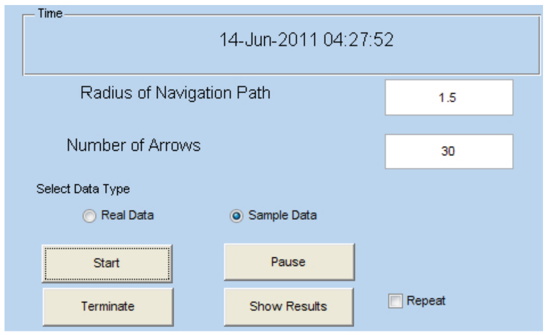


Figure A.4: The NIDVS control panel.

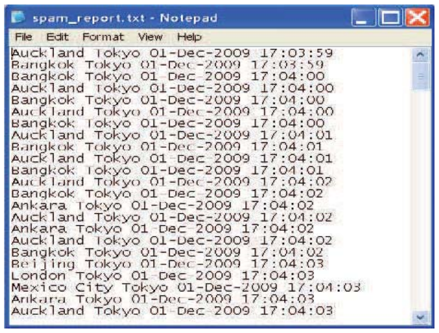
2. Control Panel

   The user is able to change the view of visualisation by using NIDVS control panel shown in Figure A.4. The 'Radius of Navigation Path' is the parameter controlling the height of the network traffic path. The 'Number of Arrows' affects the size of arrows. For a fixed source and target network traffic, a larger number of arrows result in relatively smaller size of the arrows.

   The four buttons control the NIDVS system: 'Start' initiates the visualisation demo, 'Pause/Continue' suspends or proceeds with the visualisation of data flow. 'Terminate' stops the visualisation demonstration and reports the present detected spam/intrusion list as shown in Figure A.5a The spam/intrusion list is stored meanwhile in a 'txt' file named 'spam report.txt', recording the details of every spam, its source and target city, and the time when the spam was detected. Button'Show results' displays a summary of NID performance in a pop-up window shown in Figure A.5b. This summarises the performance of the used NID algorithm, the number of normal data samples, spam sample, the accuracy of normal data detection (i.e. truth negative accuracy), the accuracy of spam intrusion data (i.e. truth positive accuracy) and the general accuracy.

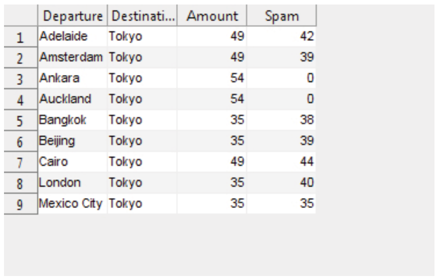
3. Network Intrusion Detection Report

   Figure A.6 depicts the statistical report of the current status of NID, which includes information about the number of processed data samples, the source and target city of the data package (reflected by the IP address), and the number of samples that are labelled as intrusions.



```
Auckland Tokyo 01-Dec-2009 17:03:59
Bangkok Tokyo 01-Dec-2009 17:03:59
Bangkok Tokyo 01-Dec-2009 17:04:00
Auckland Tokyo 01-Dec-2009 17:04:00
Bangkok Tokyo 01-Dec-2009 17:04:00
Auckland Tokyo 01-Dec-2009 17:04:00
Bangkok Tokyo 01-Dec-2009 17:04:00
Auckland Tokyo 01-Dec-2009 17:04:01
Bangkok Tokyo 01-Dec-2009 17:04:01
Auckland Tokyo 01-Dec-2009 17:04:01
Bangkok Tokyo 01-Dec-2009 17:04:01
Auckland Tokyo 01-Dec-2009 17:04:02
Bangkok Tokyo 01-Dec-2009 17:04:02
Ankara Tokyo 01-Dec-2009 17:04:02
Auckland Tokyo 01-Dec-2009 17:04:02
Ankara Tokyo 01-Dec-2009 17:04:02
Auckland Tokyo 01-Dec-2009 17:04:02
Bangkok Tokyo 01-Dec-2009 17:04:02
Beijing Tokyo 01-Dec-2009 17:04:03
London Tokyo 01-Dec-2009 17:04:03
Mexico City Tokyo 01-Dec-2009 17:04:03
Ankara Tokyo 01-Dec-2009 17:04:03
Auckland Tokyo 01-Dec-2009 17:04:03
```

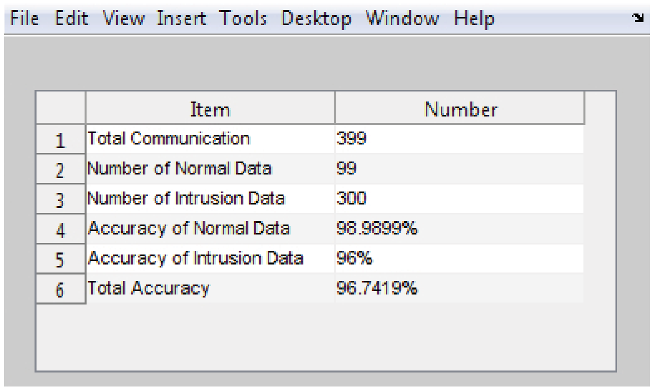
(a) The display of Spam list file



	Departure	Destinati...	Amount	Spam
1	Adelaide	Tokyo	49	42
2	Amsterdam	Tokyo	49	39
3	Ankara	Tokyo	54	0
4	Auckland	Tokyo	54	0
5	Bangkok	Tokyo	35	38
6	Beijing	Tokyo	35	39
7	Cairo	Tokyo	49	44
8	London	Tokyo	35	40
9	Mexico City	Tokyo	35	35

(b) The display of NID performance report

Figure A.5: Detection results and accuracy.


File Edit View Insert Tools Desktop Window Help

	Item	Number
1	Total Communication	399
2	Number of Normal Data	99
3	Number of Intrusion Data	300
4	Accuracy of Normal Data	98.9899%
5	Accuracy of Intrusion Data	96%
6	Total Accuracy	96.7419%

Figure A.6: NID report including information about the source and target city of data package, the number of data packages passed by, and the number of spam/intrusion detected.

4. NID Accuracy Tracking

   Figure A.7 presents a real time intrusion detection report in graphical form. The performance of intrusion detection by HCVM varies over time. The final detection accuracy is fixed until all data runs out.

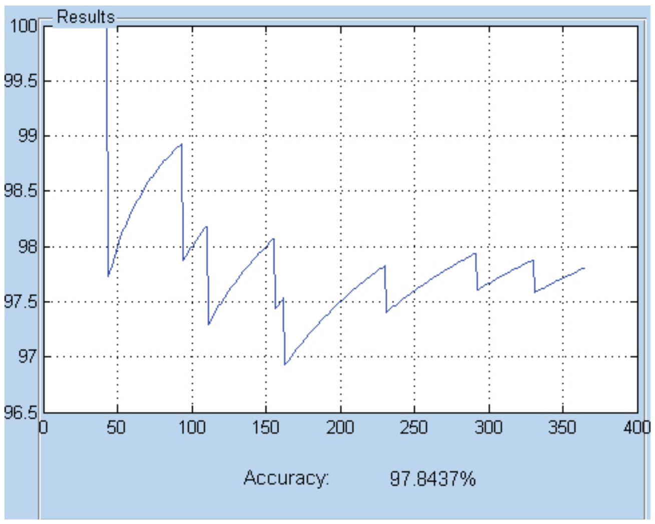


Figure A.7: Graph for real time intrusion detection report, where the $x - axis$ represents the amount of data processed, and $y - axis$ represents the detection accuracy in presents

Appendix B

HCVM PROTOTYPE SYSTEM

The HCVM prototype system demonstrates the learning progress of HCVM which gives a better understanding of on-line learning process. This system also provides the comparison of globe modelling and local modelling. User can evaluate the advantage of using local modelling easily from the visualisation.

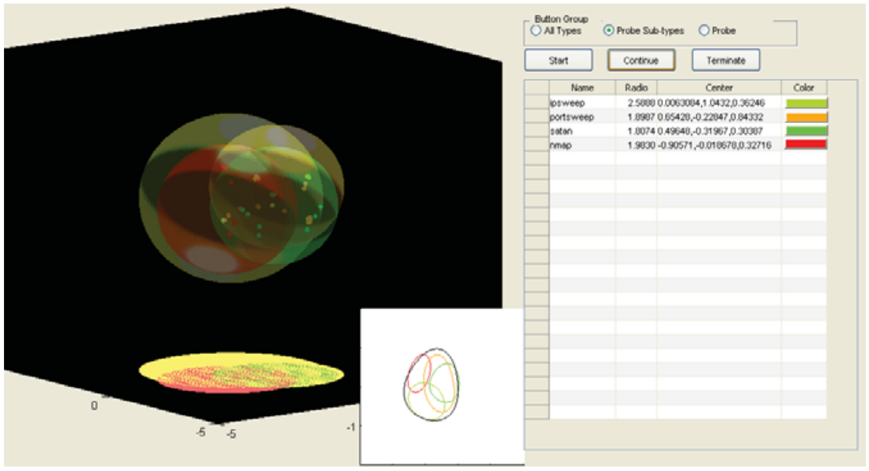
B.1 DESIGN OVERVIEW

The HCVM Prototype System simulates the process of learning on-line stream data. It uses 10% of KDD'99 training data. This prototype system demonstrates the MEB updating progress for each cluster as 3D balls and 2D boundary image. The core vectors for each MEB are represented as small dots. For convenience of observation, the point of observation can be changed to suit the user.

The interface of the HCVM Prototype System is shown in Figure B.1. Users are able to observe the learning progress on the left. The control panel is on right and details for each MEB are shown just under the control panel.

B.2 HARDWARE AND SOFTWARE REQUIREMENTS

The NIDVS requires Intel E8400@3.00 GHz or faster with at least 1.97 GB RAM. The video card should have over 256 MB RAM with support on Pixel Shader 3.0. The required software environment for NIDVS is: MATLAB



Name	Radio	Center	Color
ipsweep	2.5888	0.0063084,1.0432,0.36246	🟩
portsweep	1.8987	0.65428,-0.22847,0.84332	🟧
satan	1.8074	0.49848,-0.31967,0.30387	🟩
nmap	1.9830	-0.90571,-0.018678,0.32716	🟥

Figure B.1: The HCVM Prototype System.

R2009a on the platform of Microsoft Windows XP Professional with Service Pack 3 or advanced. The best screen resolution is 1280 by 1024 pixels.

B.3 USER INTERFACE DESIGN

B.3.1 *Description of the User Interface*

The user interface consists of four components: MEBs visualisation, control panel, MEB information list table as explained below.

B.3.2 *Objects and Actions*

1. MEBs visualisation

   Figure B.2 depicts a dynamic visualisation of MEB, its core vectors and the corresponding boundary for each cluster. While 3D balls in different colours represent a cluster for a sub-class, the dots in the balls represent the core vectors. Boundaries shown in the right bottom corner are formed by the same-colour core vectors. The parent class boundary is in black colour.

2. Control Panel

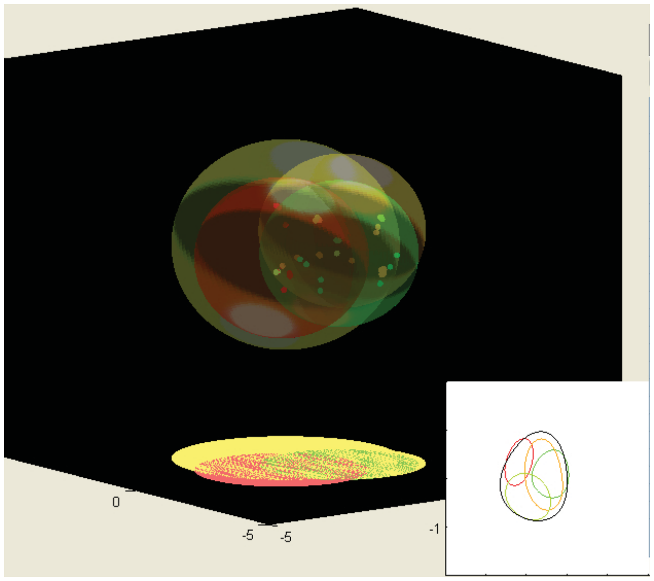


Figure B.2: MEBs, core vectors and the corresponding boundaries visualisation.

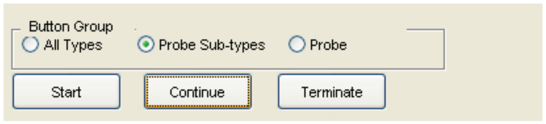


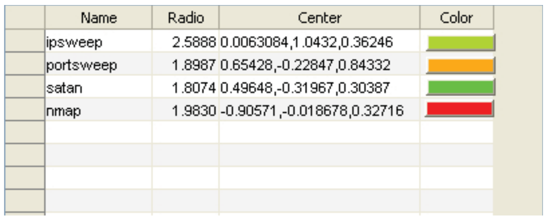
Figure B.3: The control panel.

The user is able to select different learning modes by using the control panel shown in Figure B.3. Global mode learning can be demonstrated if ‘Probe’ is selected. You can compare the effectiveness of global mode learning with local mode by selecting ‘Probe sub-type’. The theory of local mode learning can be seen if ‘All Types’ is selected.

The three buttons control the prototype system. ‘Start’ initiates the visualisation demo. ‘Pause/continue’ suspends or proceeds with the visualisation of data flow. ‘Terminate’ stops the visualisation demonstration.

3. MEB information list table

   Figure B.4 depicts the dynamical report of the details of each MEB shown in the visualisation area, which includes information about the name of each class, current ball centre and radius corresponding to the same colour of MEB.



Name	Radio	Center	Color
ipsweep	2.5888	0.0063084,1.0432,0.36246	🟩
portsweep	1.8987	0.65428,-0.22847,0.84332	🟧
satan	1.8074	0.49648,-0.31967,0.30387	🟩
nmap	1.9830	-0.90571,-0.018678,0.32716	🟥

Figure B.4: MEB information list table

REFERENCES

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on very large data bases - volume 29* (pp. 81–92). VLDB Endowment.

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2004). A framework for projected clustering of high dimensional data streams. In *Proceedings of the thirtieth international conference on very large data bases - volume 30* (pp. 852–863). VLDB Endowment.

Akyildiz, I., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002, aug). A survey on sensor networks. *Communications Magazine, IEEE*, *40*(8), 102 - 114.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, *215*(3), 403 - 410.

Angelov, P. (2004). An approach for fuzzy rule-base adaptation using on-line clustering. *International Journal of Approximate Reasoning*, *35*(3), 275 - 289. (Integration of Methods and Hybrid Systems)

Angelov, P. (2010). Evolving takagi-sugeno fuzzy systems from streaming data (ets+). In *Evolving intelligent systems* (pp. 21–50). John Wiley & Sons, Inc.

Angelov, P., & Buswell, R. A. (2003). Automatic generation of fuzzy rule-based models from data by genetic algorithms. *Information Sciences*, *150*(1-2), 17 - 31.

The Australian. (2010). Retrieved April 14, 2010 from http://www.theaustralian.com.au.

Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 1–16). New York, NY, USA: ACM.

Babcock, B., Datar, M., Motwani, R., & O'Callaghan, L. (2003). Maintaining variance and k-medians over data stream windows. In *Proceedings of the twenty-second acm sigmod-sigact-sigart symposium on principles of* *database systems* (pp. 234–243). New York, NY, USA: ACM.

Bā̄doiu, M., Har-Peled, S., & Indyk, P. (2002). Approximate clustering via core-sets. In *Stoc ’02: Proceedings of the thiry-fourth annual acm symposium on theory of computing* (pp. 250–257). New York, NY, USA: ACM.

Bandyopadhyay, S., Holder, L. B., & Maulik, U. (2006). *Advanced methods for knowledge discovery from complex data; electronic version*. Dordrecht: Springer.

Ben-Hur, A., Horn, D., Siegelmann, H. T., & Vapnik, V. (2002). Support vector clustering. *J. Mach. Learn. Res.*, *2*, 125–137.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on computational learning theory* (p. 144-152). New York, NY, USA: ACM.

Boutell, M. R. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757-1771.

Brazdil, P., Giraud-Carrier, C., Soares, C., & Vilalta, R. (2008). *Metalearning: Applications to data mining*. New York,NY: Springer-Verlag.

Brazdil, P., Soares, C., & Da Costa, J. (2003). Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, *50*(3), 251–277.

Burges, C. J. (1998a). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121-167.

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

Burl, M. C., Fowlkes, C., Roden, J., Stechert, A., & Mukhtar, S. (1999). Diamond eye: A distributed architecture for image data mining. In *In spie aerosense conf. on data mining and knowledge discovery.*

Cevikalp, H., & Polikar, R. (2008, oct.). Local classifier weighting by quadratic programming. *Neural Networks, IEEE Transactions on*, *19*(10), 1832 -1838.

Chang, C. chung, & Lin, C.-J. (2001). *Libsvm: a library for support vector machines.*

Charikar, M., O'Callaghan, L., & Panigrahy, R. (2003). Better streaming algorithms for clustering problems. In *Proceedings of the thirty-fifth annual acm symposium on theory of computing* (pp. 30–39). New York, NY, USA: ACM.

Chen, W.-H., Hsu, S.-H., & Shen, H.-P. (2005). Application of svm and ann for intrusion detection. *Computers and Operations Research*, *32*(10), 2617 - 2634. (Applications of Neural Networks)

Chen, Y., Dong, G., Han, J., Wah, B. W., & Wang, J. (2002). Multi-dimensional regression analysis of time-series data streams. In *Proceedings of the 28th international conference on very large data bases* (pp. 323–334). VLDB Endowment.

Chen, Y., Pang, S., & Kasabov, N. (2010). Factorizing class characteristics via group mebs construction. In *Proceedings of the 17th international conference on neural information processing: models and applications - volume part ii* (pp. 283–290). Berlin, Heidelberg: Springer-Verlag.

Chen, Y., Pang, S., Kasabov, N., Ban, T., & Kadobayashi, Y. (2009). Hierarchical core vector machines for network intrusion detection. In C. Leung, M. Lee, & J. Chan (Eds.), *Neural information processing* (Vol. 5864, p. 520-529). Springer Berlin / Heidelberg.

Chirita, P.-A., Diederich, J., & Nejdl, W. (2005). Mailrank: using ranking for spam detection. In *Cikm '05: Proceedings of the 14th acm international conference on information and knowledge management* (pp. 373–380). New York, NY, USA: ACM.

Combs, G. (2008). Wireshark-network protocol analyzer. *Current July*.

Cormode, G., & Muthukrishnan, S. (2003). What's hot and what's not: tracking most frequent items dynamically. In *Proceedings of the twenty-second acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 296–306). New York, NY, USA: ACM.

Cortes, C., Haffner, P., & Mohri, M. (2004, December). Rational kernels: Theory and algorithms. *J. Mach. Learn. Res.*, *5*, 1035–1062.

CSI, & FBI. (2005). Proceedings of the 10th annual computer crime and security survey 10. In (p. 1-23).

De-Bie, T., & Cristianini, N. (2004). Kernel methods for exploratory pattern analysis: A demonstration on text data. *Structural, Syntactic, and*

Statistical Pattern Recognition, 16–29.

Dickerson, J., Juslin, J., Koukousoula, O., & Dickerson, J. (2001, July). Fuzzy intrusion detection. In *Ifsa world congress and 20th nafips international conference, 2001. joint 9th* (Vol. 3, p. 1506-1510 vol.3).

Ding, Q., Ding, Q., & Perrizo, W. (2002). Decision tree classification of spatial data streams using peano count trees. In *Proceedings of the 2002 acm symposium on applied computing* (pp. 413–417). New York, NY, USA: ACM.

Diplaris, s. (2005). Protein classification with multiple algorithms. In *Proceedings of the 10th panhellenic conference on informatics (pci 2005).*

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth acm sigkdd international conference on knowledge discovery and data mining* (pp. 71–80). New York, NY, USA: ACM.

Domingos, P., & Hulten, G. (2001). A general method for scaling up machine learning algorithms and its application to clustering. In *In proceedings of the eighteenth international conference on machine learning* (pp. 106–113). Morgan Kaufmann.

Dong, G., Han, J., Lakshmanan, L. V. S., Pei, J., Wang, H., & Yu, P. S. (2003). Online mining of changes from data streams.

Elkan, C. (2000, January). Results of the kdd'99 classifier learning. *SIGKDD Explor. Newsl.*, *1*, 63–64.

Elson, J. (2003). *tcpflow - a tcp flow recorder.* Retrieved November 29, 2009 from http://www.circlemud.org/ jelson/software/tcpflow/.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of data mining in computer security.* Kluwer.

Fawcett, T. (2003). *Roc graphs: Notes and practical considerations for data mining researchers.*

Fayyad, U. M. (1998). Mining databases: Towards algorithms for knowledge discovery. *IEEE Data Engineering Bulletin*, *21*, 39–48.

Fontenla-Romero, O., Alonso-Betanzos, A., Castillo, E., Principe, J., & Guijarro-Berdias, B. (2002). Local modeling using self-organizing maps and single layer neural networks. In J. Dorronsoro (Ed.), *Artificial neural* *networks icann 2002* (Vol. 2415, p. 142-142). Springer Berlin / Heidelberg.

Frank, J., & Mda-c, N. U. (1994). Artificial intelligence and intrusion detection: Current and future directions. In *In proceedings of the 17th national computer security conference.*

Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992, September). Knowledge discovery in databases: an overview. *AI Mag.*, *13*, 57–70.

Freund, Y. (1997, January). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119-139.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition (2nd ed.)*. San Diego, CA, USA: Academic Press Professional, Inc.

Furdík, K., Paralič, J., & Tutoky, G. (2008). Meta-learning method for automatic selection of algorithms for text classification. In *Central European Conference on Information and Intelligent Systems 19 th International Conference 2008 Proceedings.*

Ganti, V., Gehrke, J., & Ramakrishnan, R. (2002, January). Mining data streams under block evolution. *SIGKDD Explor. Newsl.*, *3*, 1–10.

Garcia, V., Alejo, R., Sinchez, J. S., Sotoca, J. M., & Mollineda, R. A. (2008). Combined effects of class imbalance and class overlap on instance-based classification.

Giannella, C., Yang, J., Zhang, J., Yan, X., & Yu, P. S. (2008). Chapter 3 mining frequent patterns in data streams at multiple time.

Giannotti, F., Nanni, M., Pinelli, F., & Pedreschi, D. (2007). Trajectory pattern mining. In *Proceedings of the 13th acm sigkdd international conference on knowledge discovery and data mining* (pp. 330–339). New York, NY, USA: ACM.

Gilbert, A. C., Kotidis, Y., Muthukrishnan, S., & Strauss, M. J. (2003). One-pass wavelet decompositions of data streams. *IEEE Transactions on Knowledge and Data Engineering*, *15*, 541-554.

Giraud-Carrier, C., Vilalta, R., & Brazdil, P. (2004). Introduction to the special issue on meta-learning. *Machine Learning*, *54*(3), 187–193.

Goel, A., Indyk, P., & Varadarajan, K. (2001). Reductions among high dimensional proximity problems. In *Soda '01: Proceedings of the* *twelfth annual acm-siam symposium on discrete algorithms* (pp. 769–778). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Golab, L., & Özsu, M. T. (2003, June). Issues in data stream management. *SIGMOD Rec.*, *32*, 5–14.

Guha, S., Mishra, N., Motwani, R., & O'Callaghan, L. (2000). Clustering data streams. In *Foundations of computer science, 2000. proceedings. 41st annual symposium on* (p. 359 -366).

Guralnik, V., & Srivastava, J. (1999). Event detection from time series data. In *Proceedings of the fifth acm sigkdd international conference on knowledge discovery and data mining* (pp. 33–42). New York, NY, USA: ACM.

Hall, P. M., Marshall, D., & Martin, R. R. (1998). Incremental eigenanalysis for classification. In *in british machine vision conference* (pp. 286–295).

Hand, D. J. (1999, June). Statistics and data mining: intersecting disciplines. *SIGKDD Explor. Newsl.*, *1*, 16–19.

Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of Data Mining*. MIT Press.

Hastie, T., Tibshirani, R., & Friendman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag.

Hendrik, B., Leander, S., Jan, S., & Amanda, C. (2006). Decision trees for hierarchical multilabel classification: A case study in functional genomics. *Journal of Machine Learning Research*, *4213/2006*, 18-29.

Henikoff, J. (1992). Amino acid substitution matrices from protein blocks. *PNAS*, *89*.

Henzinger, M. R., & Raghavan, P. (1998). *Computing on data streams* (Tech. Rep.).

Hersh, W. (2008). *Information retrieval: A health and biomedical perspective*. New York,NY: Springer Verlag.

Himberg, J., Tikanmaki, J., Toivonen, H. T., Korpiaho, K., & Mannila, H. (2001). Time series segmentation for context recognition in mobile devices. *Data Mining, IEEE International Conference on*, *0*, 203.

Hripcsak, G., & Rothschild, A. S. (2005). Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical In-*

formatics Association, *12*(3), 296 - 298.

Hubbard, P. M. (1996). Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.*, *15*(3), 179–210.

The Independent. (2010). Retrieved April 11, 2010 from http://www.independent.co.uk.

Indyk, P., Koudas, N., & Muthukrishnan, S. (2000). Identifying representative trends in massive time series data sets using sketches. In *Proceedings of the 26th international conference on very large data bases* (pp. 363–372). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Islam, M., Yao, X., Shahriar Nirjon, S., Islam, M., & Murase, K. (2008, june). Bagging and boosting negatively correlated neural networks. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *38*(3), 771 -784.

Jang, J. (1993, may/jun). Anfis: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, *23*(3), 665 -685.

Jang, J., & Sun, C. (1995, mar). Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, *83*(3), 378 -406.

Jang, J., Sun, C., & Mizutani, E. (1997, oct). Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [book review]. *Automatic Control, IEEE Transactions on*, *42*(10), 1482 -1484.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In (pp. 200–209). Morgan Kaufmann.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. *Proceedings of the International Conference on Machine Learning (ICML)*, 290 - 297.

Kargupta, H., Park, B.-H., Pittie, S., Liu, L., Kushraj, D., & Sarkar, K. (2002, January). Mobimine: monitoring the stock market from a pda. *SIGKDD Explor. Newsl.*, *3*, 37–46.

Kasabov, N. (1998). Ecos: Evolving connectionist systems and the eco learning paradigm. *Proc. ICONIP'98. Kitakyushu, Japan*, 1222-1235.

Kasabov, N. (2001, dec). Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *31*(6), 902 -918.

Kasabov, N. (2002). *Evolving connectionist systems: Methods and applications in bioinformatics, brain study and intelligent machines*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Kasabov, N. (2007a). Brain-, gene-, and quantum inspired computational intelligence: Challenges and opportunities. In W. Duch & J. Mandziuk (Eds.), *Challenges for computational intelligence* (Vol. 63, p. 193-219). Springer Berlin / Heidelberg.

Kasabov, N. (2007b). *Evolving connectionist systems: The knowledge engineering approach*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Kasabov, N. (2007c). Global, local and personalised modeling and pattern discovery in bioinformatics: An integrated approach. *Pattern Recognition Letters*, *28*(6), 673 - 685. (Pattern Recognition in Cultural Heritage and Medical Applications)

Kasabov, N., & Pang, S. (2003, dec.). Transductive support vector machines and applications in bioinformatics for promoter recognition. In *Neural networks and signal processing, 2003. proceedings of the 2003 international conference on* (Vol. 1, p. 1 - 6 Vol.1).

Kasabov, N., & Song, Q. (2002, apr). Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *Fuzzy Systems, IEEE Transactions on*, *10*(2), 144 -154.

Katagiri, S., & Abe, S. (2006a). Incremental training of support vector machines using hyperspheres. *Pattern Recognition Letters*, *27*(13), 1495 - 1507.

Katagiri, S., & Abe, S. (2006b). Incremental training of support vector machines using truncated hypercones. In F. Schwenker & S. Marinai (Eds.), *Artificial neural networks in pattern recognition* (Vol. 4087, p. 153-164). Springer Berlin / Heidelberg.

KDD99. (1999). Kdd cup 1999 data received 2008 from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Keogh, E., & Lin, J. (2005). Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and Information Systems*, *8*, 154-177. (10.1007/s10115-004-0172-7)

Kim, H. C., Pang, S., Je, H.-M., Kim, D., & Bang, S.-Y. (2002). Support vector machine ensemble with bagging. In S.-W. Lee & A. Verri (Eds.), *Pattern recognition with support vector machines* (Vol. 2388, p. 131-141). Springer Berlin / Heidelberg.

Kim, M. S., Kim, D., & Lee, S.-Y. (2003). Face recognition using the embedded hmm with second-order block-specific observations. *Pattern Recognition*, *36*(11), 2723 - 2735.

Kim, T. K., Wong, S. F., Stenger, B., Kittler, J., & Cipolla, R. (2007, june). Incremental linear discriminant analysis using sufficient spanning set approximations. In *Computer vision and pattern recognition, 2007. cvpr '07. ieee conference on* (p. 1 -8).

Kofcz, E., Chowdhury, A., & Alspector, J. (n.d.). *The impact of feature selection on signature-driven spam detection.*

Kord, P., Koutn, J., Drchal, J., Kov, O., Cepek, M., & Snorek, M. (2010). Meta-learning approach to neural network optimization. *Neural Networks*, *23*(4), 568 - 582. (The 18th International Conference on Artificial Neural Networks, ICANN 2008)

Kumar, P., Mitchell, J. S. B., & Yildirim, E. A. (2003). Approximate minimum enclosing balls in high dimensions using core-sets. *J. Exp. Algorithmics*, *8*, 1.1.

Lam, W., & Lai, K. (2001). A meta-learning approach for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR conference on Research and Development in Information Retrieval* (p. 309).

Last, M. (2002, April). Online classification of nonstationary data streams. *Intell. Data Anal.*, *6*, 129–147.

Learning with progressive transductive support vector machine. (2003). *Pattern Recognition Letters*, *24*(12), 1845 - 1855.

Leslie, C., Eskin, E., Weston, J., & Noble, W. S. (2002). Mismatch string kernels for svm protein classification. *Neural Information Processing Systems*, *16*.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, *5*, 361–397.

Li, J., & Chua, C.-S. (2003, sept.). Transductive inference for color-based particle filter tracking. In *Image processing, 2003. icip 2003. proceedings. 2003 international conference on* (Vol. 3, p. III - 949-52 vol.2).

Lin, J., Keogh, E., Lonardi, S., & Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th acm sigmod workshop on research issues in data mining and knowledge discovery* (pp. 2–11). New York, NY, USA: ACM.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *The Journal of Machine Learning Research*, *2*, 444.

Lucks, M., & Oki, N. (1999). A radial basis function network (rbfn) for function approximation. In *Circuits and systems, 1999. 42nd midwest symposium on* (Vol. 2, p. 1099 -1101 vol. 2).

Manku, G. S., & Motwani, R. (2002). Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on very large data bases* (pp. 346–357). VLDB Endowment.

Marin, J., Ragsdale, D., & Sirdu, J. (2001). A hybrid approach to the profile creation and intrusion detection. In *Darpa information survivability conference and exposition ii, 2001. discex '01. proceedings* (Vol. 1, p. 69-76 vol.1).

Mayur, B. B., Babcock, B., Datar, M., & Motwani, R. (2003). Load shedding techniques for data stream systems. In *In proc. of the 2003 workshop on management and processing of data streams (mpds.*

Megiddo, N. (1983). Linear-time algorithms for linear programming in r^3 and related problems. *SIAM Journal on Computing*, *12*(4), 759-776.

Mukkamala, S., & Sung, A. H. (2003). Identifying significant features for network forensic analysis using artificial intelligent techniques. *Intl. Journal of Digital Evidence*, *1*, 2003.

Multilayer feedforward networks are universal approximators. (1989). *Neural Networks*, *2*(5), 359 - 366.

Muthukrishnan, S. (2003). *Data streams: algorithms and applications.*

New Zealand Herald. (2010). Retrieved April 12, 2010 from http://www.nzherald.co.nz.

Nguyen, M. H., Abbass, H., & McKay, R. (2008, jan.). Analysis of ccme: Coevolutionary dynamics, automatic problem decomposition, and regularization. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *38*(1), 100 -109.

O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., & Motwani, R. (2002). Streaming-data algorithms for high-quality clustering. In *Data engineering, 2002. proceedings. 18th international conference on* (p. 685 -694).

Ordonez, C. (2003). Clustering binary data streams with k-means. In *Proceedings of the 8th acm sigmod workshop on research issues in data mining and knowledge discovery* (pp. 12–19). New York, NY, USA: ACM.

Ozawa, S., Pang, S., & Kasabov, N. (2010). Online feature extraction for evolving intelligent systems. In *Evolving intelligent systems* (pp. 151–171). John Wiley & Sons, Inc.

Panda, M., & Patra, M. R. (2007). Network intrusion detection using naive bayes. *International journal of computer science and network security,*, 258-263.

Pang, S., Havukkala, I., & Kasabov, N. (2006). Two-class svm trees (2-svmt) for biomarker data analysis. In J. Wang, Z. Yi, J. Zurada, B.-L. Lu, & H. Yin (Eds.), *Advances in neural networks - isnn 2006* (Vol. 3973, p. 629-634). Springer Berlin / Heidelberg.

Pang, S., & Kasabov, N. (2004, july). Inductive vs transductive inference, global vs local models: Svm, tsvm, and svmt for gene expression classification problems. In *Neural networks, 2004. proceedings. 2004 ieee international joint conference on* (Vol. 2, p. 1197 - 1202 vol.2).

Pang, S., Kim, D., & Bang, S. Y. (2003). Membership authentication in the dynamic group by face classification using svm ensemble. *Pattern Recognition Letters*, *24*(1-3), 215 - 225.

Pang, S., Kim, D., & Bang, S. Y. (2005, march). Face membership authentication using svm classification tree generated by membership-based lle data partition. *Neural Networks, IEEE Transactions on*, *16*(2), 436 -446.

Papadimitriou, S., Brockwell, A., & Faloutsos, C. (2003). Adaptive, hands-off stream mining. In *Proceedings of the 29th international conference on very large data bases - volume 29* (pp. 560–571). VLDB Endowment.

Park, B., & Kargupta, H. (2002). Distributed data mining: Algorithms, systems, and applications.

Perlman, E., & Java, A. (2002, Dec). *Predictive mining of time series data in astronomy* (Tech. Rep. No. astro-ph/0212413).

Rieck, K., & Laskov, P. (2007). Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology*, *2*(4), 243–256.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323-2326.

Sagot, M. (1998). Spelling approximate repeated or common motifs using a suffix tree. In C. Lucchesi & A. Moura (Eds.), *Latin'98: Theoretical informatics* (Vol. 1380, p. 374-390). Springer Berlin / Heidelberg. (10.1007/BFb0054337)

Schapire, R. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2), 135-168.

Schechter, S., Parnell, T., & Hartemink, A. (1999). Anonymous authentication of membership in dynamic groups. In M. Franklin (Ed.), *Financial cryptography* (Vol. 1648, p. 184-195). Springer Berlin / Heidelberg.

Sharma, O., Girolami, M., & Sventek, J. (2007). Detecting worm variants using machine learning. In *Proceedings of the 2007 acm conext conference* (pp. 1–12).

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis* (illustrated edition ed.). Cambridge University Press. Hardcover.

Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, *24*(4), 35–43.

Soares, C., Brazdil, P., & Kuba, P. (2004). A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, *54*(3), 195–209.

Song, Q., & Kasabov, N. (2005, dec.). Nfi: a neuro-fuzzy inference method for transductive reasoning. *Fuzzy Systems, IEEE Transactions on*, *13*(6), 799 - 808.

Spam assassin public mail corpus. (2002). %url-http://spamassassin.apache.org/publiccorpus/. (Retrieved December 23, 2009)

Srivastava, A. N. (2003). *Onboard detection of snow, ice, clouds and other geophysical processes using kernel methods.*

S. Sonnenburg, C. S. B. S., G. Raetsch. (2006). Large scale multiple kernel learning. *The Journal of Machine Learning Research*, *7*, 1565.

Staff, C. (2005). Hackers: companies encounter rise of cyber extortion. *Computer Crime Research Center, 2006*.

Suykens, J., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, *9*, 293-300.

Tague-Sutcliffe, J. (1997). The pragmatics of information retrieval experimentation, revisited. , 205–216.

Tanner, S., Alshayeb, M., Criswell, E., Iyer, M., & Mcdowell, A. (2008). Eve: On-board process planning and execution.

The Times. (2010). Retrieved April 12, 2010 from http://www.timesonline.co.uk.

Tsang, I. W., Kwok, J. T., & Cheung, P.-M. (2005). Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, *6*, 363-392.

Tsoumakas, G., & Katakis, I. (2007, Jul-Sep). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, *3*(3), 1-12.

Ueffing, N. (2007). Transductive learning for statistical machine translation. In *In proc. of acl* (pp. 25–32).

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience. Hardcover.

Vilalta, R., Giraud-Carrier, C., Brazdil, P., & Soares, C. (2004). Using meta-learning to support data mining. *International Journal of Computer Science Applications*, *1*(1), 31–45.

Wang, D., Zhang, B., Zhang, P., & Qiao, H. (2010). An online core vector machine with adaptive meb adjustment. *Pattern Recognition*, *43*(10), 3468 - 3482.

Wang, H., Fan, W., Yu, P. S., & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth acm sigkdd international conference on knowledge discovery and data mining* (pp. 226–235). New York, NY, USA: ACM.

Welzl, E. (1991). Smallest enclosing disks (balls and ellipsoids). In *Results and new trends in computer science* (pp. 359–370). Springer-Verlag.

Xie, F., Xu, G., & Hundt, E. (2001). A face verification algorithm integrating geometrical and template features. In H.-Y. Shum, M. Liao, & S.-

F. Chang (Eds.), *Advances in multimedia information processing pcm 2001* (Vol. 2195, p. 253-260). Springer Berlin / Heidelberg.

Yager, R., & Filev, D. (1994, aug). Approximate clustering via the mountain method. *Systems, Man and Cybernetics, IEEE Transactions on*, *24*(8), 1279 -1284.

Yamada, T., Yamashita, K., Ishii, N., & Iwata, K. (2006). Text classification by combining different distance functions withweights. *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, International Conference on & Self-Assembling Wireless Networks, International Workshop on*, *0*, 85-90.

Yang, M.-H., Ahuja, N., & Kriegman, D. (2000). Face recognition using kernel eigenfaces. In *Image processing, 2000. proceedings. 2000 international conference on* (Vol. 1, p. 37 -40 vol.1).

Yang, S., Ho, C., & Lee, C. (2006, march). Hbp: improvement in bp algorithm for an adaptive mlp decision feedback equalizer. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, *53*(3), 240 - 244.

Yao, X., & Liu, Y. (1996, may). Ensemble structure of evolutionary artificial neural networks. In *Evolutionary computation, 1996., proceedings of ieee international conference on* (p. 659 -664).

Yao, X., & Liu, Y. (1998, jun). Making use of population information in evolutionary artificial neural networks. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *28*(3), 417 -425.

Ye, N., & Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *International*, *17*, 105–112.

Zhang, T., & Oles, F. J. (2000). Text categorization based on regularized linear classification methods. *Information Retrieval*, *4*, 5–31.

Zhou, Z.-H., & Jiang, Y. (2003, march). Medical diagnosis with c4.5 rule preceded by artificial neural network ensemble. *Information Technology in Biomedicine, IEEE Transactions on*, *7*(1), 37 -42.

Zhu, Y., & Shasha, D. (2002). Statstream: statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on very large data bases* (pp. 358–369). VLDB Endowment.