Kobe University Repository : Kernel

PDF issue: 2026-11-18

Incremental Learning for Large-Scale Stream Data and Its Application to Cybersecurity

SITI HAJAR AMINAH BINTI ALI

(Degree)
博士（工学）

(Date of Degree)
2015-09-25

(Date of Publication)
2017-09-25

(Resource Type)
doctoral thesis

(Report Number)
甲第6472号

(URL)
https://hdl.handle.net/20.500.14094/D1006472

※ 当コンテンツは神戸大学の学術成果です。無断複製・不正使用等を禁じます。著作権法で認められている範囲内で、適切にご利用ください。

Doctoral Dissertation

Incremental Learning for Large-Scale Stream Data and Its Application to Cybersecurity

大規模ストリームデータのための追加学習とサイバーセキュリティへの応用

July 2015

Graduate School of Engineering

Kobe University

Siti Hajar Aminah Binti Ali

Acknowledgements

First of all, *Alhamdulillah* thanks to our creator, *Allah*, for the continuous blessings and strengths which had been given to me and my beloved family.

I am extremely grateful to my supervisor, Professor Dr. Seiichi Ozawa, for his guidance, support and helpful suggestions throughout my Ph.D study. I wish to thanks Associate Professor Dr. Toshiaki Omori and Dr. Jun Kitazono for willing to give advices and guidances as to improve the quality of my research. Special thanks go to Professor Dr. Masakatu Morii and Professor Dr. Chikara Ohta for the suggestions and comments to improve my dissertation report.

My family deserves special mention for their constant support and their role of being the driving force towards the success of my Ph.D project especially my dearest husband Mohd Fahrul Radzi, my sons: Abdullah Zubair, Abdullah Umar and Abdullah Abbas, and my big family Miskan Siman, Sapiah A. Rahman, Ali Naem, Patimah Ebrahim, Nozli Hanim & Zulkifli, Nor Faraliza & Fuad, Mohd Fahrulnizam & Nuwal, Noor Azila & Rushdi, Mohd Firdaus, Ahmad Talal & KNazwa, Ahmad Omar & Siha, Ahmad Othman & Fairuz and Ahmad Sallehin. Thank you for being patient and always supporting me towards the end.

My sincere appreciation also goes to everyone whom I may not have mentioned above who have helped me directly or indirectly in the completion of this project especially ES5 lab members and MADANI Postgraduates Comittee.

Finally, I would like to express my gratitude to the Ministry of Education (MOE) Malaysia and University Tun Hussein Onn Malaysia (UTHM) for their scholarship award and study leave. This thesis would have not been accomplished without funds which supported by both parties.

Thank you very much.

Abstract

As many human currently depend on technologies to assist with daily tasks, there are more and more applications which have been developed to be fit in one small gadget such as smart phone and tablet. Thus, by carrying this small gadget alone, most of our tasks are able to be settled efficiently and fast. Until the end of 20th century, mobile phones are only used to call and to send short message service (sms). However, in early 21st century, a rapid revolution of communication technology from mobile phone into smart phone has been seen in which the smart phone is equipped by 4G Internet line along with the telephone service provider line. Thus, the users are able to make a phone call, send messages using variety of application such as *Whatsapp* and *Line*, send email, serving websites, accessing maps and handling some daily tasks via online using online banking, online shopping and online meetings via video conferences. In previous years, if there are cases of missing children or missing cars, the victims would rely on the police investigation. But now, as easy as uploading a notification about the loss on *Facebook* and spread the news among *Facebook* users, there are more people are able to help in the search. Despite the advantages that can be obtained using these technologies, there are a group of irresponsible people who take advantage of current technologies for their own self-interest. Among the applications that are usually being used by almost Internet users and also are often misused by cyber criminals are email and websites. Therefore, we take this initiative to make enhancement in cyber security application to avoid the Internet users from being trapped and deceived by the trick of cyber criminals by developing detection system of malicious spam email and Distributed Denial of Services (DDoS)

backscatter.

Imagine that a notice with a logo of Mobile Phone company is received by an email informing that the customer had recently run up a large mobile phone bill. A link regarding the bill is attached for him/her to find out the details. Since, the customer thinks that the billing might be wrong, thus the link is clicked. However, the link is directed to a webpage which displays a status that currently the webpage is under construction. Then the customer closes the page and thinking of to visit the website again at other time. Unfortunately, after a single click actually a malicious file is downloaded and installed without the customer aware of it. That malicious file most probably is a Trojan that capable to steal confidential information from victim's computer. On the next day, when the same person is using the same computer to log in the online banking, all of a sudden find out that his/her money is lost totally. This is one of a worst case scenario of malicious spam email which is usually handled by cybersecurity field. Another different case of cybersecurity is the Distributed Denial of Services (DDoS) attack. Let say, *Company X* is selling flowers via online in which the market is from the local and international customer. The online business of *Company X* is running normally as usual, until a day before mother's day, the webpage of *Company X* is totally down and the prospective customers could not open the webpage to make order to be sent specially for their beloved mother. Thus, the customers would search another company that sells the same item. The *Company X* server is down, most probably because of the DDoS attack where a junk traffic is sent to that company server which makes that server could not serve the request by the legitimate customers. This attack effect not only the profit of the company, but also reputation damage, regular customer turnover and productivity decline.

Unfortunately, it is difficult for a normal user like us to detect malicious spam

email or DDoS attack with naked eyes. It is because recently the spammers and attacker had improved their strategy so that the malicious email and the DDoS packets are hardly able to be differentiated with the normal email and data packets. Once the *Social Engineering* is used by the spammers to create relevant email content in the malicious spam email and when a new campaign of DDoS attack is launched by the attacker, no normal users are capable to distinguish the benign and malicious email or data packets. This is where my Ph.D project comes in handy. My Ph.d is focusing on constructing a detection system of malicious spam email and DDoS attack using a large number of dataset which are obtained by a server that collect double-bounce email and darknet for malicious spam email detection system and DDoS backscatter detection system, respectively. As many up-to-date data are used during the learning, the detection system would become more robust to the latest strategy of the cybercriminal. Therefore, the scenario mentioned above can be avoided by assisting the user with important information at the user-end such as malicious spam email filter or at the server firewall. First of all, the method to learn large-scale stream data must be solved before implementing it in the detection system. Therefore, in Chapter 2, the general learning strategy of large-scale data is introduced to be used in the cybersecurity applications which are discussed in Chapter 3 and Chapter 4, respectively.

One of a critical criterion of the detection system is capable to learn fast because after the learning, the updated information needs to be passed to user to avoid the user from being deceived by the cybercriminal. To process large-scale data sequences, it is important to choose a suitable learning algorithm that is capable to learn in real time. Incremental learning has an ability to process large data in chunk and update the parameters after learning each chunk. Such type of learning keep and update only the minimum information on a classifier model.

Therefore, it requires relatively small memory and short learning time. On the other hand, batch learning is not suitable because it needs to store all training data, which consume a large memory capacity. Due to the limited memory, it is certainly impossible to process online large-scale data sequences using the batch learning. Therefore, the learning of large-scale stream data should be conducted incrementally.

This dissertation contains of five chapters. In Chapter 1, the concept of incremental learning is briefly described and basic theories on Resource Allocating Network (RAN) and conventional data selection method are discussed in this chapter. Besides that, the overview of this dissertation is also elaborated in this chapter. In Chapter 2, we propose a new algorithm based on incremental Radial Basis Function Network (RBFN) to accelerate the learning in stream data. The data sequences are represented as a large chunk size of data given continuously within a short time. In order to learn such data, the learning should be carried out incrementally. Since it is certainly impossible to learn all data in a short period, selecting essential data from a given chunk can shorten the learning time. In our method, we select data that are located in *untrained* or "not well-learned" region and discard data at *trained* or "well-learned" region. These regions are represented by margin flag. Each region is consisted of similar data which are near to each other. To search the similar data, the well-known LSH method proposed by Andoni et al. is used. The LSH method indeed has proven be able to quickly find similar objects in a large database. Moreover, we utilize the LSH ' s properties; hash value and Hash Table to further reduced the processing time. A flag as a criterion to decide whether to choose or not the training data is added in the Hash Table and is updated in each chunk sequence. Whereas, the hash value of RBF bases that is identical with the hash value of the training data is used to select the RBF bases that is near to the training data. The performance results of

the numerical simulation on nine UC Irvine (UCI) Machine Learning Repository datasets indicate that the proposed method can reduce the learning time, while keeping the similar accuracy rate to the conventional method. These results indicate that the proposed method can improve the RAN learning algorithm towards the large-scale stream data processing.

In Chapter 3, we propose a new online system to detect malicious spam emails and to adapt to the changes of malicious URLs in the body of spam emails by updating the system daily. For this purpose, we develop an autonomous system that learns from double-bounce emails collected at a mail server. To adapt to new malicious campaigns, only new types of spam emails are learned by introducing an active learning scheme into a classifier model. Here, we adopt Resource Allocating Network with Locality Sensitive Hashing (RAN-LSH) as a classifier model with data selection. In this data selection, the same or similar spam emails that have already been learned are quickly searched for a hash table using Locally Sensitive Hashing, and such spam emails are discarded without learning. On the other hand, malicious spam emails are sometimes drastically changed along with a new arrival of malicious campaign. In this case, it is not appropriate to classify such spam emails into malicious or benign by a classifier. It should be analyzed by using a more reliable method such as a malware analyzer. In order to find new types of spam emails, an outlier detection mechanism is implemented in RAN-LSH. To analyze email contents, we adopt the Bag-of-Words (BoW) approach and generate feature vectors whose attributes are transformed based on the normalized term frequency-inverse document frequency. To evaluate the developed system, we use a dataset of double-bounce spam emails which are collected from March 1, 2013 to May 10, 2013. In the experiment, we study the effect of introducing the outlier detection in RAN-LSH. As a result, by introducing the outlier detection, we confirm that the detection accuracy is enhanced on

average over the testing period.

In Chapter 4, we propose a fast Distributed Denial of Service (DDoS) backscatter detection system to detect DDoS backscatter from a combination of protocols and ports other than the following two *labeled* packets: Transmission Control Protocol (TCP) Port 80 (80/TCP) and User datagram Protocol (UDP) Port 53 (53/UDP). Usually, it is hard to detect DDoS backscatter from the *unlabeled* packets, where an expert is needed to analyze every packet manually. Since it is a costly approach, we propose a detection system using Resource Allocating Network (RAN) with data selection to select essential data. Using this method, the learning time is shorten, and thus, the DDoS backscatter can be detected fast. This detection system consists of two modules which are pre-processing and classifier. With the former module, the packets information are transformed into 17 feature-vectors. With the latter module, the RAN-LSH classifier is used, where only data located at *untrained* region are selected. The performance of the proposed detection system is evaluated using 9,968 training data from 80/TCP and 53/UDP, whereas 5,933 test data are from *unlabeled* packets which are collected from January 1st, 2013 until January 20th, 2014 at National Institute of Information and Communications Technology (NICT), Japan. The results indicate that detection system can detect the DDoS backscatter from both *labeled* and *unlabeled* packets with high recall and precision rate within a short time.

Finally, in Chapter 5, we discussed the conclusions and the future work of our study: RAN-LSH classifier, malicious spam email detection system and DDoS backscatter detection system.

Contents

Acknowledgements **i**

Chapter 1 Introduction **1**

1.1 Background 1

1.2 Basic Theories of Machine Learning 7

1.2.1 Type of Learning 7

1.2.2 Learning Scheme 8

1.3 Previous Studies: Conventional Classifier, Data Selection Method, RBF Bases Selection and Fast Near Neighbor Search 10

1.3.1 Modified Resource Allocating Network (RAN) 10

1.3.2 Margin-based data selection 12

1.3.3 RBF Selection 15

1.3.4 Locality Sensitive Hashing 16

1.4 The Proposed Classifier Model 17

1.5 Cyber Security Application (1): Malicious Spam Email Detection System 19

1.6 Cyber Security Application (2): DDoS Backscatter Detection System 21

1.7 Performance Evaluations 23

1.8 Thesis Outlines and Research Objectives 24

Chapter 2 Propose Method: A Fast Online Learning Algorithm of Radial Basis Function Network with Locality Sensitive Hashing **28**

2.1 Introduction 28

2.2 The proposed LSH-Based Data Selection and RBF Bases Selection 31
2.2.1 Learning Algorithm 31
2.2.2 LSH-based Data Selection 33
2.2.3 LSH-based RBF Selection 36
2.3 Performance Evaluation 40
2.3.1 Experimental Setup 40
2.3.2 Granularity in Hash Encoding 41
2.3.3 Effectiveness of Data Selection 43
2.3.4 The Effect of RBF Bases Selection and Comparison to State-of-the-Art Method 46
2.4 Conclusions 50

Chapter 3 An Online Malicious Spam Email Detection System Using Resource Allocating Network with Locality Sensitive Hashing 52

3.1 Introduction 52
3.2 The Proposed Malicious Spam Email Detection System 54
3.2.1 System Architecture 54
3.2.2 Autonomous Spam Email Collection System 56
3.2.3 Autonomous Labeling System 56
3.2.4 Text Processing and Feature Transformation 57
3.2.5 Outlier Detection 59
3.2.6 RAN-LSH Classifier 62
3.3 Performance Evaluation 65
3.3.1 Experimental Setup 65
3.3.2 Effects of Threshold Parameters 68
3.3.3 Effectiveness of Incremental Learning 70
3.3.4 Overall Performance of Malicious Spam Email Detection System 72
3.4 Conclusions 73

Chapter 4 Distributed Denial of Service (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection **75**

4.1 Introduction 75

4.2 The Proposed DDoS Backscatter Detection System 78

4.2.1 System Architecture 78

4.2.2 Pre-processing 80

4.2.3 RAN-LSH Classifier 82

4.3 Performance Evaluation 86

4.3.1 Experimental Setup 86

4.3.2 Parameter Tuning for Data Selection 88

4.3.3 Performance of DDoS Detection System 90

4.4 Conclusions 91

Chapter 5 Conclusions **93**

5.1 Contributions of This Dissertation 95

5.2 Future Work 96

Bibliography **98**

Appendix AAlgorithms From Chapter 1 **107**

A.1 Algorithm 1: Modified RAN Learning 107

Appendix BAlgorithms From Chapter 2 **108**

B.1 Algorithm 2: RAN-LSH Learning Algorithm 108

B.2 Algorithm 3: Initialization 108

B.3 Algorithm 4: Update Hash Table 109

B.4 Algorithm 5: LSH-Based Data Selection 109

B.5 Algorithm 6: LSH-Based RBF Selection 110

Appendix CAlgorithms From Chapter 3 **111**

C.1 Algorithm 7: Malicious Spam Email Detection System 111

C.2 Algorithm 8: Pre-processing of Malicious Spam Email Detection System 111

C.3 Algorithm 9: Outlier Detection 112

Appendix DAlgorithms From Chapter 4 **113**

D.1 Algorithm 10: DDoS Backscatter Detection 113

D.2 Algorithm 11: Pre-processing of DDoS Backscatter Detection . . 114

D.3 Algorithm 12: Labeling (for TCP Port 80 and UDP Port 53) . . . 114

Appendix EThesis Outlines Mind Mapping **115**

E.1 Mind Mapping of Chapter 1 116

E.2 Mind Mapping of Chapter 2 117

E.3 Mind Mapping of Chapter 3 118

E.4 Mind Mapping of Chapter 4 119

List of Publications **120**

List of Figures

1.1 The differences between supervised learning and unsupervised learning. 9
1.2 Training data and test data of large-scale data sequences. 9
1.3 Radial Basis Function Networks (RBFN) 10
1.4 Margin-based data selection. 14
1.5 The illustration of global-RAN and local-RAN. 15
1.6 The overall network structure of proposed model. 27

2.1 Network Structure of RAN-LSH. 32
2.2 Calculation of hash values using an eigenvector. 39
2.3 Data projection into eigenvectors. 39
2.4 Index representation for each inputs. 39
2.5 Effect of the number of partitions P to the Davies-Bouldin index I_{DB} and the memory consumption for (a) Shuttle, (b) Pendigits, and (c) MAGIC datasets. 42
2.6 The effectiveness of selecting similar data using CIFAR-10 dataset. 44
2.7 Effect of RBF bases selection with different θ_p. Best viewed in color. 46

3.1 Network structure of the proposed autonomous malicious spam email detection system. 55
3.2 Example of web crawler and content analysis using SPIKE. 57
3.3 Example of normal and spam email. 60
3.4 Comparison of learning scheme between batch learning and incremental learning. 66
3.5 Transitions of recall rates in the malicious spam email detection system with three learning schemes. 71

3.6 Transitions of precision in the malicious spam email detection system with three learning schemes. 71

4.1 The architecture of DDoS backscatter detection system. 79
4.2 The example of darknet packets activity based on traffic features for: (a) DDoS backscatter and (b) DDoS non-backscatter.[69] . . . 82
4.3 Example of darknet packet for pre-processing. 84

E.1 Outline of Chapter 1. 116
E.2 Outline of Chapter 2. 117
E.3 Outline of Chapter 3. 118
E.4 Outline of Chapter 4. 119

List of Tables

1.1 Example of RAN calculation 13
1.2 Previous studies on malicious spam email detection. 20
1.3 Previous studies on DDoS attack. 22
1.4 The notation of TP, FP, FN and FP. 24
1.5 Thesis outlines. 26

2.1 Hash Table 37
2.2 Classification using RAN-LSH 38
2.3 Evaluated UCI Datasets. 40
2.4 Effect of the number of partitions P to learning time [sec.] 43
2.5 Effectiveness of introducing the data selection. The performance is compared among three models; modified RAN, RAN-MRG and RAN-LSH, regarding (a) the final recognition accuracy [%], (b) learning time [sec.], and (c) the total number of selected data. Note that modified RAN has no data selection mechanism. 45
2.6 The effect of selecting RBF centers using different threshold values θ_p namely through; (a) recognition accuracy [%], (b) incremental learning time [s], and (c) the total number of RBF bases selected. The values represented are the mean and standard deviation of each method. 47

3.1 Example of pre-processing for double-bounce email. 61
3.2 The evaluation using several values of accumulation ratio θ_a and output margin threshold θ_m for the spam email detection system. The performance measures are: (a) the F1 measure, and (b) initial learning time. 70

3.3 The performance using different values of tolerant distance θ_p. . . 70

3.4 Overall performance of malicious spam email detection system. . . 72

4.1 Example of pre-processing and data representation for darknet packet. 83

4.2 Parameter tuning of data selection using three months training data and 20 days test data. The performance are measured regarding the (a) total number of selected data, (b) F1 measure [%] and (c) learning time [sec.] 89

4.3 Comparison of three different model: RBFN, RAN and RAN-LSH, using different learning scheme. RBFN uses batch learning, whereas RAN and RAN-LSH use incremental learning setting. The performance are measured regarding (a) the recall rate [%], (b) precision rate [%], (c) F1 measure [%], and (d) learning time [s]. . 90

List of Abbrevistions

AEB	-	Advanced Entropy-Based
BoW	-	Bag-of-Words
DDoS	-	Distributed Denial of Services
DoS	-	Denial of Services
DNS	-	Domain Name System
FN	-	False Negative
FP	-	False Positive
HTTP	-	Hypertext Transfer Protocol
ID	-	identification
IDF	-	Inverse Document Frequency
IP	-	Internet Protocol
IPCA	-	Incremental Principal Component Analysis
IPv4	-	Internet Protocol version 4
LDOS	-	Low-rate Denial of Services
LSH	-	Locality Sensitive Hashing
MAGIC	-	MAGIC Gamma Telescope
MOE	-	Ministry of Education
NADA	-	Network Anomaly Detection Algorithm
NICT	-	National Institute of Information and Communications Technology
NNs	-	Nearest Neighbors
PCA	-	Principal Component Analysis
PREDICT	-	Protected Repository for Defense of Infrastructure Against Cyber Threats

QR	-	Query/Response flag
RAN	-	Resource Allocating Network
RAN-LSH	-	Resource Allocating Network-Locality Sensitive Hashing
RAN-LTM	-	Resource Allocating Network with Long Term Memory
RAN-MRG	-	RAN with margin-based data selection
RBF	-	Radial Basis Function
RBFN	-	Radial Basis Function Network
SVD	-	Singular Value Decomposition
SVM	-	Support Vector Machines
TCP	-	Transmission Control Protocol
TF	-	Term Frequency
TF-IDF	-	Term Frequency - Inverse Document Frequency
TME	-	Targeted Malicious Email
TN	-	True Negative
TP	-	True Positive
UCI	-	University of California, Irvine
UDP	-	User Datagram Protocol
URL	-	Uniform Resource Locator
UTHM	-	Universiti Tun Hussein Onn Malaysia
53/UDP	-	User Datagram Protocol Port 53
80/TCP	-	Transmission Control Protocol Port 80

List of Symbols

δ	RBF width threshold
σ	RBF width
θ_z	Network output
ε	Error threshold
$\boldsymbol{c}$	Center of RBF
δz	Output margin
Φ	RBF outputs matrix
θ_N	Occurrence frequency threshold
θ_p	Tolerant distance
θ_m	Output margin threshold
θ_a	Accumulation ratio
θ_o	Outlier threshold
θ_z	Highest network output
c^*	Nearest center to $\boldsymbol{x}_i$
$\mathcal{C}$	Index set of RBFs position in hash table
$\boldsymbol{d}$	K-dimensional target vector in which the class label element is one and the remainings are zeros (eg. for class label two with five total different classes; $\boldsymbol{d} = \{0, 1, 0, 0, 0\}$)
d_j^*	LSH distance
d	One of documents in D
$\boldsymbol{D}$	Target matrix
D	Documents in a corpus
E	Error
$f_d(t)$	Frequency of term t in a document d
F^M	Margin flag

F^O	Outlier flag
F^R	Response flag
$\boldsymbol{h}$	Hash value
$\mathcal{H}$	Hash table
$H(v_i)$	Hash function
I_{DB}	Davies-Bouldin index
$N_{d'}$	Number of training data
$N_{d''}$	Number of test data
N_e	Number of occurrence of similar data in each entry e
$N_{\mathcal{H}}$	Number of entries in hash table
N_o	Number of initial training data
P	Number of partitions
$\mathcal{R}$	Index set of selected RBFs
S_i	The variance of data allocated to the ith segment of P partition
t	Spesific term in document d
$\boldsymbol{U}_l$	Eigenvectors
$\boldsymbol{V}$	Projection vectors
v_i^+	Upper values of typical projections v_i
v_i^-	Lower values of typical projections v_i
$\boldsymbol{W}$	Connection weights
$\boldsymbol{x}$	Feature vector
$\bar{\boldsymbol{x}}$	Prototype
$\boldsymbol{X}'$	Training data
$\boldsymbol{X}''$	Test data
$\boldsymbol{X}_o$	Initial training data
$\tilde{\boldsymbol{X}}_s$	All data in each s subsets of unique hash values
$\boldsymbol{y}(\boldsymbol{x})$	RBF outputs
$\boldsymbol{z}(\boldsymbol{x})$	Network outputs

z_{c1}	Largest network output
z_{c2}	Second largest network output

Chapter 1

Introduction

1.1 Background

The rapid growth of storage technology and computer networks has brought the opportunity for researchers to get involved in the processing of large-scale stream data which consists of collecting data in real time, storing, mining and analyzing the knowledge from data. Along with the development of the internet and sensor technologies, a large amount of data is continuously generated in our daily life and such data are typically provided as a time series of large data chunk, so-called *big stream data*. For such a large-scale stream data, it is usually difficult to learn all the data given at the same time or within a short period. Stream data are characterized as an unlimited sequence of data which are given continuously in a short time and whose data distribution is changed over time[1)]. There are at least three major issues to be solved: (1) system that capable to monitor for each second of activity continuously, (2) the growing number of data and (3) how to learn fast. To solve the first issue, it seems that human resource is not an option since human need to rest and could not maintain our focus for a long time, besides our limit to process complicated calculation with high speed. Thus, machine learning is a good alternative to replace human resource which is capable to work non-stop with high speed and with a consistent working performance. Since machine learning is used to solve the first issue, the solution for the other two remaining issues must be related to machine learning too. As for the second issue, it can be solved by learning incrementally using machine learning approach which able to keeps only important information on the previously given data.

Whereas, to cater third issue, the computational time need to be reduced by removing unimportant calculations. The redundant calculations to be discarded should be selected carefully so that the performance would not degrade much. Therefore, in this dissertation, a novel technique to select essential data and to learn Radial Basis Function (RBF) bases locally is introduced which based on Locality Sensitive Hashing (LSH) approach.

Among the research area within the scope of large-scale stream data stream study is cybersecurity application that meets the criteria of stream data. Those criteria are the generation of unlimited data and the characteristics that represent a class label are constantly changing when a cyber criminals attack's techniques evolved. Since more and more people rely on electronic gadgets (i.e., computers, smartphones and tablets) and internet connections, there are those who try to manipulate the advantages of these technologies for *cyberbullying*. Because of this reason, the importance of cyber security research has raised up in order to avoid users being deceived by the recent tactics of cyber criminals. For example, the spammers send spam emails that contain malicious software to steal important data from a particular individual or organization and the attackers send an attack toward certain websites via Distributed Denial of Services (DDoS) to prevent other users from accessing certain website. Both examples of cyber security applications have been developed in this study to detect any harmful input (i.e., malicious spam emails and DDoS backscatter) that would contribute to adverse effect on the users.

Technology without borders has now becoming preference for human to communicate to each other. It is because this technology is not only very fast, but also is easy to be used to facilitate our daily tasks. One of striking example is email application that functions as virtual letter. Although the letter was still used to send original copy of document, usually an email would be sent prior to the original copy. Email is not just used by those registered companies and organizations, in fact email is also used by individuals for personal usage. Due to

the efficiency of email application, most companies and organizations own their internal email system to distribute important information from board of executive to their subordinates without keying in one by one email addresses. It is quite problematical task if the organizations have hundreds of staff as their subordinate. Apart of simplicity and fast operation offered by email application, an email address has also becoming an identification (ID) of Internet users in cyber world. For example, to sign up a social media like *Facebook*, an email address is required as the ID to distinguish every single user of *Facebook*. Since the email application is used by almost all Internet users, thus this application is manipulate by a group of individuals to commit cybercrime. As far as I know, the studies which conducted to combat cybercrime through the medium of email only focused on spam email detection and also malicious emails detection targeted at a specific individual known as Targeted Malicious Email (TME). To the best of my knowledge from previous studies, there are still no studies on the detection of malicious spam email in general, which does not only targeted at TME, but also all email users. Thus, the development of malicious spam email detection system has become another novelty in this dissertation.

Besides the email application, another important application is website or Internet browsing. Currently, the websites are officially being used to represent some organizations by explaining their objectives, vision and general overview. In addition, the websites also usually equipped with online services to assist the deal between the customers and the service provider. For instance, the bank institutions provide the online banking for transferring money using Internet and also online shop for purchasing desired items without having to go to the shop physically. Thus, to attack a specific websites would means that the attacker has intent to bring down the high-profit company and high-profile organizations through cyberattacks or known as DDoS attack. As a consequent of DDoS attack, the website is likely to face server-down in which the customers or Internet users are not able to access the website that needed at specific period. This can

lead to the loss of millions of dollars for a large corporation if the traffic on the company network is not resolved quickly. DDoS attacks can be through *labeled packet* from Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) using Port 80 and Port 53, respectively. Generally, it is very difficult to identify DDoS attacks from other combination of protocols and port numbers other than TCP Port 80 (80/TCP) and UDP Port 53 (53/UDP), because there are no specific rules to differentiate the network traffic of protocol and port number excluding 80/TCP and 53/UDP. Normally, an expert is needed to analyze the traffic of those *unlabeled packets*. Thus, the DDoS backscatter detection system has became my third novelty in this dissertation.

To process large-scale stream data, it is important to choose a suitable learning algorithm that is capable to learn in real time. Incremental learning has an ability to process large data[2)] in chunk and update the parameters after learning each chunk. Such type of learning, keep and update only the minimum information on a classifier model. Therefore, it requires relatively small memory and short learning time. On the other hand, batch learning is not suitable because it needs to store all training data, which consume a large memory capacity. Due to the limited memory, it is certainly impossible to process large-scale stream data online using the batch learning. Therefore, the learning of large-scale stream data should be conducted incrementally.

Another criteria to be considered for learning large-scale stream data is to select essential data to be learned. The selection of essential data is often conducted based on the margin of classifier outputs. Even if a large number of data are given as a data sequences, there is no guarantee that all the data are useful for learning and can efficiently improved the accuracy of test data. Besides, in supervised learning, a large training data correspond to a larger number of data to be labeled by a supervisor. Hence, selecting the most useful training data can reduce not only the amount of supervision required for effective learning[3)], but also the processing time due to the decreasing number of training data.

Although Nearest Neighbors (NNs) search plays an important role in various machine learning areas, it has a limitation when a high dimensional data is given. To overcome this problem, LSH provides an alternative to process high-dimensional data by approximating similarity search problem[4, 5, 6, 7, 8]. In a general hashing, every input is assigned to a different bucket to avoid collision, while inputs of LSH are assigned to the same bucket when the input data are close to each other. Consequently, LSH is able to search similar data faster because LSH can find the nearest neighbors in linear time as opposed to the quadratic pair-wise approach of the conventional nearest neighbor, which give advantage to high dimensional data and large-scale data. However, a major limitation of LSH is originated from its large memory consumption; that is, to keep both high recall and high precision, a large hash table is needed.

Besides the challenge of processing stream data, cybersecurity field is facing another serious issue which is to obtain cybersecurity data[9] that can be used for developing detection system and evaluating the effectiveness of the proposed method. To overcome this, White House Cyberspace has given some recommendations to the action plan in the Cyberspace Policy Review[10], which is one of the recommendations is to set up a database for cybersecurity data. The database is used to assist researchers whom are struggling in the challenges of cybersecurity attacks that are continuously evolved in a short time. This idea had inspired RTI International and DHS Science & Technology (S&T) Directorate to established a sharing medium for different types of security-related datasets based on trusted framework, namely the Protected Repository for Defense of Infrastructure Against Cyber Threats (PREDICT). In addition to providing secure and centralized data, PREDICT also ensure the confidentiality and privacy of those who contribute data to the system. This effort had given a drive to the cybersecurity researchers to design robust software to counter cyber criminals.

Although PREDICT is capable to provide a large number of cybersecurity data, in reality (1) it is difficult to assign a class label to each of data, and (2) the

datasets usually have unbalanced number of positive and negative examples. The labeling process for those unbalanced data is very challenging especially when a new campaign is launched rapidly. The experts are required to analyze the new form of attack which requires a high focus and a quite amount of time. In general, the unbalanced datasets have a higher proportion negative example (normal data) than the positive example (sample to be detected i.e., malicious spam email and DDoS attack). Therefore, in this dissertation, another method to acquire the data is introduced. In this study, the autonomous data collection system which is used has a little difference compared to PREDICT system. Instead of obtaining the data from particular individuals and networks like PREDICT, the data for this study were obtained from servers that are collecting suspicious data or data that are not only failed to be received by recipients, but also unable to be sent back to the sender. Thus, it is expected that there are more positive examples able to be obtained using this approach compared to that in PREDICT.

For the detection malicious spam email application, the servers that collect double-bounce email are used to obtain malicious spam email data, while darknets are used to collect DDoS backscatter data. The double-bounce email happens when the email sent by the sender is not received by the recipient. It might be due to the owner of the email address has already deactivate the email address or it might be because of the recipient's inbox is full and overload. This requires the email to be returned back to the sender. However, since the attacker wanted to hide his/her identity, the sender's email address is often faked, and thus the email could not be returned back to the sender. This unreceived email and unreturned email which is described as bouncing two times, would be collected by a particular server. Double-bounce email consists of spam emails that usually have a malicious intention. Besides the spam emails, there are also a small portion of normal email which is gathered by double-bounce email server, as a result of unintentionally filled in wrong recipients' email addresses and senders' email addresses. A similar mechanism of double-bounce email can be seen in the backscatter packet data

collected by darknet. It is obvious that the packets data received by darknets would also hold a malicious intention, because in reality, there is no purpose to communicate with IP address without any host. Since the target server's source IP address is not connected to any host, the data packet must be returned to its destination IP address. However, due to the destination IP address is faked and could not be sent back to the destination IP address, the backscatter effect is happening.

In this study, an autonomous labeling system is introduced and used in the propose detection system. Due to the positive samples were more numerous in this autonomous data collection, the process of analyzing the group to be detected (positive examples) is more convenient and reassuring. To assign a class label, rule-based approach is used in which the behaviors of malicious spam email and DDoS attack are analyzed using training data. For the malicious spam email detection system, SPIKE system has been used to label the double-bounce email, while for the DDoS attack, the analysis of traffic features has been used to assign a class label to darknet packets. This approach provides an autonomous labeling system that can reduce the usage of manpower.

This chapter is organized as follows. In Section 1.2, a brief explanation are given on basic theories of machine learning. Previous studies on conventional incremental classifier, selection strategies for data and RBF bases, and general LSH are discussed in Section 1.3. The general overview of the proposed methods in Chapter 2, 3 and 4 are discussed in Section 1.4, Section 1.5 and Section 1.6, respectively. Section 1.7 shows the evaluation methods that are in this dissertation. Finally, the thesis outlines and objectives are explained in Section 1.8.

1.2 Basic Theories of Machine Learning

1.2.1 Type of Learning

The machine learning field has become one of the most famous method to solve classification and detection problem for various application. Some classi-

fication task is easy for human such as differentiating images like cat and dog. However, most of the classification task require a full attention and generally are very complex to be done by human due to our limitation to analyze such a high-computational processes. In machine learning, training data are used by the classifier model to learn and identify the class boundary which can separate different classes. Thus, when new data namely test data are given to the system, the updated model can provide class label of all test data.

Classification or detection task is a process to obtain the class label of unseen data that has not been learned by classifier model or the developed system. To perform a classification task, there are two different approaches to be chose which are through supervised learning or through unsupervised learning. Figure 1.1 shows the difference of these types of learning. The most noticeable difference of both learning type are the necessity of the class label during learning phase. For supervised learning, the class labels served as a supervisor to find the class boundary (see Fig. 1.1 : left side). Meanwhile, unsupervised learning does not require class label. Thus, the classification of unsupervised learning is made using clustering technique (see Fig. 1.1 : right side) in which the system will grouped the nearby data as a class.

1.2.2 Learning Scheme

Figure 1.2 illustrates the learning and evaluation procedures using training data and test data, respectively. The upper-side of this figure shows the illustration of large-scale stream data in which the incoming data are given in chunk of different sizes at a spesific time sequences. Here, initial training data is denoted as t_0, training data as t_i and test data as t_{i+1}. Each training data consists of feature vectors $\boldsymbol{x}$ and target vector $\boldsymbol{d}$. Whereas, test data only have feature vectors $\boldsymbol{x}$ as input to the classifier model. As mention in Section 1.2.1, the training data would identify the class boundary and update the previous parameters of the model. This updated classifier model is used to classify the test set which is

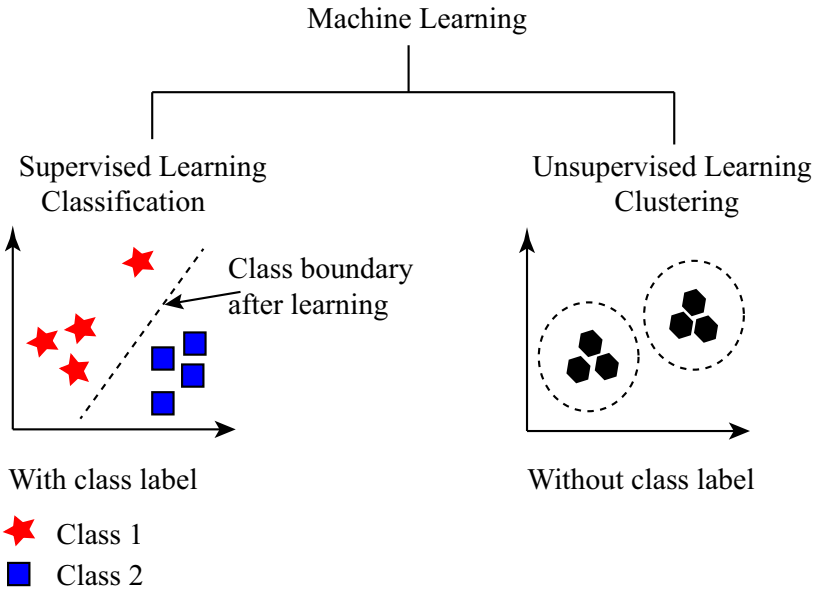


Figure 1.1 The differences between supervised learning and unsupervised learning.

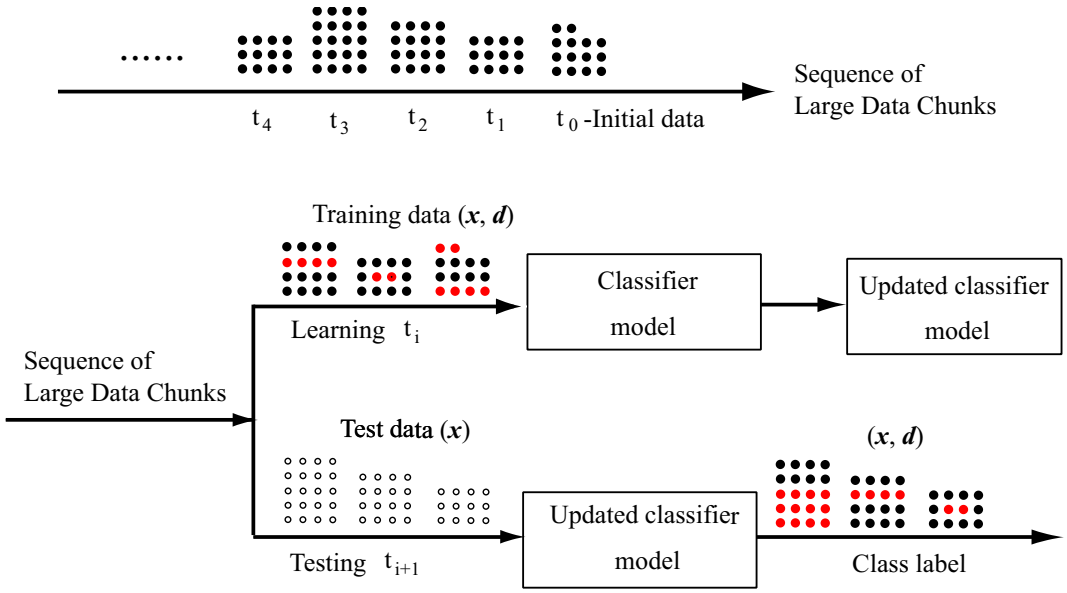


Figure 1.2 Training data and test data of large-scale data sequences.

unseen by the model.

There are two types of learning scheme which are incremental learning and batch learning. With the former type of learning scheme, the learning is carried

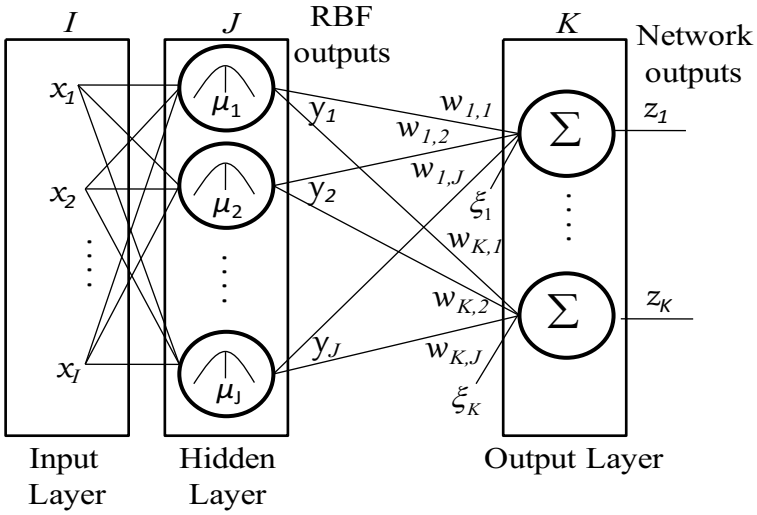


Figure 1.3 Radial Basis Function Networks (RBFN)

out chunk by chunk without storing previous data. For the first learning stage, initial structure is developed using initial training data with chunk-t_1 as test data. The learning continues by using the following chunk such as chunk-t_1 as training data and chunk-t_2 as test data, until all data are finished. With the later type of learning scheme, the learning is carried out using a group of chunk data. For instance, chunk-t_0 to chunk-t_2 as training data and chunk-t_3 to chunk-t_4 as test data.

1.3 Previous Studies: Conventional Classifier, Data Selection Method, RBF Bases Selection and Fast Near Neighbor Search

1.3.1 Modified Resource Allocating Network (RAN)

Figure 1.3 illustrates the structure of the RAN classifier. Let I, J, and K be the numbers of inputs, RBF outputs, and network outputs, respectively. When inputs $\boldsymbol{x} = \{x_1, \cdots, x_I\}^T$ are given to RAN, the RBF outputs $\boldsymbol{y}(\boldsymbol{x}) = \{y_1(\boldsymbol{x}), \cdots, y_J(\boldsymbol{x})\}^T$ and the network outputs $\boldsymbol{z}(\boldsymbol{x}) = \{z_1(\boldsymbol{x}), \cdots, z_K(\boldsymbol{x})\}^T$ are

calculated as follows:

$$y_j(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_j\|^2}{\sigma_j^2}\right) \quad (j=1,\cdots,J) \tag{1.1}$$

$$z_k(\boldsymbol{x}) = \sum_{j=1}^{J} w_{kj} y_j(\boldsymbol{x}) + \xi_k \quad (k=1,\cdots,K) \tag{1.2}$$

where $\boldsymbol{c}_j = \{c_{j1},\cdots,c_{jI}\}^T$ and σ_j^2 are the center and the variance of the jth RBF unit, respectively; w_{kj} and ξ_k are the connection weight from the jth RBF unit to the kth output unit and its bias, respectively.

Algorithm 1 shows the learning algorithm of a modified Resource Allocating Network (RAN). The learning of RAN is divided into two phases: the addition of RBF units (Lines 4-5 and Lines 16-17) and the update of connection weights between hidden and output units (Lines 9-13). Once RBF units are allocated, the centers are fixed afterwards. The connection weights $\boldsymbol{W} = \{\{w_{jk}\}_{j=1}^{J}\}_{k=1}^{K}$ of the modified RAN are updated by solving the following equation[11]:

$$\boldsymbol{\Phi W} = \boldsymbol{D} \tag{1.3}$$

where $\boldsymbol{\Phi}$ is a matrix of RBF outputs and $\boldsymbol{D}$ is a target matrix. Suppose that a training data $(\boldsymbol{x}, \boldsymbol{d})$ is given and Jth RBF centers $(\boldsymbol{c}_j, \boldsymbol{w}_j)$ $(j = 1,\cdots,J)$ have already been allocated. In order to suppress the interference caused by the incremental learning, the modified RAN is trained with some RBF centers as well as a new data. Therefore, a target matrix $\boldsymbol{D}$ in Eq. (1.3) is formed as follows: $\boldsymbol{D} = \{\boldsymbol{d}_x, \boldsymbol{d}_{c1},\cdots,\boldsymbol{d}_{cJ}\}^T$, while a matrix of RBF outputs $\boldsymbol{\Phi}$ is given by $\boldsymbol{\Phi} = \{\boldsymbol{y}(\boldsymbol{x}), \boldsymbol{y}(\boldsymbol{c}_1),\cdots,\boldsymbol{y}(\boldsymbol{c}_J)\}$. To obtain $\boldsymbol{W}$ in Eq. (1.3), Singular Value Decomposition (SVD) is often used, and the solution is given by the following equation:

$$\boldsymbol{W} = \boldsymbol{V}\boldsymbol{H}^{-1}\boldsymbol{U}^T\boldsymbol{D} \tag{1.4}$$

where $\boldsymbol{V}$ and $\boldsymbol{U}$ are orthogonal matrices and $\boldsymbol{H}$ is a diagonal matrix.

Since the computational complexity of SVD is $O(J^3)$, the number of RBFs is a key to reduce the learning time. Let us introduce the selection of active RBF in

the modified RAN. Algorithm 1 shows a pseudo-code of the RAN learning with RBF selection. As seen at Line 8 in Algorithm 1, when the connection weights are updated, only a part of RBFs are selected and used to define the matrix of RBF outputs $\boldsymbol{\Phi}'$. The matrix $\boldsymbol{\Phi}'$ is decomposed into the orthogonal matrices $\boldsymbol{V}'$, $\boldsymbol{U}'$ and a diagonal matrix $\boldsymbol{H}'$ by using SVD. Then, the connections weights $\boldsymbol{W}'$ are updated for the selected RBF units, while the connections for the other RBF units are not changed. An example of adding new RBF bases in RAN classifier and the calculation of RBF output, network output and connection weights is shown in Table 1.1.

In order to suppress the catastrophic interference, it is empirically known that some RBF units should be learned along with a given training data, so that the previous memories are not forgotten when learning new data. In conventional RAN, the problem is solved by learning local region which is approximated using C^1 continuous polynomial without calculating the RBF outputs[12]. Whereas, the same goal is achieved by another variant of RAN, RAN *with Long Term Memory* (RAN-LTM) where it carried out the learning with memory items that are stored in LTM[13, 14, 15]. Nevertheless, different approach are used in this study which can be easily implemented by using LSH where active RBF units are able to be selected. This will be described in the later chapter.

1.3.2 Margin-based data selection

One of the most frequently used approaches in active learning for data selection is based on output margins. The calculation of an output margin Δz is conducted by subtracting the second largest network output z_{c_2} from the largest network output z_{c_1} as follows:

$$\Delta z = z_{c_1} - z_{c_2} = \underset{k' \in \{1,\ldots,K\}}{\operatorname{argmax}} z_{k'}(\bar{\boldsymbol{x}}_e) - \underset{k'' \in \{1,\ldots,K\} \setminus k'}{\operatorname{argmax}} z_{k''}(\bar{\boldsymbol{x}}_e). \tag{1.5}$$

where z_{c_1} and z_{c_2} are the largest and the second largest outputs of modified RAN, respectively. Network output value represent the degree of belonging to a specific classes. If the network output of an A output unit is high z_A, it means that

Table 1.1 Example of RAN calculation

Input vector		Target vector			RBF output	Network output			Error	Nearest RBF	New center		New weights			Update weights using Eq. (1.4)		
x_1	x_2	d_1	d_2	d_3	y_j	z_1	z_2	z_3	E	d	c_1	c_2	w_1	w_2	w_3	w_1	w_2	w_3
0.5	0.5	1	0	0							0.5	0.5	1	0	0			
0.5	0.95	0	1	0	0.81	0.81	0	0	1.29	0.45	0.5	0.95	−0.49	0.60	0	2.95 −2.40	−2.40 2.95	0 0

Example of RBF output $y_j(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_j\|^2}{2*\sigma_j^2}\right)$, network output $z_k(\boldsymbol{x}) = \sum_{j=1}^{J} w_{kj} y_j(\boldsymbol{x}) + \xi_k$, error $E = \|\boldsymbol{z}(\boldsymbol{x}) - \boldsymbol{d}\|$ and distance to nearest RBF bases $d = \|\boldsymbol{x} - \boldsymbol{c}^*\|$ calculation:

$\boldsymbol{X}' = \begin{bmatrix} 0.50 & 0.50 & 1 \\ 0.50 & 0.95 & 2 \\ 0.60 & 1.00 & 3 \\ 1.00 & 0.40 & 1 \end{bmatrix}$, $y_1 = \exp\left(-\frac{\sqrt{((0.5-0.5)^2+(0.5-0.95)^2}}{2*0.7*0.7}\right) = 0.81$, $z_1 = \sum\left(\begin{bmatrix} 0.81 \\ 0.81 \\ 0.81 \end{bmatrix} . \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.81 \\ 0 \\ 0 \end{bmatrix}$ $E = \sqrt{\left([0.81-0\ 0-1\ 0-0]. \begin{bmatrix} 0.81-0 \\ 0-1 \\ 0-0 \end{bmatrix}\right)} = 1.29$,

$d = \sqrt{((0.5-0.5)^2 + (0.5-0.95)^2)} = 0.45$

Although $E > 0.5$, the conditions to add new RBF bases are not satisfied because the distance $d^* < 0.45$. Thus, new network output and error are calculated using the updated weight. The weights are updated using SVD.

$\boldsymbol{W} = \begin{bmatrix} 0.60 \\ 0.49 \\ 0 \end{bmatrix}$, $z_1 = \sum\left(\begin{bmatrix} 0.81 \\ 0.81 \\ 0.81 \end{bmatrix} . \begin{bmatrix} 0.60 \\ 0.49 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.49 \\ 0.4 \\ 0 \end{bmatrix}$,E$= \sqrt{\left([0.49-0\ 0.4-1\ 0-0]. \begin{bmatrix} 0.49-0 \\ 0.4-1 \\ 0-0 \end{bmatrix}\right)} = 0.77$.

Since the error $E > 0.5$, a new RBF bases is added with center $\boldsymbol{c}_2 = \{0.5\ 0.95\}$ and weight $\boldsymbol{w}_2 = \boldsymbol{d} - \boldsymbol{z} = \{0-0.49\ 1-0.4\ 0-0\} = \{-0.49\ 0.6\ 0\}$.

Using the initial weight of $\boldsymbol{c_2}$, the overall connection weights are updated using SVD.

$\boldsymbol{W} = \begin{bmatrix} 2.95 & -2.4 \\ -2.4 & 2.95 \\ 0 & 0 \end{bmatrix}$.

0.6	1	0	0	1	$y_1 = 0.77$ $y_2 = 0.99$	−1.11	1.07	0	1.47	0.11	0.6	1	−0.05	0.5	0.46	2.63 −0.82 −1.27	−0.82 0.48 0.65	−1.27 0.65 0.89
1	0.4	1	0	0	$y_1 = 0.77$ $y_2 = 0.57$ $y_3 = 0.59$	0.81	0.03	−0.08	0.21	0.51						3.08 −4.6 2.15	−1.11 5.53 −4.17	−0.89 −4.88 6.11

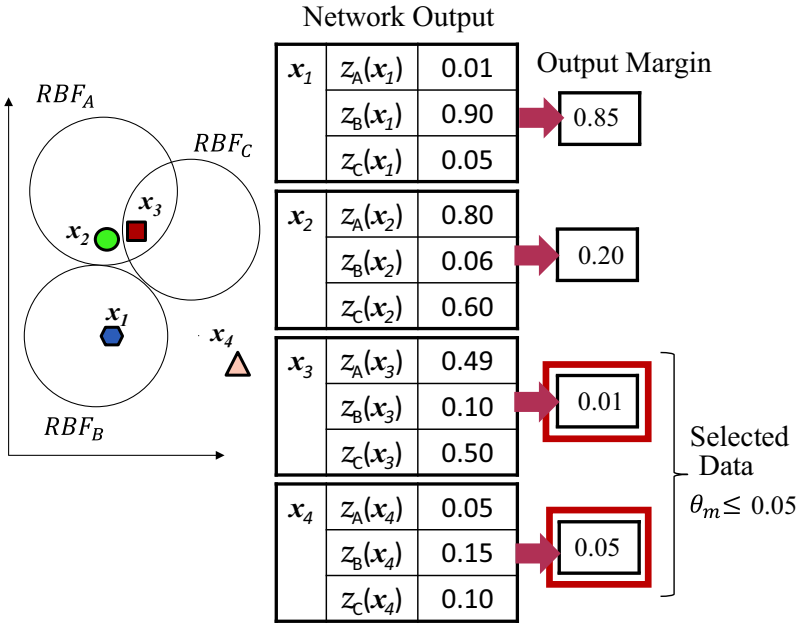


Figure 1.4 Margin-based data selection.

the data is highly belongs to the A class. This data will give a high output margin because the second highest network output is quite low. Compare to another data that does not dominant to any classes where both the highest and the second highest network outputs does not differ much, the output margin will be quite small. Generally, data with low network outputs should be selected to be learned so that the classifier can classify correctly by modifiying the weight. Thus, if the threshold of the margin θ_m is set to be very small, data that equal or lower than the assigned margin threshold $\Delta z \leq \theta_m$ would be selected. By doing this, the number of training data to be learned by RAN are able to be reduced.

For instance (see Fig. 1.4), four training data are given which are $(\boldsymbol{x}_1, \boldsymbol{d}_B)$, $(\boldsymbol{x}_2, \boldsymbol{d}_A)$, $(\boldsymbol{x}_3, \boldsymbol{d}_C)$ and $(\boldsymbol{x}_4, \boldsymbol{d}_C)$. The network output are calculated using Eq. (1.2). If a threshold θ_m is set to 0.05, only $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ will be selected. As shown in Fig. 1.4, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are correctly classified to their given class where the network outputs at the same class (z_B for data $\boldsymbol{x}_1$ and z_A for data $\boldsymbol{x}_2$) are high. Whereas for $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$, since the network output does not dominant to any class which indicate by a low output margin, these data will be selected to be trained

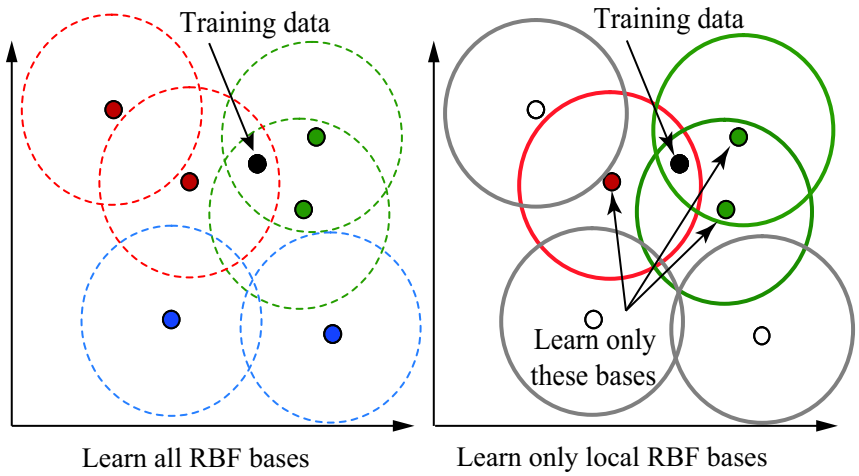


Figure 1.5 The illustration of global-RAN and local-RAN.

by classifier model.

1.3.3 RBF Selection

The training data after data selection and RBF bases are used to calculate the connection weight as shown in Algorithm 1. Since the calculation of SVD to solve the linear equation in Eq. (1.3) require a lot of time, by reducing the number of RBF in the calculation will shortened the processing time. With this general idea, the RBF bases selection are proposed to be included after data selection. Basically Algorithm 1 shows the step of learning RAN. The difference between global-RAN and local-RAN is the RBF bases that is used to calculate connection weight. In global-RAN, all RBF bases are used to update the weight. Compared to local-RAN, only RBF bases that are near to the training data are considered during the calculation of weight. Figure 1.5 demonstrates the difference of global-RAN and local-RAN, where only RBF bases that is near to the training data are chosen. The RBF bases that is near to the training data will give more influence to the network output value compared to RBF bases that is far. In Eq. (1.1) shows that the farther the RBF bases or the value of μ, the smaller the value of RBF output and will not give much effect to the network output. Due to this reason, only near RBF bases are selected to be learn since the elimination does

not change the final result. Furthermore, the learning time is able to be reduced if not all RBF bases are used.

1.3.4 Locality Sensitive Hashing

From large stream data, it is generally possible to obtain similar data in a different data sequence. Such data are redundant to a system because training similar data will not improve the performance of a system. Based on this idea, only data that are new to a system and have not been trained well should be selected. To achieve this, similar data are needed to be search and removed from training data. Unfortunately, the well-known nearest neighbor method is not sufficient to find similar data in high-dimensional data. The larger the dimensions are, the harder it will be to calculate the nearest neighbor due to the *curse of dimensionality*.

Given that only approximate value of the similarity are needed, LSH provides an alternative to process high-dimensional data by approximating similarity search problem[4, 5, 6, 7, 8]. So far this method has only been applied to quickly find similar entries in large databases such as images, videos and songs. The LSH method used randomized algorithm which does not guarantee an exact answer but instead provides a high probability guarantee that it will return the correct answer or close to it[16]. In a general hashing, every input is assigned to a different bucket to avoid collision, while inputs of LSH are assigned to the same bucket when input data are close to each other. Consequently, LSH is able to search similar data faster because LSH can find the nearest neighbors in linear time as opposed to the quadratic pair-wise approach of the conventional nearest neighbor. This will give an advantage to high dimensional data and large-scale data. However, a major limitation of LSH is originated from its large memory consumption; that is, to keep both high recall and high precision, a large hash table is needed.

1.4 The Proposed Classifier Model

In Chapter 2, a new incremental learning algorithm of Radial Basis Function Network (RBFN) is proposed to accelerate the learning for large-scale stream data. Along with the development of the internet and sensor technologies, a time series of large data chunk are continuously generated in our daily life. Thus it is usually difficult to learn all the data within a short period. A remedy for this is to select only essential data from a given data chunk and provide them to a classifier model to learn. In the proposed method, only data in untrained regions, which correspond to a region with a low output margin, are selected. The regions are formed by grouping the data based on their near neighbor using LSH, in which LSH has been developed to search neighbors quickly in an approximated way. As the proposed method does not use all training data to calculate the output margins, the time of the data selection is expected to be shortened. In the incremental learning phase, in order to suppress catastrophic forgetting, LSH technique is also exploited to select neighbor RBF units quickly. In addition, a method to update the hash table in LSH is proposed so that the data selection can be adaptive during the learning. From the performance of nine datasets, it is confirmed that the proposed method can learn large-scale stream data quickly without sacrificing the classification accuracies. This fact implies that the data selection and the incremental learning work effectively in the proposed method. The data selection based on output margins is one of the most frequently used approaches in active learning. The conventional RAN-LTM method requires the calculation of output margins for every training data candidates. Hence, such a margin-based method results in increasing the number of RBF units and consequently increasing the computational time. Therefore, as more data are given at a time, real-time learning becomes more difficult.

In the proposed method, to avoid the repetition of the output margin calculation, the data are divided into buckets where adjacent data are grouped in the same bucket. To do this, a data selection method using LSH is proposed by com-

bining with the margin-based method, where only unknown data in a new bucket and not well-learned data are learned by RAN, whereas data that are well-learned are ignored during the learning phase. However, by only implementing this idea, the learning time is not drastically reduced because the learning time strongly depends on the number of RBF bases[17]. Therefore, instead of learning all RBF bases, a local learning algorithm is proposed in which only RBF bases with high activations for a training data are selectively learned.

Apart of continuously generated, the distribution of data stream is also constantly changes over time. The changes of data distribution would cause training data to be scattered over the input space and located either at a knowned region or at a new region. A known region is region that has been learned from previous training data. Whereas new region is region that have not been trained. By learning a known and well-learned region repeatedly would not improve the recognition accuracy, thus that training data that lies in that region should be discarded to reduce the number of training data. In this study, data that lies in unknown or not well-learned regions are used for learning.

To decide whether the region is unknown or not, neighbor of the given training data is searched. If the training data has no neighbor or in other word the training data is not surrounded by previous data, that region is defined as unknown. A commonly used distance metric for searching neighbor is Euclidean distance. However, this method is usually computationally expensive, especially for high-dimensional data. Another alternative method to approximate neighbor search in high dimensional spaces is using LSH technique. The LSH method indeed has proven be able to quickly find neighbor or similar objects in a large database.

The key idea of LSH method is to group a nearby data into the same group or bucket. Firstly, the training data is mapped from a high-dimensional data to several low-dimensional subspace which called hash functions. These hash functions are projected from multiple directions and each of them are divided into several partitions with a same width that assigned with a decimal values.

The combination of decimal values from all hash functions would produce a hash value where one bucket is represented by one hash value. Thus, training data which are close to one another would be projected in the same partitions for each hash functions and is assigned with the same bucket.

A data structure is needed to represent a large data input into hash value so that the similarity search would only need a short time. By constructing a hash table, similar data is able to be traced quickly by finding data with the same hash value. Since the hash table allows us to quickly find neighbor data for data selection part, neighbor of RBF bases for learning near RBF bases can be found by reusing the hash table to shorten the learning time in the classifier. Hence, for the purpose of designing a fast learning algorithm, the following two functions are introduced in the previously propose RAN-LTM:

(1) Fast data selection using LSH.

(2) Fast RBF-bases selection using LSH.

1.5 Cyber Security Application (1): Malicious Spam Email Detection System

In Chapter 3, a new online system is proposed to detect malicious spam emails and capable to adapt the changes of malicious URLs in the body of spam emails by updating the system daily[18]. For this purpose, an autonomous system that learns from double-bounce emails and collected at a mail server is developed. To adapt to new malicious campaigns, only new types of spam emails are learned by introducing an active learning scheme into a classifier model. Here, Resource Allocating Network with Locality Sensitive Hashing (RAN-LSH) is adopted as a classifier model with data selection. In this data selection, the same or similar spam emails that have already been learned are quickly searched for a hash table using Locally Sensitive Hashing, and such spam emails are discarded without learning. On the other hand, malicious spam emails are sometimes drastically changed along with a new arrival of malicious campaign. In this case, it is not

Table 1.2 Previous studies on malicious spam email detection.

Definition and Background			
Malicious spam email : Email that are embedded with attachments and/or URLs, which give the oppurtunity for the malware to infect the computer[19]. The global spam rate out of total of email traffic is 60% in 2014, 66% in 2013 and 69% in 2012[20], which shown a gradual decreased every year. Although, it is reported that the spam rate is dropped about 9% from 2012 to 2014, however according to this report[20], the total volume of spam email shows a consistent dropped about 1 billion every year. The main goal of most malicious programs distributed via email is to steal confidential data. However, malware capable of spreading spam and launching DDoS attacks was also popular. The most popular types of malware are now multifunctional: they can steal data from the victim computer, make the computer part of a botnet, or download and install other malicious programs without the user's knowledge[21].			
Related Work			
Author	Proposed Method	Motivation	Advantages of the Proposed Method
Deshmukh et. al.[22]	Focus on feature extraction of TME based on random forest classifier. Consists of five features: (i) Sender, (ii) Subject, (iii) Attachments, (iv) Source, (v) Date and Time)	Number of TME dataset is small compare to spam email.	(i) Random forest classifier is under unsupervised learning, (ii) Dataset used is pre anti-virus (dataset includes only those emails that were passed by a commercial anti-spam system), (iii) Able to hansle unbalanced data
Alazab et. al.[23]	Features such as prevalence, frequency, duration of attacks, and other features are used to group them according to authorship.	Lack of knowledge of cybercriminals, their motivation and their modus operandi leads to many lost battles between attacker and anti-virus researcher.	The information of authorship can be used to recognize future attacks by the same groups of spammers
Song et. al[24]	Spam clustering method based on IP addresses. Consists of two main modules: (i) IP Based Clusterer and (ii) Document Based Clusterer	The fatal weekness of three approaches for clustering spam: (i) email content (mainly words):periodically change the contents of emails[25], (ii) URLs:active period of URLs is extremely short[26] and (iii) domain names:domain names associated with the scams change frequently[27].	The clustering method is more robust to changes of new campaign.

appropriate to classify such spam emails into malicious or benign by a classifier. It should be analyzed by using a more reliable method such as a malware analyzer. In order to find new types of spam emails, an outlier detection mechanism is implemented in RAN-LSH. To analyze email contents, the Bag-of-Words (BoW) approach is adopted to generate feature vectors whose attributes are transformed based on the normalized term frequency-inverse document frequency. To evaluate the developed system, a dataset of double-bounce spam emails is used which collected from March 1st, 2013 to May 10th, 2013. In the experiment, the effect of introducing the outlier detection in RAN-LSH is analyzed. As a result, by introducing the outlier detection, it is confirmed that the detection accuracy is enhanced on average over the testing period.

To my knowledge, there is still no study which focus on malicious spam email in general except the TME that aim for specific targeted people. Malicious spam email detection system in this study does not only focus to certain targeted people, but instead, all the spam email are treated as potential malicious spam email that can give harm to the email users' confidential data. A general literature review on TME is shown by Table 1.2.

1.6 Cyber Security Application (2): DDoS Backscatter Detection System

In Chapter 4, a fast detection system for DDoS backscatter using packets from various protocols and port numbers is proposed, which is not restricted to only the following two types of packets that can be labeled with simple rules called "*labeled packet*": Transmission Control Protocol (TCP) Port 80 (80/TCP) and User Datagram Protocol (UDP) Port 53 (53/UDP). Usually, it is not easy to detect DDoS backscatter from the *unlabeled packets*, where an expert needs to analyze packet traffic manually. To deal with *unlabeled packets*, first, the detection system would learns general rules of DDoS backscatter using information from

Table 1.3 Previous studies on DDoS attack.

Definition and Background			
DDoS attacks are launched by generating extremely large volume of traffic that rapidly exhausts available resources of target systems to intentionally disrupt network services[28]. DDoS attacks explicitly threatens the stability of the Internet. Computer Economics[29] estimated that the total economic impact of Code Red was $2.6 billion, and Sircam cost another $1.3 billion.			
Related Work			
Author	Proposed Method	Motivation	Advantages of the Proposed Method
Li et. al[30]	Attack detection can be performed through abnormal behavior identification. The energy distribution is utilized based on wavelet analysis to detect DDoS attack traffic.	Most of the methods introduced so far are based on appearance of DDoS attack, such as spoofed source IP address, bandwidth distribution, attack packet pattern etc. However, attacker can hide the appearance of attack traffic via packet reshaping.	Wavelet analysis is able to capture complex temporal correlation across multiple time scales with very low computational complexity, where a spike is generated when traffic behaviors affected by DDoS attack.
Chen et. al.[28]	Proposed a novel network anomaly detection algorithm (NADA) to detect the abnormal traffic: (i) Pre-process network traffic by cumulatively averaging it with a time range, (ii) AR model which is a time series model is used to generate the prediction of network traffic, (iii) Chaos theory is used to analyze the prediction error, (iv) Train a neural network to detect DDoS attacks.	It is difficult to distinguish between DDoS attacks and normal traffic, specifically to differentiate DDoS attack packets from legitimate abrupt packets.	This learning algorithm can detect an anomaly caused either by bursty legitimate traffic or by DDoS flooding attacks.
Zhang et. al.[31]	Propose an advanced entropy-based (AEB) scheme, which divides variable rate DDOS attacks into different fields and treats each field with different methods.	The key challenge of entropy-based DDOS detection is how to determine the most suitable threshold to detect DDOS attacks accurately.	Has significant better performance on distinguishing waves of legal traffic and flash crowds from Low-rate Denial of Service (LDOS) attack. Advantages: simple calculation, high sensitivity, low false positive rate, no need additional network traffic and device.

80/TCP and 53/UDP. After the learning process, the generalized detection system is used to detect the DDoS backscatter from *unlabeled packets*. This detection system consists of two main modules which are pre-processing and classifier. In the pre-processing module, the incoming packets are transformed into feature vectors. As for the classifier module, since it is important to detect DDoS backscatter from all protocols as early as possible, RAN with data selection is used. Using this classifier, the learning time is shorten because the classifier only learns essential data. Here, essential data means the data located in "not well-learned" regions, in which the classifier could not give trustable predictions. To quickly search for the regions closest to given data, the well-known LSH method is used.

The performance of the proposed detection system is evaluated using 9,968 training data from *labeled packets* and 5,933 test data from *unlabeled packets*. They are collected from January 1st, 2013 until January 20th, 2014 at National Institute of Information and Communications Technology (NICT), Japan. The results indicate that the detection system can detects the DDoS backscatter with high detection rate within a short time.

Table 1.3 shows the previous works on DDoS attack detection system which uses different approaches than the proposed system in this study, as an overview to the existing method by the other researchers.

1.7 Performance Evaluations

The accuracy rate is a standard measure to evaluate the performance of a classifier for a balance dataset. However, in this study, the data from one of two classes is exceeded the number of the other class. Besides that, it is important to evaluate the performance regarding the capability of the classifier to classify correctly the positive class (malicious email or DDoS backscatter) and at the same time able to differentiate the negative class (non-malicious email or non-backscatter). Thus, it is not appropriate to evaluate the detection of the proposed detection system using the accuracy rate alone. Therefore, in this study, four

Table 1.4 The notation of TP, FP, FN and FP.

		True class label	
		Malicious	Non-malicious
Predicted	Malicious	TP	FP
	Non-malicious	FN	FP

evaluations are used which are: accuracy rate, recall rate, precision rate and F1 measure rate. The evaluations can be calculated as follow:

$$Accuracy\ rate = \frac{TP + TN}{P + N} \tag{1.6}$$

$$Recall\ rate = \frac{TP}{TP + FN} \tag{1.7}$$

$$Precision\ rate = \frac{TP}{TP + FP} \tag{1.8}$$

$$F1\ measure = 2.\frac{Precision\ .\ Recall}{Precision + Recall} \tag{1.9}$$

where the definition of True Positive (TP), False Positive (FP), False Negative (FN) and False Positive (TN) are simplified in Table 1.4.

The experiments are carried out on a personal computer with Intel Core Quad (3.4 GHz) CPU and 32 GB main memory. To evaluate the learning time, the computational costs are estimated based on the actual learning time measured by the Matlab stopwatch timer function.

1.8 Thesis Outlines and Research Objectives

The main purpose of this dissertation is to solve the classification problem of large-scale stream data which focus on cybersecurity application. In order to ensure fast and accurate learning, the data selection mechanism should be introduced into a classifier model to find such essential data from a given chunk of data.

The rest of this dissertation is organized as follows:

Chapter 2: A Fast Online Learning Algorithm of Radial Basis Function Network with Locality Sensitive Hashing

This chapter contains related works on RAN and the proposed classifier model which is the extended RAN classifier embedded with LSH-based data selection and RBF bases selection namely RAN-LSH. RAN-LSH is able to accelerate the learning time and it is suitable to be used in large-scale stream data processing.

Chapter 3: An Online Malicious Spam Email Detection System Using Resource Allocating Network with Locality Sensitive Hashing[32)]

This chapter contains related works on double-bounce email and the proposed detection system which is malicious spam email detection system. This detection system uses RAN-LSH classifier with outlier detection. It is able to learn quickly and also capable to reduce the number of training data that need to be labeled using SPIKE system by introducing outlier detection.

Chapter 4: Distributed Denial of Service (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection[33)]

This chapter contains related work on darknet and the proposed detection system which is DDoS backscatter detection system. The RAN-LSH classifier uses an improved condition to select essential data. This detection system is capable to select a small number of essential data compared to previous RAN-LSH.

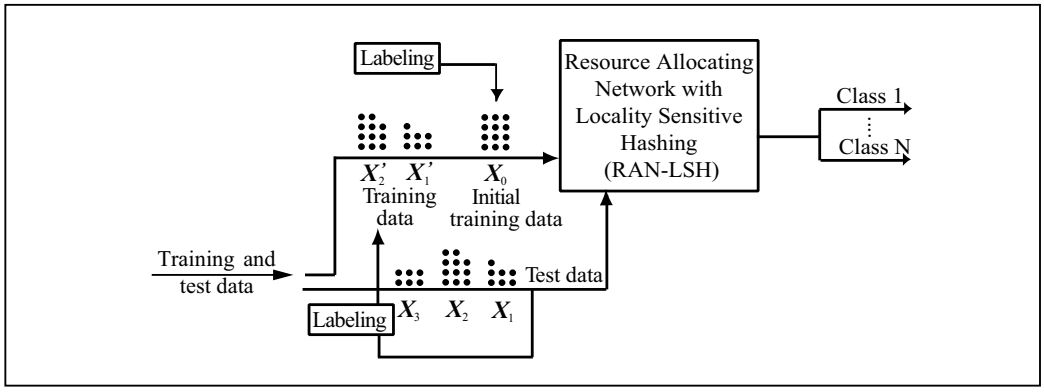
Chapter 5: Conclusions

This chapter contains the summary of the dissertation and future works of this study.

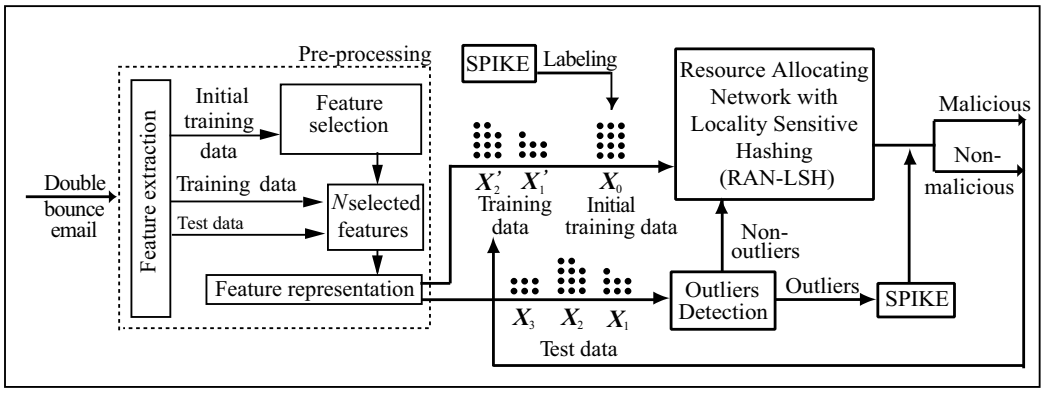
The objectives of each chapter and overall network structure of proposed system are shown by Table 1.5 and Fig. 1.6, respectively.

Table 1.5 Thesis outlines.

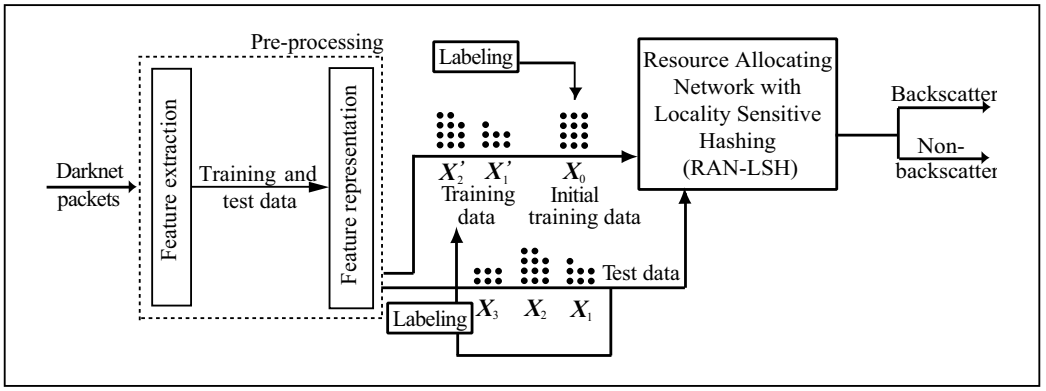
Chapter #	Project Purpose	Objectives	Dataset
Chapter 1	P1.1) Introduction of an incremental classifier and a method to quickly search similar data.	O1.1) Eloborating RAN and LSH.	
Chapter 2	P2.1) Accelerate the incremental learning of large-scale stream data. P2.2) Evaluate the performance of the proposed model in general.	O2.1) Proposed incremental classifier model with data selection namely RAN-LSH. O2.2) Introducing margin flag F^M as follows: $$F_e^M = \begin{cases} 1 & (\Delta z(\boldsymbol{x}_e) > \theta_m) \\ 0 & (\text{otherwise}) \end{cases}$$	Nine UCI datasets: (i) Hiragana, (ii) Adult, (iii) Shuttle, (iv) Bank, (v) Pendigits, (vi) Letter, (vii) MAGIC gamma telescope, (x) Optical digits and (xi) Semeion
Chapter 3	P3.1) Constructing a detection system using autonomous data collection and labeling. P3.2) Accelerate the incremental learning of malicious spam email detection.	O3.1) Collecting double-bounce email and labeling using SPIKE. O3.2) Learning using RAN-LSH. O3.3) Introducing the following two flags: margin flag F^M and outlier flags F^O which calculated as follow: $$F_e^M = \begin{cases} 1 & (\Delta z(\boldsymbol{x}_e) > \theta_m) \\ 0 & (\text{otherwise}) \end{cases}$$ $$F_e^O = \begin{cases} 0 & (F_e^M = 0) \\ 0 & (\theta_m \leq \Delta z(\bar{\boldsymbol{x}}_e) < \theta_o \ \ \& \ \ N_e < \theta_N). \\ 1 & (\text{otherwise}) \end{cases}$$	Double-bounce email.
Chapter 4	P4.1) Constructing a detection system using autonomous data collection and labeling. P4.2) Accelerate the incremental learning of DDoS backscatter detection system.	O4.1) Collecting darknet packets and labeling using analysis of traffic features. O4.2) Learning using RAN-LSH. O4.3) Redefined margin flags F^M as follows: $$F_e^M = \begin{cases} 1 & (\Delta z(\boldsymbol{x}_e) > \theta_m) \ \ \& \ \ (z_{k'}(\boldsymbol{x}_e) > \theta_z) \\ 0 & (\text{otherwise}) \end{cases}$$	Darknet packets.
Chapter 5	O5.1) Conclusions	O5.1) Summarizing all projects in Chapter 2 to Chapter 5.	



(a) Network structure of learning and testing using RAN-LSH on UCI datasets.



(b) Network structure of malicious spam email detection system using RAN-LSH with outlier detection.



(c) Network structure of DDoS backscatter detection system using RAN-LSH.

Figure 1.6 The overall network structure of proposed model.

Chapter 2

Propose Method: A Fast Online Learning Algorithm of Radial Basis Function Network with Locality Sensitive Hashing

2.1 Introduction

Due to the recent development of computer networks and sensor technologies, many application software and devices (e.g. www, twitter, surveillance camera) keep generating large-scale and high-dimensional data continuously as a stream. There are at least two major issues to be solved: (1) how to handle the growing number of data while maintaining high-performance, and (2) how to adapt to dynamic environment quickly. The first issue can be solved by learning only important information incrementally[34]. On the other hand, for the second issue, the computational time needs to be reduced by removing unimportant calculations in the learning.

For the incremental learning algorithm, an extended version of modified RAN classifier[12] is proposed in which LSH is used to select data and active RBF centers. RAN with local learning (i.e., learning only active RBF centers) can effectively suppress the so-called *catastrophic forgetting*[35] by learning some RBF centers with a given training data. However, the learning could be problematic if many data are provided to learn at the same time. In order to ensure real-time incremental learning, a mechanism to select and learn only essential data is solicited in modified RAN. For this purpose, a margin-based data selection has often been adopted in neural networks. In this approach, a training data is given

to a neural classifier and the margins of the two largest outputs are calculated, and then only data with small margins are selected to learn. However, the computational costs to calculate output margins are generally expensive, especially when a large number of data are given as a continuous data sequence.

One way to alleviate the computational burden in the data selection is that the margin calculation is restricted only for training data in "untrained" and "not well-trained" regions. To specify such regions with less computation, LSH[4, 5, 6, 7, 8] is adopted. With LSH, the region of a given training data is quickly specified by transforming into a hash value and the learning status for the specified region is checked in a hash table. If the region is judged as "untrained", the training data would be selected to learn; otherwise, it would be discarded. Since the calculation of hash values in LSH is generally much smaller than that of output margin, the data selection is expected to be fast by using LSH. The idea of using LSH can also be applied to the selection of active RBF units. Since the learning time strongly depends on the number of RBF units[17] in modified RAN, if only RBF units with high activations are selected, the learning of connection weights would be significantly faster. For this purpose, LSH is adopted again to find active RBF units quickly; that is, instead of calculating RBF activations, active RBF units are searched by calculating the LSH distances between the hash values of an input and all RBF centers.

There have been several relevant studies on incremental learning with data selection which have been proposed using different types of classifiers such Support Vector Machines (SVM)[36, 37, 38] and neuro-fuzzy inference system[39]. One of adaptive SVM for classification is proposed in[36] which not only presented data selection in the incremental SVM, but also able to adapt the SVM to changes in regularization and kernel parameters. The training data are partitioned into three following categories: the set which strictly on the margin, exceeding the margin and within the margin, namely margin support vectors, error support vectors and reserve vectors, respectively. Based on this approach, the margin support vectors

and error support vectors are used for learning, whereas the reserve vectors are discarded from training data. With the same approach, Diehl et. al. conducted numerical experiments on Pima Indian dataset regarding regularization parameter C and kernel parameter sigma. In addition to works by Cauwenberghs et al.[36] and Diehl et al.[37], Spüler et al.[38] provides multiclass classification task and uses Platt's probabilistic output to ensure high probability of predicting the test data correctly, based on studies in[40, 41], respectively.

More recently, Subramaniam et al.[39] proposed a metacognitive neuro-fuzzy inference system in which employs a quite similar categorization of training data. In the study, a training data is assigned to either of three following learning strategies: sample deletion, sample learning and sample reserve. The training data are categorized as sample deletion when the data are correctly classified and the estimated posterior probability of the class is above a certain threshold. Since the knowledge has already presented, thus these data are discarded and not used for learning. On the other hand, when training data are misclassified, the data fall into sample learning category. In this category, it is obvious that the system needs to allocate a new rule or update the parameter of the rules or to remove redundant rules, so that the data are able to be classified correctly. If these misclassified data do not fulfill either one of rule growing conditions or parameter update conditions, the data would be reserved for learning at later learning stages.

Although the previous studies mentioned above have utilize the data selection mechanism to discard training data that are classified correctly, to the best of my knowledge, the case of treating the training data as multiple groups of similar data has not been given great attention by the researchers in the past and this motivated the present study. In previous studies, the training data is treated as individual data when assigning category to be learned or not. However, in my approach, the training data are firstly grouped into similar data where the condition of the data whether well-learned or not well-learned is decided by the

flag given by margin flag item in hash table, without repeatedly calculating the condition's equation. Thus, it is expected that the proposed method is able to shorten more learning time instead of only removing unimportant training data.

In this chapter, a fast learning algorithm of modified RAN is proposed which consists of the following two mechanisms: the LSH-based data selection and the LSH-based RBF selection. With the former mechanism, even if a large chunk of training data is given, only essential data are trained in modified RAN. With the latter mechanism, even if the number of RBF units becomes large, the learning time is expected to be fairly constant because the number of selected RBF units are limited to the near RBF. This chapter is organized as follows. Section 2.2 presents its extended model, in which the data selection and the RBF selection are implemented by using LSH. In Section 2.3, the performance of the proposed model is evaluated for several benchmark datasets in the University of California, Irvine (UCI) Machine Learning Repository. Finally, Section 2.4 gives the conclusions of this work.

2.2 The proposed LSH-Based Data Selection and RBF Bases Selection

2.2.1 Learning Algorithm

Only a limited number of training data is assumed available in the initial learning phase. These initial training data are used to decide the structure of a learning model and to set up its appropriate parameters. On the other hand, in the incremental learning phase, it is assumed that a large set of training data are given sequentially and it is too large to learn all data at every time step.

Under such learning environments, training data to be learned in the incremental learning mode should be restricted to only essential ones such as those existing in untrained regions or data that are "not well-trained". For this purpose, the margin-based data selection[42)] can be used to evaluate the status of each training data given, whether they are "well-trained" or not. However, to introduce this

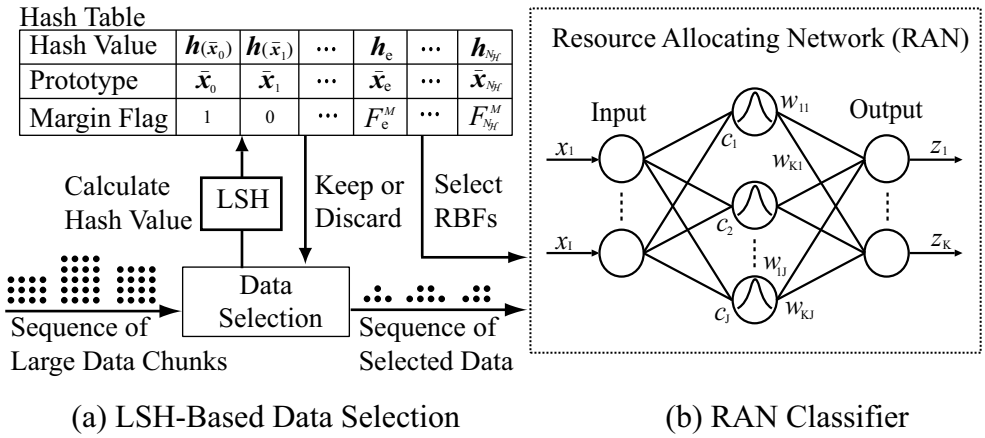


Figure 2.1 Network Structure of RAN-LSH.

margin-based method in modified RAN, the output margins must be calculated for all training data. Obviously, it requires expensive computations for the data selection, and it makes difficult for modified RAN to learn a large data chunk in real time.

To select untrained data from a chunk of given data quickly, LSH is introduced in the data selection. Figure 2.1 illustrates the proposed model called *RAN with LSH-based Data Selection* (RAN-LSH) and Algorithm 2 shows the main flows of the RAN-LSH learning. As seen in Fig. 2.1, RAN-LSH is composed of two parts: (a) LSH-based data selection and (b) RAN classifier. The learning is divided into the following two phases: initial learning phase (Algorithms 2 Lines 1-2) and incremental learning phase (Algorithms 2 Lines 3-10). As mentioned above, in the initial learning phase, only a limited number of data are assumed to be given. They are used to get proper values of the RBF width σ^2 and the number of partitions P, and to create an initial hash table $\mathcal{H}$ (see Line 1 in both Algorithms 2 and 3).

After the initialization, the learning mode is changed to incremental learning. Whenever a new chunk of data is given, it is first forwarded to LSH to calculate a hash value. After updating the hash table, the chunk of data is sent to the data selection part to select only essential data from the given chunk $\boldsymbol{X}$. In order to

make the learning of RAN-LSH faster, a local learning algorithm is adopted in which only a limited number of active RBF units are selected by using LSH and their connection weights are trained incrementally (see Fig. 2.1). Since the time complexity of learning connection weights in modified RAN is the order $O(J^3)$ (J: the number of RBF units)[17], it is expected that the proposed local learning algorithm would also contribute to reducing the learning time in RAN-LSH. These steps are summarized in Algorithm 2 (RAN-LSH learning algorithm).

The detailed procedures in the LSH-based data selection and the whole learning algorithm of RAN-LSH are explained in the following sections.

2.2.2 LSH-based Data Selection

The LSH-based data selection is carried out based on a hash table $\mathcal{H}$ which consists of the following three items: hash value $\boldsymbol{h}(\boldsymbol{x})$, prototype vector $\bar{\boldsymbol{x}}$ and margin flag F^M. A hash function is used to encode an input data into a hash value as an index in the hash table. In LSH, input data that exist within a local region are encoded into the same hash value[43]. A prototype vector corresponds to a typical input data (e.g., mean vector) which represents a local region, and the margin flag means the learning status of the local region: *trained* or *untrained*.

Since the proposed LSH-based data selection is carried out under an incremental learning environment, the hash table $\mathcal{H}$ in the proposed method should be updated on an online basis. For this purpose, the LSH method used here has the following differences from the conventional projection-based implementations of LSH proposed by Andoni et. al.[5]:

(1) A hash function H_i is defined as the projection to an eigenvector obtained by Principal Component Analysis (PCA), instead of that to a random vector.

(2) Only one hash table $\mathcal{H}$ is used and $\mathcal{H}$ is evolved in both size and contents to adapt to time-varying data distributions; that is, the entries in $\mathcal{H}$ is increased in number and updated online. Therefore, it is unlikely that $\mathcal{H}$ covers all areas of an input space, although it may end up covering the

whole space for a long run.

In the next subsections, let us give detailed explanations on the hash encoding, how to update a hash table and the propose data selection algorithm.

Hash Encoding

The basic idea of LSH is to hash high-dimensional input data so that similar data are mapped to the same buckets (subregions of inputs) with high probability. Here, a projection-based LSH is adopted, in which the projection in a hash encoding is conducted for eigenvectors obtained by PCA, instead of random vectors. It is well known that an eigen-subspace spanned by such eigenvectors gives a good approximation to an input space and its approximation error can be easily controlled by the accumulation ratio, which is defined by the ratio of input components in the approximated subspace over those in the whole input space. Therefore, by introducing eigenvectors instead of random vectors in the projection-based LSH, a suitable number of hash functions can be easily determined from a given data distribution.

Let l be the number of eigenvectors which are obtained by setting the threshold θ_a for the accumulation ratio; that is, l is given as the minimum number of eigenvectors whose accumulation ratio is larger than or equal to the threshold θ_a. In addition, let the obtained eigenvectors be $\boldsymbol{U}_l = \{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_l\}$. Then, a projection vector $\boldsymbol{V} = \{v_1, \cdots, v_l\}^T$ of an input data $\boldsymbol{x}$ is obtained by

$$\boldsymbol{V} = \boldsymbol{U}_l^T \boldsymbol{x}. \tag{2.1}$$

In the proposed hash encoding, each projection v_i $(i = 1, \cdots, l)$ is divided into P partitions along the one-dimensional space spanned by the eigenvector $\boldsymbol{u}_i$. Let $H(v_i)$ be a hash function that transforms v_i into an l-series of hash value: $H(v_i) \in \{1, \ldots, P\}_{i=1}^{l}$, and let v_i^- and v_i^+ be the lower and upper values of typical projections v_i on the ith eigenvector $\boldsymbol{u}_i$, respectively. In the proposed RAN-LSH, v_i^+ and v_i^- are determined in the initial learning phase (see Algorithm 3 Line 5),

and they are defined as the minimum and maximum values of the projections for initial training data $\boldsymbol{X}_0$. The hash function $H(v_i)$ is given as follow:

$$H(v_i) = \max\left\{\left\lceil \frac{\min\{\max\{v_i, v_i^-\}, v_i^+\} - v_i^-}{v_i^+ - v_i^-} P \right\rceil, 1\right\}. \tag{2.2}$$

Therefore, a hash value $\boldsymbol{h}(\boldsymbol{x})$ is defined as the following decimal code, which is given by concatenating all the decimal codes of a hash function $H(v_i)$:

$$\boldsymbol{h}(\boldsymbol{x}) = \{H(v_1), \cdots, H(v_l)\} \tag{2.3}$$

as such, an input $\boldsymbol{x}$ is encoded into an l-series of decimal hash vector.

Update of Hash Table

As seen in Algorithm 2 Line 5, the hash values of training data $\boldsymbol{X}$ are first calculated, and $\boldsymbol{X}$ is redefined by adding the hash values $\boldsymbol{h}(\boldsymbol{x}_i)$: $\hat{\boldsymbol{X}} = \{(\boldsymbol{x}_i, \boldsymbol{d}_i, \boldsymbol{h}(\boldsymbol{x}_i))\}_{i=1}^{N_{d'}}$. Then, a hash table $\mathcal{H}$ is updated with the newly defined training data $\hat{\boldsymbol{X}}$. Algorithm 4 shows how to update a hash table $\mathcal{H}$ with $\hat{\boldsymbol{X}}$. In the following, let us assume that a hash table has $N_{\mathcal{H}}$ entries (i.e., $\mathcal{H} = \{\mathcal{H}_e\}_{e=1}^{N_{\mathcal{H}}}$) and the eth entry $\mathcal{H}_e$ is represented by a triplet $\mathcal{H}_e = \{\boldsymbol{h}(\bar{\boldsymbol{x}}_e), \bar{\boldsymbol{x}}_e, F_e^M\}$ where $\bar{\boldsymbol{x}}_e$, $\boldsymbol{h}(\bar{\boldsymbol{x}}_e)$, and F_e^M are a prototype vector, its decimal hash encoding, and the margin flag, respectively.

For each training data in $\hat{\boldsymbol{X}}$, the following procedures are carried out. First, a hash value $\boldsymbol{h}(\boldsymbol{x}_i)$ is searched from $\mathcal{H}$. If $\boldsymbol{h}(\boldsymbol{x}_i)$ is matched with $\boldsymbol{h}(\bar{\boldsymbol{x}}_e)$ in $\mathcal{H}$, it implies that the neighbor area around $\boldsymbol{x}_i$ has already been learned. In this case, the prototype vector $\bar{\boldsymbol{x}}_e$ is updated with $\boldsymbol{x}_i$ as follow:

$$\bar{\boldsymbol{x}}_e' = \frac{1}{N+1}\left(N\bar{\boldsymbol{x}}_e + \boldsymbol{x}_i\right). \tag{2.4}$$

If the margin flag $F_e^M = 0$, the output margin should be checked for the updated prototype $\bar{\boldsymbol{x}}_e'$ using Eq. (1.5). Then, the margin flag F_e^M is updated as follow:

$$F_e^M = \begin{cases} 0 & (\Delta z(\bar{\boldsymbol{x}}_e') \leq \theta_m) \\ 1 & (\text{otherwise}) \end{cases} \tag{2.5}$$

where θ_m is a positive threshold.

If there is no matched entry in $\mathcal{H}$, it implies that a training data $\boldsymbol{x}_i$ has not been learned so far. Thus, the output margin for $\boldsymbol{x}_i$ need to be calculated, so that a margin flag would be assigned for the new entry. Then, a new entry $\mathcal{H}_{N_\mathcal{H}+1}$ is created in $\mathcal{H}$ as follow: $\mathcal{H}_{N_\mathcal{H}+1} = \{\boldsymbol{h}(\boldsymbol{x}_i), \boldsymbol{x}_i, F^M_{N_\mathcal{H}+1}\}$.

Data Selection

Algorithm 5 shows the steps to select essential data. In the data selection, the hash value for each data is examined (see also the upper left of Fig. 2.1(a)). If the margin flag F^M_e is '0', it implies that a training data exists in an untrained region; thus, such a training data should be selected and learned by RAN. On the other hand, if the margin flag F^M_e is '1', it implies that the neighbor region of such a training data is well trained. Therefore, such a data is not considered to be essential for RAN, and it can be discarded to reduce the learning time.

2.2.3 LSH-based RBF Selection

As mentioned before in Section 1.3.1, RBF bases in hidden layer are used to suppress the interference effectively. However, it is not efficient to retrieve and train all RBF bases. One of the reason is the calculation of the SVD in Eq. (1.4) would become heavier for a larger number of hidden units as such, if all RBF centers are used to update the weight. Thus, the processing time would require a longer time. Therefore, the RBF bases are restricted to learn only near RBF centers, in which must satisfy a specific condition (i.e. LSH distance $d^*_j \leq \theta_p$). The algorithm to retrieve only a part of RBF bases is shown in Algorithm 6.

In this study, LSH distance with decimal hash code is adopted, instead of the well-known Hamming distance of binary hash code. This is because, the Hamming distance of binary values is only appropriate to measure distance of projection vector $\boldsymbol{V}$ which is divided into at most three partitions. If larger number of partitions is used, the Hamming distance value would not represents the distance correctly. For example, in the case of $P = 8$, 3-bits binary would be used to

Table 2.1 Hash Table

Index $\boldsymbol{h}(\boldsymbol{x}_e)$	{1, 2, 3}	{2, 2, 2}	{2, 4, 4}
Prototype $\boldsymbol{x}_e$	$\boldsymbol{x}_1$	$(\boldsymbol{x}_2+\boldsymbol{x}_3)/2$	$\boldsymbol{x}_4$
Margin Flag F_e^M	1	1	0

represent each segment/bucket where v_l is located (i.e. segment $\in \{1, \ldots, P\}$). Let say segment = 5 and segment = 6 are represented by binary code 011_b and 100_b respectively. When the Hamming distance between these segments are calculated, it would give 3-bit difference instead of 1-bit difference. Since in this study number of partitions larger than three is used, it is inappropriate to use binary hash code and Hamming distance to find near RBFs.

To find the RBFs neighbor of a training data using LSH, firstly, the hash values of RBFs need to be retrieved from the constructed hash table $\mathcal{H}$ where the $\mathcal{C}_j$th entries in $\mathcal{H}$ are correspond to the hash values of the jth RBFs. Next, the LSH distance d_j^* for each jth RBFs can be calculated as follows;

$$\begin{aligned} d_j^* &= \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{h}(\boldsymbol{c}_j) \\ &= \sum_{i=1}^{l} \Big(\Big| H(v_{x_i}) - H(v_{cj_i}) \Big| \Big). \end{aligned} \tag{2.6}$$

where the distance (using hash values in decimal) between a given training data $\boldsymbol{h}(\boldsymbol{x}_i)$ and all RBF centers $\boldsymbol{h}(\boldsymbol{c}_i)$ for l-projection vector in $\boldsymbol{V}$ are accumulated. Then the RBF centers with LSH distances that are less or equal to θ_p would be selected to solve the linear equation in Eq. (1.4).

To explain the LSH-based data selection and LSH-based RBF bases selection steps using dummy dataset, Figs. 2.2, 2.3 and 2.4 are used. The elaboration of hash encoding, creating hash table, selecting data based on LSH and selecting near RBF bases based on LSH are summarized in Table 2.2.

Table 2.2 Classification using RAN-LSH

Steps #	Elaboration
Given training data $\boldsymbol{X}'$ where the dataset is as follow: $\boldsymbol{X}' = (\boldsymbol{x}$,class label$) = \begin{bmatrix} 0.50 & 0.50 & 1 \\ 0.50 & 0.95 & 2 \\ 0.60 & 1.00 & 3 \\ 1.00 & 0.40 & 1 \end{bmatrix}$.	
Hash encoding	As seen in Fig. 2.2, the projection value v_i is divided into four partitions/segment $P = 4$ with equal size that is: "1", "2", "3" or "4" which projected into $L = 3$ eigenvectors. To describe the proposed method concisely, example in Fig. 2.3 is used. Figure 2.4 illustrates the step to set the index value from the projected data for four examples input. The data are projected into $L = 3$ eigenvectors with 4 partitions $P = 4$ divided into equal size.
Create/update hash table	A hash table is composed of the three items: hash value, prototype and margin flag. As for the first item, the hash values are used as the index of the hash table, which is shown by Fig. 2.4. For the second item, the average of training data in each entries are recorded as prototypes in a hash table. The last item is the margin flag which is set after calculating the output margin. The calculation of an output margin Δz is conducted by subtracting the second largest network output z_{c_2} from the largest network output z_{c_1} using Eq. (eqn:delta$_z$). *Letseeanexampleofinput*1$\boldsymbol{x}_1$ (see Fig. 2.3). By using Eq. (2.2), $H(v_1) = 0$, $H(v_2) = 1$ and $H(v_3) = 2$. Thus, $\boldsymbol{x}_1$ is assigned with hash value $\boldsymbol{h}(\boldsymbol{x}_1) = \{0, 1, 2\}$ as demonstrates in Fig. 2.4. From Fig. 2.4 also we can see that $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ share the same bucket with index of "{2, 2, 2}" while the other two inputs ($\boldsymbol{x}_1$ and $\boldsymbol{x}_4$) are assigned in their own bucket. Table 2.1 shows the constructed hash table from examples data given in Fig. 2.3 and 2.4. As seen in this table, when a new data assigned partition have been registered (in this case $\boldsymbol{x}_3$), a new average value of the region is calculated to be stored as prototype.
LSH-based data selection	Based on LSH-data selection, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are not selected because the margin flag is equal to "1". Besides that, $\boldsymbol{x}_3$ is also not being selected to be learned although the margin flag for $\boldsymbol{x}_3$ is "0". This is because the margin flag value is only be assigned during the first registration in the hash table and during the update of margin flag. Meanwhile, $\boldsymbol{x}_4$ is selected because its margin flag is "0". If we have another new data $\boldsymbol{x}_5$ with the same index as $\boldsymbol{x}_4$, "{2, 4, 4}", $\boldsymbol{x}_5$ will be selected because the margin flag is equal to "0". These selected data will be learned by RAN classifier, where an example of RAN calculation is shown by Table 1.1.
LSH-based RBF bases selection	For instance, let partition P is equal to 16 and the number of eigenvectors i given by $\theta_a = 0.7$ is equal to 4. Thus the projection vector is $\{v_i\}_{i=1}^{4} \in \{1, \cdots, 16\}$. Let say, the hash value of a training data $\boldsymbol{h}(\boldsymbol{x})$ is $\{6, 3, 1, 10\}$. While the hash values of two different RBF centers $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are $\boldsymbol{h}(\boldsymbol{c}_1) = \{5, 12, 7, 1\}$ and $\boldsymbol{h}(\boldsymbol{c}_2) = \{6, 2, 1, 9\}$, respectively. Therefore, the LSH distance d_j^* between a training data $\boldsymbol{x}$ and RBF centers $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are $d_1^* = |6-5| + |3-12| + |1-7| + |10-1| = 25$ and $d_2^* = |6-6| + |3-2| + |1-1| + |10-9| = 2$, respectively. Hence for $\theta_p = 3$, RBF center $\boldsymbol{c}_2$ would be selected. This strategy to select RBF bases is expected to be fast because the hash functions are not recalculated but instead are recalled from the hash table which had been constructed during data selection. Besides that, it is assumed that more learning time would be reduced especially for large dimensions dataset, since the number of projection vectors are much smaller than the number of feature vectors. To update the weights in column *Update weights using Eq. (1.4)*, only the selected RBF bases are used.

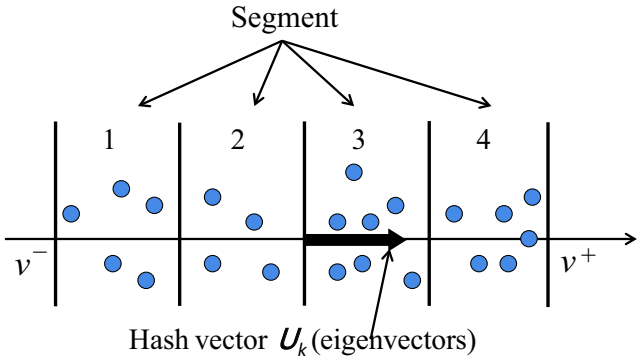


Figure 2.2 Calculation of hash values using an eigenvector.

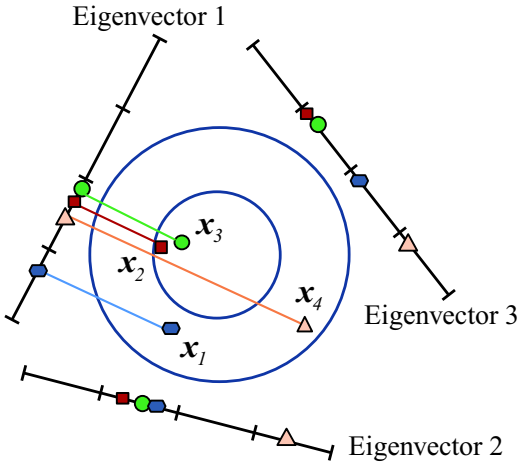


Figure 2.3 Data projection into eigenvectors.

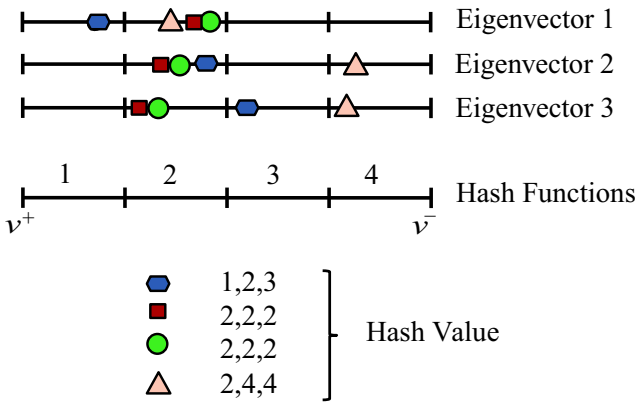


Figure 2.4 Index representation for each inputs.

Table 2.3 Evaluated UCI Datasets.

Dataset	#Init. Data	#Train. Data	#Test Data	#Attrib.	#Classes
Hiragana	500	8366	8365	13	38
Adult	500	22611	22611	14	2
Shuttle	500	28989	28988	9	5
Bank	500	22606	22605	16	2
Pendigits	500	5496	5496	16	10
Letter	500	10000	10000	16	26
MAGIC	500	9510	9510	10	2
Optical Digits	281	2810	2810	64	10
Semeion	80	797	796	256	10

2.3 Performance Evaluation

2.3.1 Experimental Setup

To evaluate the performance of the proposed method, the nine benchmark datasets in the UCI Machine Learning Repository[44] are used. The dataset information is shown in Table 2.3. Although training and test data are separately provided in some data sets, they are merged and randomly divided into two subsets to evaluate the average performance through the two-fold cross-validation. Since the performance generally depends on the sequence of training data, 30 sequences in total are trained incrementally and the average performance is evaluated for the test data. For the first seven datasets in Table 2.3, 500 initial training data are randomly selected from the training data. For the other datasets (Optical Digits and Semeion), 10% of the training data are randomly selected as initial data. After the initial learning, it is assumed that a large chunk of data is given sequentially in the incremental learning phase. The number of data in a chunk is set to 1,000 at every learning stage except for the last stage. Since training data are assumed to be given only once, the data chunk given at the last stage may consist of less than 1,000 data.

In the following experiments, the proposed RAN-LSH are evaluated from the following three points: (1) appropriateness of the granularity in the hash encoding, (2) effectiveness of the data selection, and (3) the effectiveness of the local learning in modified RAN. For the notational convenience, let us denote modified

RAN with the margin-based data selection as RAN-MRG.

In RAN-LSH, the three parameters on accumulation ratio θ_a, output margin θ_m, and tolerant distance θ_p should be determined empirically. In this experiment, the following parameter settings are used: $\theta_a = 0.7$, $\theta_m = 0.05$, and $\theta_p = 3$. Meanwhile, the error threshold ε is set to 0.5. The other parameters, RBF width σ and the number of partitions P, are determined in the initial learning phase through the cross-validation.

2.3.2 Granularity in Hash Encoding

To avoid unnecessary calculations of output margins in the data selection, similar data are associated with the same partition to represent the learning status (i.e., well-trained or untrained). As easily expected, if the number of partitions is small, a state space is roughly represented in the hash encoding and the *collision* probability of dissimilar data tends to be high. On the contrary, if the number of partitions is large, the collision probability would be low (i.e., data are sparsely allocated to each segment/bucket of P partition), while the size of a hash table would be large, resulting in the increase of the memory and computational costs.

To find a suitable granularity in the hash encoding (i.e., a proper number of partitions P), let us introduce the following Davies-Bouldin index I_{DB}[45]:

$$I_{DB} = \frac{1}{P}\sum_{i=1}^{P}\max_{j:i\neq j}\frac{S_i + S_j}{\|\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_j\|} \tag{2.7}$$

where $\bar{\boldsymbol{x}}_i$ and S_i are the centroid and the variance of data allocated to the ith-segment of P partition, respectively. The numerator in Eq. (2.7) corresponds to the within-cluster scatter for the segments i and j, which should be as small as possible. The denominator corresponds to the between-cluster separation between the centroids $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{x}}_j$. This Davies-Bouldin index I_{DB} depends on the data as well as the hash encoding method, and ideally has to be as small as possible. However, as mentioned above, too small I_{DB} leads to the sparse data allocation

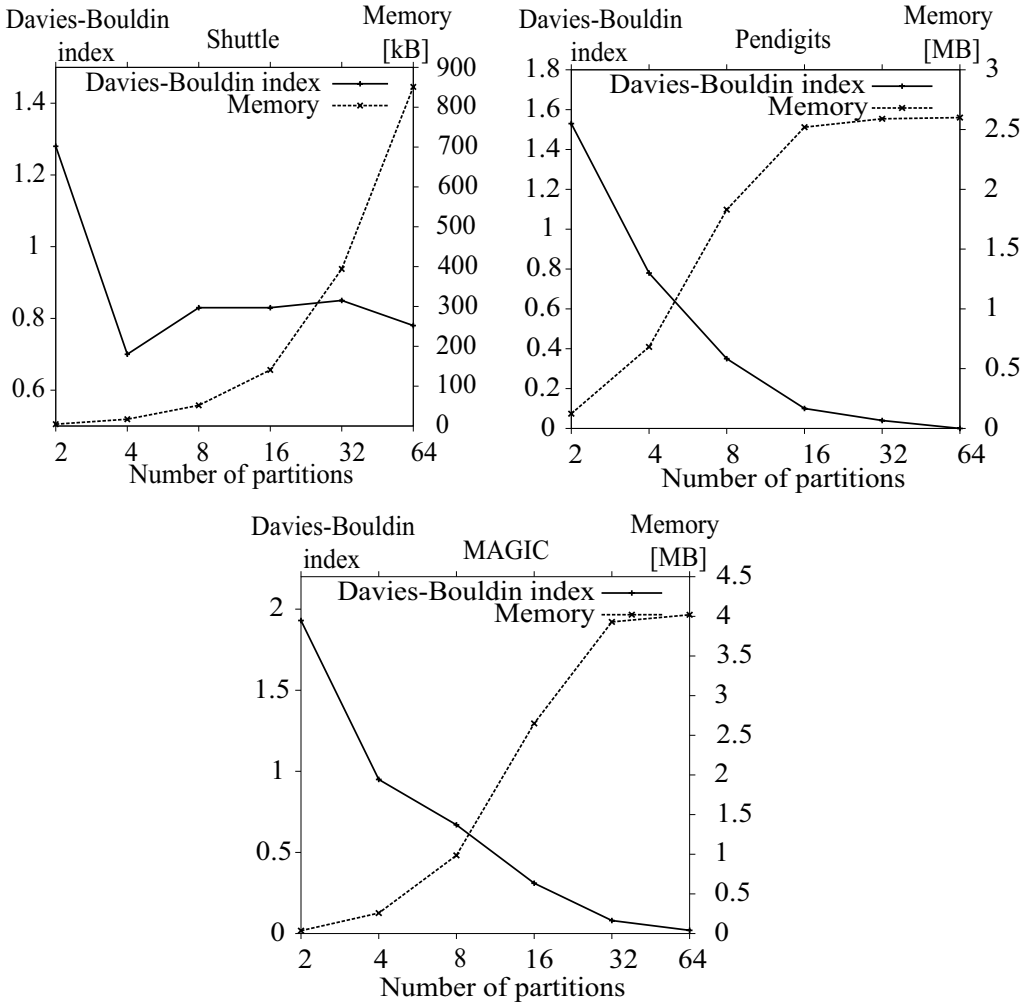


Figure 2.5 Effect of the number of partitions P to the Davies-Bouldin index I_{DB} and the memory consumption for (a) Shuttle, (b) Pendigits, and (c) MAGIC datasets.

in the hash encoding and results in the increase in computation and memory costs. Therefore, a proper value of P is needed to be determined such that a moderate value of I_{DB} with less memory and computational costs can be obtained.

Figure 2.5 illustrates I_{DB} and the memory size for different numbers of partitions P. Shuttle, Pendigits and MAGIC Gamma Telescope (MAGIC) datasets are used for the evaluation. As seen in Fig. 2.5, when P increases, I_{DB} decreases; that is, the collision probability of dissimilar data is getting low. In contrast,

Table 2.4 Effect of the number of partitions P to learning time [sec.]

Methods	Shuttle	Pendigits	MAGIC
RAN-MRG	1.6±0.4	1.2±0.1	13.8±1.7
RAN-LSH ($P = 2$)	0.6±0	0.3±0.1	0.3±0.2
RAN-LSH ($P = 4$)	1.0±0.1	0.9±0.1	4.4±0.8
RAN-LSH ($P = 8$)	1.0±0.1	1.4±0.1	10.0±1.6
RAN-LSH ($P = 16$)	1.0±0.1	1.6±0.1	14.4±1.9
RAN-LSH ($P = 32$)	1.3±0.2	1.6±0.1	14.7±1.8
RAN-LSH ($P = 64$)	3.3±0.5	2.2±0.1	16.5±1.7

the required memory size is increased due to the increase in the number of hash entries. Meanwhile, Table 2.4 shows the learning time for different numbers of partitions. Obviously, when the number of partitions increases, the time for incremental learning becomes long.

From the results in Fig. 2.5 and Table 2.4, one can say that an optimal value of P exists between 4 and 16. Therefore, let us find a suitable number of partitions P within this range by applying the cross-validation to an initial dataset.

2.3.3 Effectiveness of Data Selection

To see the effectiveness of the data selection in RAN-LSH, the effectiveness of the proposed model to find similar data is investigated using CIFAR-10 dataset. CIFAR-10 dataset consists of 32x32 colour images of ten different classes which are horse, airplane, car, bird, cat, dog, frog, ship, truck and deer. It is sensible to use this dataset because the strength of the proposed method to gather similar data in the same classes as one group and avoid grouping different classes into a same group is able to be shown, although the data is almost similar such as dog and cat classes. Figure 2.6 illustrates the first 13 images which are sorted in descending order, from the most similar to the most not similar data. We can see that LSH-based data selection which embedded in RAN-LSH classifier, not only capable to group similar data in a same group, but also showed that most of these similar data is grouped according to their correct classes. This result has proven that the PCA projection-based LSH is sufficient to find similar data effectively.

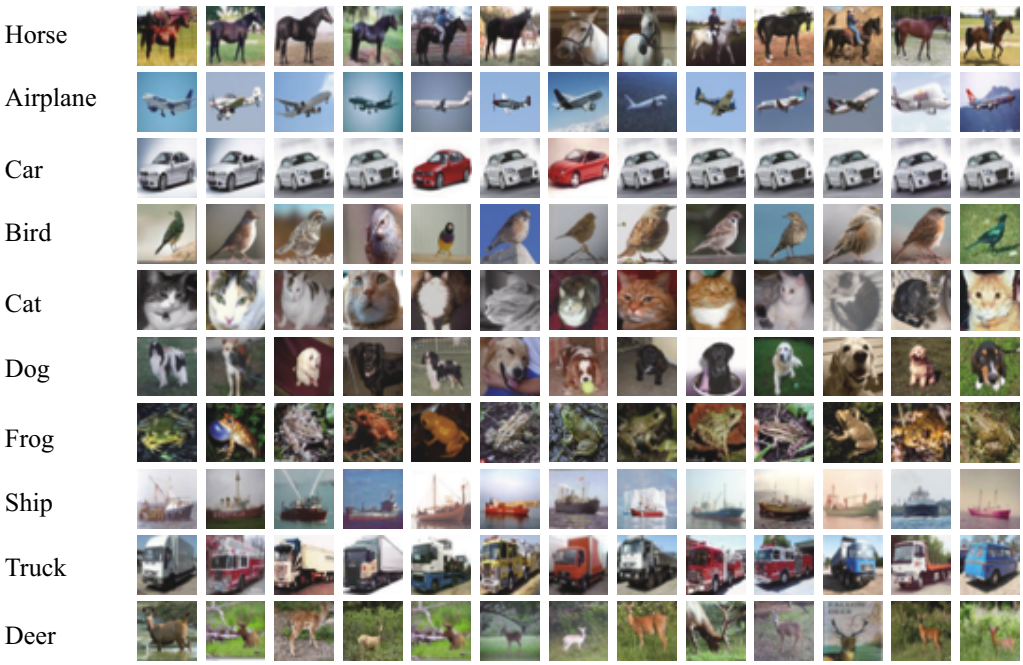


Figure 2.6 The effectiveness of selecting similar data using CIFAR-10 dataset.

Besides that, the performance of the model is compared with the other two modified RAN models, in which no data selection mechanism is introduced (RAN model) and a margin-based data selection is introduced (RAN-MRG model). Table 2.5 shows the results of the performance comparison based on the following measures: (a) the final recognition accuracy [%], (b) learning time [sec.], and (c) the total number of selected data. The final recognition accuracy is examined for the test dataset after the incremental learning phase is completed, and the learning time only includes the computational time for incremental learning. These results are averaged over 30 trials with different data sequences. Each results are represented in the form of (average) $\pm$ (standard deviation). The single asterisk (*) and the double asterisk (**) mean that the average difference against modified RAN (no data selection) is statistically significant under p-value is less than 5% and 1% level, respectively.

As seen in Table 2.5(a), the recognition accuracies of RAN-MRG and RAN-

Table 2.5 Effectiveness of introducing the data selection. The performance is compared among three models; modified RAN, RAN-MRG and RAN-LSH, regarding (a) the final recognition accuracy [%], (b) learning time [sec.], and (c) the total number of selected data. Note that modified RAN has no data selection mechanism.

(a) Recognition accuracy [%]

Dataset	RAN	RAN-MRG	RAN-LSH
Shuttle	98.4±1.8	98.9 ± 1.1	98.6 ± 0.9
Pendigits	98.9±0.2	$98.3 \pm 0.3^{**}$	$98.3 \pm 0.3^{**}$
MAGIC	76.5±0.8	$74.6 \pm 2.1^{**}$	$75.1 \pm 2.1^{*}$

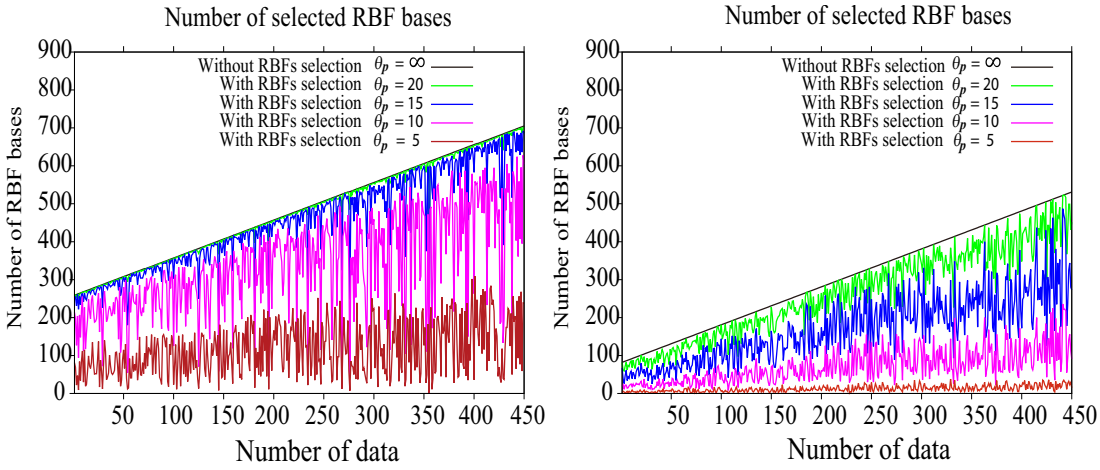
(b) Learning time [sec.]

Dataset	RAN	RAN-MRG	RAN-LSH
Shuttle	151.2±20.7	$1.6 \pm 0.4^{**}$	$1.3 \pm 0.5^{**}$
Pendigits	134.7±9.8	$1.9 \pm 0.1^{**}$	$1.7 \pm 0.1^{**}$
MAGIC	10,808±809	$18.8 \pm 1.7^{**}$	$16.2 \pm 2.2^{**}$

(c) Total number of selected data

Dataset	RAN	RAN-MRG	RAN-LSH
Shuttle	28,488	$216 \pm 101^{**}$	$95 \pm 28^{**}$
Pendigits	4,996	$67 \pm 8^{**}$	$57 \pm 10^{**}$
MAGIC	9,010	$380 \pm 27^{**}$	$249 \pm 30^{**}$

LSH are slightly dropped compared with that of modified RAN because many training data are discarded in the data selection (see Table 2.5(c)). For Shuttle, only 0.3% of training data are used for learning in RAN-LSH on average. Thus, 99.7% training data are discarded by the proposed LSH-based data selection, although the performance seems to be relatively decreased by 0.1%. On the other hand, although small performance drops are recognized for Pendigits and MAGIC, the percentages of the data reduction are 98.9% and 97.2%, respectively. Obviously, this data reduction results in significant speed-up in learning (see Table 2.5(b)). Compared with RAN-MRG, the proposed RAN-LSH can reduce more training data without deteriorating the recognition accuracy. Therefore, it can be concluded that the proposed LSH-based data selection works properly.



(a) Optical Digits (b) Semeion

Figure 2.7 Effect of RBF bases selection with different θ_p. Best viewed in color.

2.3.4 The Effect of RBF Bases Selection and Comparison to State-of-the-Art Method

To investigate the efficiency of RAN-LSH with RBF selection, two observations are carried out as follows: (1) identify the effect of selecting RBF bases using different values of θ_p, and (2) examine the performance regarding (a) the recognition accuracy and (b) learning time. Figure 2.7 and Table 2.6 show the above findings respectively. Besides that, in this subsection, a state-of-the-art approach incremental SVM with data selection is being compared with the proposed method as the baseline of the findings. To the best of my knowledge, there are some SVM algorithms which support incremental learning[36, 37, 38]. However, in[36, 37], the incremental SVM does not support multiclass classification. Then, the method that can handle multiclass classification with data selection is chosen. The results to compare the proposed method with incremental SVM in[38] is included in Table 4.

Figure 2.7 illustrates the effects of using different values of θ_p towards number of RBFs selected for two datasets (a) Optical Digits and (b) Semeion. In Fig. 2.7 (without RBFs selection), it can be seen that the number of RBF bases increases gradually in proportion to the number of training data. As previously stated

Table 2.6 The effect of selecting RBF centers using different threshold values θ_p namely through; (a) recognition accuracy [%], (b) incremental learning time [s], and (c) the total number of RBF bases selected. The values represented are the mean and standard deviation of each method.

(a) Recognition accuracy [%]

Dataset	RAN-MRG	RAN-LSH $\theta_p = \infty$	RAN-LSH $\theta_p = 10$	RAN-LSH $\theta_p = 3$	Adaptive SVM
Hiragana	96.8±0.4	96.6±0.4	96.8±0.4	96.7±0.4	97.2±0.7
Adult	78.1±1.2	78.0±1.2	78.1±1.0	77.8±1.3	75.2±0.2
Shuttle	98.9±1.1	98.6±0.9	98.3±1.4	98.1±1.3	94.9±0.1
Bank	84.1±1.8	84.4±1.7	83.9±1.9	84.6±1.5	88.3±0.2
Pendigits	98.3±0.3	98.3±0.3	98.3±0.3	97.8±0.4	97.6±1.6
Letter	86.6±0.5	86.4±0.5	86.5±0.5	86.4±0.8	86.5±1.0
MAGIC	74.6±2.1	75.1±2.1	75.7±1.8	73.9±2.0	68.9±5.4
Optical Digits	97.7±0.5	97.7±0.5	97.7±0.5	97.7±0.5	98.2±0.9
Semeion	90.4±0.1	90.4±0.1	90.4±0.1	90.4±0.1	85.1±15.1

(b) Incremental learning time [sec.]

Dataset	RAN-MRG	RAN-LSH $\theta_p = \infty$	RAN-LSH $\theta_p = 10$	RAN-LSH $\theta_p = 3$	Adaptive SVM
Hiragana	48.0±4.5	42.9±4.0	10.5±1.6	7.4±1.1	288.4±9.8
Adult	102.4±10.8	90.1±9.0	30.0±6.2	14.9±2.6	23.2±2.1
Shuttle	1.6±0.4	1.3±0.5	1.2±0.5	1.2±0.4	281.2±0.1
Bank	17.3±1.7	15.7±1.8	15.2±1.9	6.8±1.0	38.3±40.6
Pendigits	1.9±0.1	1.7±0.1	1.0±0.0	0.9±0.0	36.8±3.5
Letter	218.1±13.8	179.0±16.4	156.0±8.9	8.5±0.3	338.7±69.7
MAGIC	18.8±1.7	16.2±2.2	4.5±0.8	2.6±0.3	28.9±24.2
Optical Digits	65.8±3.2	63.2±2.9	18.2±2.4	9.3±4.3	20.9±0.5
Semeion	124.5±1.2	121.9±3.1	8.9±1.2	4.1±0.6	10.6±0.6

(c) Total number of RBF bases

Dataset	RAN-MRG	RAN-LSH $\theta_p = \infty$	RAN-LSH $\theta_p = 10$	RAN-LSH $\theta_p = 3$	Adaptive SVM
Hiragana	559.9±17.5	557.7±16.8	575.0±19.0	572.2±19.1	-
Adult	670.2±21.0	688.0±23.8	757.3±54.0	608.4±19.6	-
Shuttle	97.0±5.1	56.5±10.6	60.1±14.7	53.9±10.4	-
Bank	274.1±15.2	277.2±16.4	273.6±17.3	268.1±40.9	-
Pendigits	176.6±6.7	176.7±6.7	176.2±6.7	159.4±6.5	-
Letter	818.6±16.4	817.2±13.9	814.9±15.2	832.2±32.3	-
MAGIC	352.6±13.4	374.1±13.8	401.7±28.7	333.7±11.9	-
Optical Digits	727.4±10.4	727.9±10.1	727.7±10.2	728.3±10.3	-
Semeion	792.1±2.1	792.1±2.1	792.1±2.1	792.1±2.1	-

in Algorithm 1, the RBF center would be added if the current model is not well-trained and thus RAN-LSH model without RBFs selection is not suitable

for large-scale data sequences. This is because the learning time would take an eternity to update the weights of a very large RBF outputs matrix $\mathbf{\Phi}$. As a result, the model could not catch up with the speed of the next data chunk. However, the influence of expansion on hidden units' size can reduced by learning only RBF bases which are located near to the training data.

Learning a training sample along with near RBF bases could prevent catastrophic forgetting in the incremental learning. However, learning all RBF bases would cause a new interference due to the distribution of the RBF bases that is sparsely distributed. One of the RBF bases selection strategy is to choose the active RBFs using Euclidean distance. Conversely, in the proposed method's approach, the LSH distance is used to choose RBF bases that are near to the given training data. In this present study, θ_p is used to select the maximum LSH distance between a training data and all RBF centers in the hidden layer nodes. When a small value of θ_p is used, only a small area of the near RBF bases to a training data is involved and thus a few RBF centers would be selected and used to update the weights. Meanwhile, when θ_p approached ∞, RAN-LSH model would act as such there is no RBF bases selection since the area of the nearest RBF centers would cover almost all regions in the input space. The advantage of this approach is shown through this experiment in that; it selects only one nearest RBF bases during learning by using $\theta_p = 3$, thus enabling this model to learn a large-scale data sequence in high-speed.

By selecting near RBF, it is assumed that the active RBF bases would be selected to be learned together with a training data. To evaluate the performance of the proposed method, the performance of the following four models are compared: conventional RAN-MRG (without RBFs selection), RAN-LSH without RBFs selection ($\theta_p = \infty$), RAN-LSH with RBFs selection using $\theta_p = 10$ and RAN-LSH that selects the nearest RBFs ($\theta_p = 3$). Nine UCI datasets are used to evaluate the performance of these models.

Table 2.6 demonstrates (a) the final recognition accuracy, (b) the incremental

learning time and (c) the final number of RBF bases created. In Table 2.6 (a) and (c), for RAN-LSH with $\theta = \infty$ (without RBF selection) model, it can be seen that the LSH-based data selection is comparable to the conventional margin-based data selection. This can be justified from the findings of recognition accuracy and final number of RBF bases created that are almost similar. Nevertheless, RAN-LSH with $\theta = \infty$ is capable to speed up the learning where all datasets show a time reduction during learning.

To shorten more learning time in RAN-LSH, the locally learning model (RAN-LSH with RBF selection) is adopted. Local RAN-LSH model can avoid the high computional of SVD calculation because only a few RBFs in the hidden layer are selected to update the weight. For RAN-LSH with $\theta_p = 10$ model, the recognition accuracy and the final number of RBF bases are able to be maintained as previously models, RAN-MRG and RAN-LSH without RBF selection. Furthermore, the incremental learning time is successfully reduced to 35% of the incremental learning in RAN-MRG on average for five datasets which are Hiragana, Adult, MAGIC, Optical Digits and Semeion datasets using RAN-LSH ($\theta_p = 10$) model.

On the other hand, RAN-LSH with $\theta_p = 3$ is able to achieve at most 0.9% difference of recognition rate for Shuttle dataset compared to RAN-MRG. Whereas for the other datasets, the recognition accuracy shows a difference of only 0.5% or lower. This shows that it is sufficient to use only one nearest RBF center to be learned together with a given training data. In term of incremental learning time (see Table 2.6 (b)), six datasets, which are Hiragana, Adult, Letter, MAGIC, Optical Digits and Semeion datasets would require only 20% of the incremental learning in RAN-MRG on average, where 84.6%, 85.4%, 96.1%, 81.2%, 85.9% and 96.7% of learning times are speeden up, respectively. Whereas, for Shuttle, Bank and Pendigits datasets, at least 25% of the incremental learning time is shortened compare to RAN-MRG. Another advantage of RAN-LSH with $\theta_p = 3$ model is the final number of RBF bases shown to be lower than RAN-MRG for all datasets except Hiragana, Letter and Optical Digits where an extra 20 RBF bases

are created at most. This result indicates that for especially large scale dataset, RAN-LSH using $\theta_p = 3$ model can effectively be used to reduce the learning time without sacrificing the recognition accuracy.

As shown in Table 4(a), the proposed method RAN-LSH $\theta_p = 3$ achieved a comparable accuracy rate as Adaptive SVM except for Bank dataset. On the other hand, the learning time of Adaptive SVM is longer compared to RAN-LSH with $\theta_p = 3$ for all datasets. From Table 4(b), the learning time of RAN-LSH with $\theta_p = 3$ and that of Adaptive SVM for two classes datasets (i.e., Adult, Bank and MAGIC) do not have significant difference. RAN-LSH can learn one to eleven times faster than Adaptive SVM. On the other hand, for the datasets with more than two classes, the proposed method can learn quite faster than Adaptive SVM. This finding here appear to support the weakness of SVM classifier that is originally designed to handle two classes problem or binary classification[46]. To facilitate the multiclass classification using SVM, one of suggested approach is "one-against-all" approach. However, using this approach it is obvious that the learning time of SVM is increased due to these looping. Thus, it can be concluded that RAN-LSH can learn faster than SVM with comparable classification performance, especially for large number of training data and multiclass problems.

2.4 Conclusions

In this chapter, a fast incremental learning algorithm of modified RAN namely RAN-LSH is proposed which can be used to select essential data and near RBF bases. By utilizing this, the high cost for calculating high dimensional data can be lowered, as the repetition of output margins and weights update calculations are able to be avoided. In the present study, the LSH-based data selection and LSH-based RBF selection are introduced in the modified RAN classifier to solve two major issues in large-scale data sequences. Through the experiments, it is shown that the proposed method of RAN-LSH with $\theta_p = 3$ is capable to learn a

large chunk of data within a short period of time. This can be seen when only 20% of RAN-MRG's learning time on average is used by the proposed method, while keeping a similar accuracy rate to the conventional method.

Furthermore, the proposed learning scheme provides desirable characteristics for large-scale data sequences application. First, the learning is conducted in one-pass; which is only new training data would be used to be learned by the classifier, whereas the previous training data were discarded after the learning. This property is very important for learning large-scale data sequences where the data are growing in size, thus it is nearly impossible to learn and store a massive data without using one-pass learning. The other property is that the learning time is fast although large data chunk are given in every learning stage. From several experiments using nine datasets in the UCI Machine Learning Repository, the proposed scheme is verified to has the above desired characteristics.

There still remains several open problems in the proposed RAN-LSH. It can be seen that the fastest learning time is achieved by using LSH with two partitions. However, the recognition accuracy is slightly reduced if a small number of partitions is used. The reason is because the matrix of eigenvectors U_l are only determined during the initial training and does not updated incrementally. An alternative solution is to update eigenvectors U_l incrementally and adapt with the most sufficient number of partition to assign the hash value. This can be attained by introducing Incremental PCA (IPCA)[47] into the proposed RAN-LSH with two partitions. However, there is one problem for this. After updating hash functions, the hash table should also be updated properly without unexpected forgetting of the previous knowledge. If this problem is solved, it is expected that the idea of fast processing for large-scale data sequences using LSH would work faster.

The above issues are left as the future work.

Chapter 3

An Online Malicious Spam Email Detection System Using Resource Allocating Network with Locality Sensitive Hashing

3.1 Introduction

Emails have become one of the most frequently used methods for cyber attacks. The most worrying email-based attack is TME[48, 49]. In TME, attackers send malicious emails to certain people targeted in an organization, such as executives of large companies, high-ranking government personnel, military officials and even famous researchers, in order for the attackers to obtain valuable confidential information and latest research of the targeted people. In TME, an email often has an attachment with malicious codes that can be installed automatically upon opening without the victims realizing it. In some cases, the victims' computer will become the back door for the attackers who in turn have the authority to enter the network of the targeted persons and thus steal confidential information.

Another typical email-based cyber attack is the malicious spam email attack, which aims to spread numerous emails with Uniform Resource Locator (URL) links leading to malicious websites. Previously, malicious codes were sent through the attachment of such spam emails. However, many successful filters have been developed to detect malicious attachments. Thus, attackers are now turning to malicious spam campaigns that attack using the links attached in the emails. According to the Symantec annual report in 2014[50], about 87 percent of scanned spam messages contained at least one URL hyperlink. Moreover, recent findings

by Symantec[51] show a sharp rise of emails containing malicious links, from 7% in October 2014 to 41% in the following months. Apart from that, currently, attackers also use more relevant email content[48] that are specific to their victims' line of work, besides addressing the name of the recipient in the email body to convince the victim that the email received is a normal email. For instance, a fake email notification regarding a conference or journal targeted towards a recipient with academic status, notifications regarding false documents such as telecommunication service bills, fax and voicemail in which the victims are given a link to get more information[51]. This technique is called Social Engineering[52], which Hadnagy[53] defines as "The Art of Human Hacking". It becomes difficult for normal users to distinguish not only between non-malicious and malicious spam emails but also spam email from normal emails.

The objective of this chapter is to detect the malicious spam emails so that general users can be protected from being re-directed to malicious websites. For this purpose, an autonomous online system is proposed for detecting malicious spam emails. In general, it is not easy to collect spam emails from individual persons because it is not usually permitted to access personal email spools. Therefore in the proposed system, we collect double-bounce spam emails that are delivered to unknown users. From the collected spam emails data, a classifier model is used to learn and classify the malicious spam emails. The updated connection weights of the classifier model are sent to a user's mailer software to improve the malicious spam email detection ability. Jungsuk[54] points out that the live period of malicious URLs is often very short, usually within a few days; thus, it is expected that introducing incremental learning to malicious spam email detection will be effective. The system can learn from the recent spam emails so that the spam email detection system is always up to date. On the other hand, spammers often use botnets to spread spam emails. For example, a botnet called Rustock which consists of approximately 1 million infected computers that networked together, is capable of sending up to 30 billion spam emails every day[55]. Since the distri-

bution of such spam emails is done in a short time, it is assumed that the spam emails have the same or similar contents in general[56]. Hence, the LSH[4, 5, 6, 7, 8] is adopted to quickly select important training data to be learned. For this purpose, RAN-LSH is adopted as a classifier model in the proposed detection system. This model has the following two important properties: (1) the learning is carried out incrementally, and (2) only data within an untrained region are selected and learned even when a large amount of data is given.

This chapter is organized as follows. The proposed system for detecting malicious spam emails is presented in Section 3.2. In Section 3.3, the performance of the spam email detection system is evaluated for a set of 20,448 double-bounce emails collected from 1st March, 2013 to 10th May, 2013. Finally, conclusions and future work are addressed in Section 3.4.

3.2 The Proposed Malicious Spam Email Detection System

3.2.1 System Architecture

Figure 3.1 illustrates the architecture of the proposed autonomous online malicious spam email detection system[57] which is composed of three components: (1) autonomous spam email collection system, (2) text processing and feature transformation, and (3) RAN-LSH classifier embedded with the data selection and outlier detection mechanisms.

As mentioned in Section 2.2, learning all the given data is not a good strategy under incremental learning environments because the learning may not be completed before a new data set is given[58]. To enhance the adaptibility to dynamic environments, the learning should be carried out with essential data that are selected in an online fashion. There are two types of essential data for a learning purpose. The first type is the data located close to a class boundary[59], while the other is the data located outside of the learned region (i.e., outlier). In order to ensure fast and accurate learning, the data selection mechanism should be in-

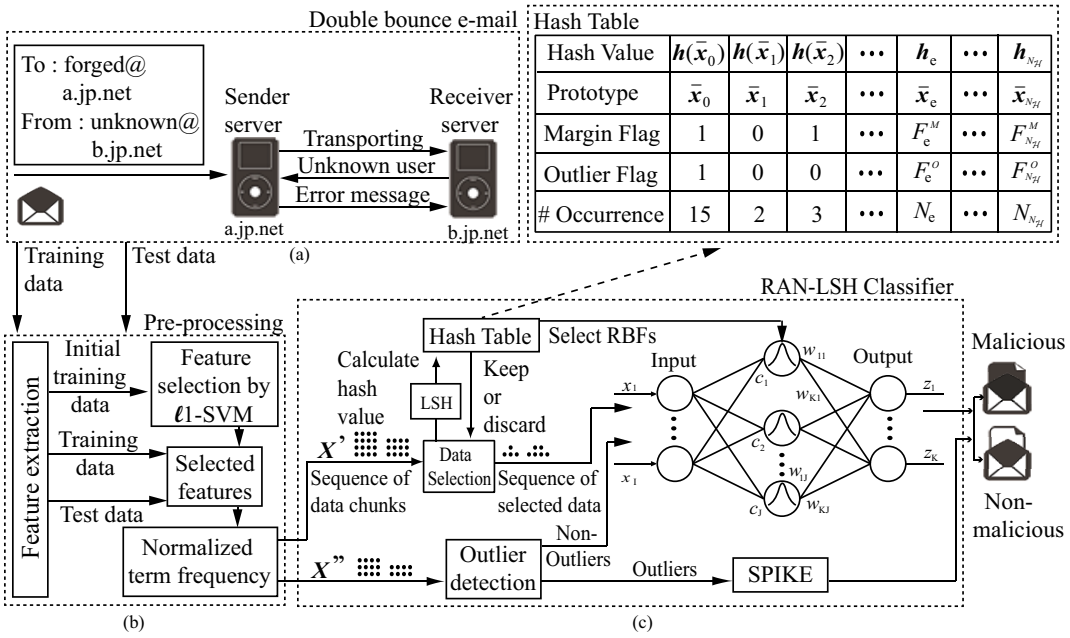


Figure 3.1 Network structure of the proposed autonomous malicious spam email detection system.

troduced into a classifier model to find such essential data from a given chunk of data.

The first type of essential data has been discussed in Section 2.2. On the other hand, the second type of essential data are selected by the outlier detection. This type of essential data selection is introduced into the previous RAN-LSH classifier. The outlier detection relies on the output margin and the number of occurrence of similar data in the input space which are represented by outlier flag F_e^O.

In the following subsection, the details of five components' autonomous online malicious spam email detection system are explained, which consists of autonomous spam email collection system, autonomous labeling system, pre-processing, outlier detection and RAN-LSH classifier. Algorithm 7 and Algorithm 8, summarized the steps to carry out the autonomous spam email detection system and the pre-processing procedure of the system, respectively.

3.2.2 Autonomous Spam Email Collection System

Figure 3.1(a) illustrates the process of obtaining double-bounce emails. Let us consider a case that a spammer sends a large number of emails. In many cases, almost all emails will reach existing users. However, it is very likely that some email addresses are no longer in use for some reason. Therefore, the email server would return such emails with unknown addresses to the sender. If the spammer intends to send a malicious email, it is also likely that the spammer has faked the originating address and such emails would be re-sent to the receiver. This type of unreachable error email is called "double-bounce email"[54] and they are usually disposed of by the email server on the receiver's side. This mechanism of generating double-bounce emails to collect malicious spam emails automatically is utilized.

3.2.3 Autonomous Labeling System

To use double-bounce emails as training data under the supervised learning, the class label of the data are needed. Needless to say, spammers try to conceal their malicious intention; therefore, it is not easy to determine the maliciousness from the collected double-bounce emails. The only way to identify the maliciousness is by clicking the URLs. Evidently, this is very dangerous for general users; therefore, a crawling-type web maliciousness analyzer called SPIKE is used, which was developed by NICT in Japan.

Figure 3.2 illustrates how the maliciousness of URLs in a spam email is analyzed in SPIKE[60]. The URL links in the email are first extracted from a double-bounce email and SPIKE downloads the html file and attached materials (e.g., java scripts, pdf, doc files) in the entrance page. It then continues to find other URLs in the downloaded pages again. This process is conducted recursively by crawling the linked websites, and all the downloaded materials are analyzed. Emails that are only link to a normal webpage with non-malicious contents are considered as non-malicious spam emails (i.e., all contents of Webpage 1-6 in Fig.

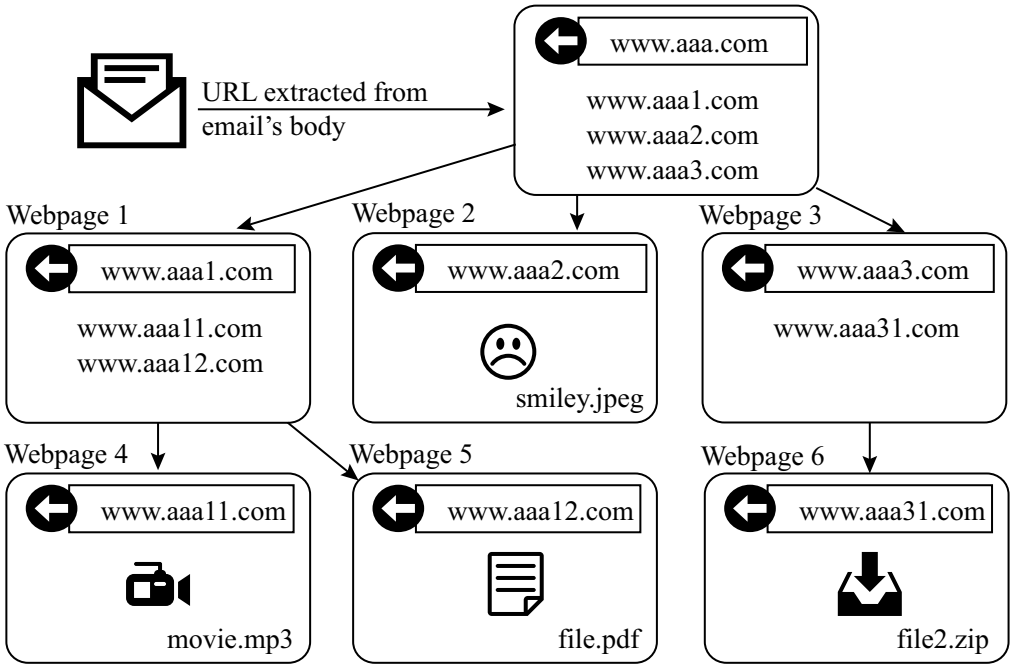


Figure 3.2 Example of web crawler and content analysis using SPIKE.

3.2 are normal), whereas the emails with at least one suspicious content (i.e., one of Webpage 1-6 in Fig. 3.2 is malicious) are identified as malicious spam emails.

3.2.4 Text Processing and Feature Transformation

In order for the classifier to carry out the classification task effectively, the classifier requires instances as the input instead of the raw spam emails for the learning purpose. The instances consist of informative features with a fixed-length which are extracted from the emails. Thus, appropriate pre-processing steps are required so that the arbitrary data of text messages are transformed into features with numerical features. Figure 3.1(b) demonstrates the pre-processing module of the spam email detection system. Feature extraction of spam emails involves tokenizing and lemmatizing the documents into bag-of-words (BoW). Tokenization breaks the sentences in the emails into pieces of words and removes frequent words called *stop words* such as 'the', 'which', 'are', etc. Besides filtering out *stop words*, lemmatization also reduces the number of words in BoW by

transforming redundant words that end with '*ing*', '*ed*' and '*s*' into their root word (e.g., 'learned' to 'learn').

The BoW features usually consist thousands or millions of feature vectors. In general, only some features are informative and are able to differentiate different classes. Therefore, feature selection is carried out to select the most informative features in order to reduce the number of dimensions and avoid the computational complexity. Firstly, the initial training data are transformed into feature vectors with term frequency-inverse document frequency (TF-IDF) feature representation. Next, linear ℓ_1-Support Vector Machines (ℓ_1-SVM) is used as a feature selection strategy that requires two steps; training using linear SVM[61] and eliminating features with low weights[62]. SVM is able to find a decision boundary by optimizing the following objective function:

$$\begin{aligned} & \min_{\boldsymbol{w},b} \frac{1}{2} \parallel \boldsymbol{w} \parallel^2 + C \sum_{i=1}^{m} \xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b) \leq 1 - \xi_i, \\ & \xi_i \leq 0, \quad i = 1, \ldots, m \end{aligned} \tag{3.1}$$

which maximizes the margin $\parallel \boldsymbol{w} \parallel^2$ of hyperplane $\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b$ between two classes $y_i \in \pm 1$ and contains only a minimum training error ξ_i (i.e., training data located above the support vectors which belong to the class of the training data). The parameter C controls the trade off between margin maximization and errors of the SVM on training data, where a larger C corresponds to a higher penalty to errors. The weights $\boldsymbol{w}$ obtained is used to select N_f number of features by choosing the highest N_f-rank weights.

To represent the selected features of initial training data and the remaining training data, the normalized TF-IDF[63] is used to measure the importance of a word to a document (i.e., document refers to the spam email) in the collection of documents given by the following equation:

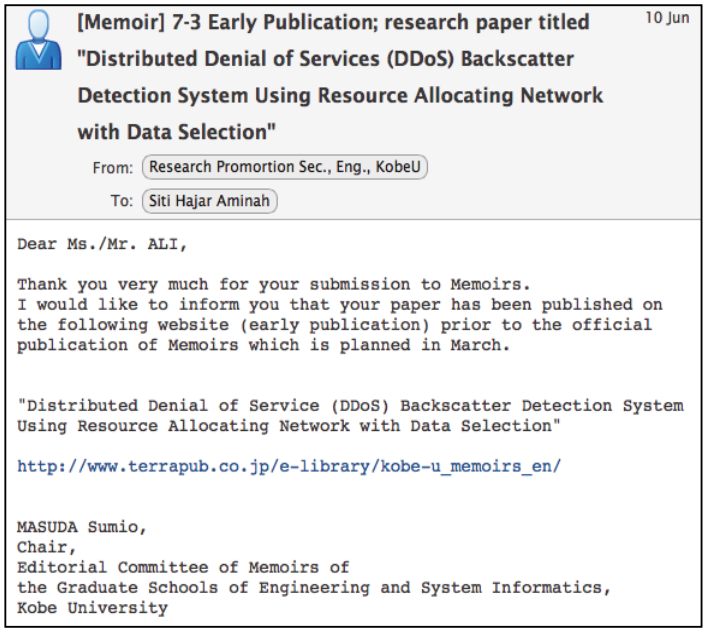
$$TF\text{-}IDF(t, d, D) = \frac{f_d(t)}{\sum_t f_d(t)} \cdot \log \left(\frac{|D|}{1 + |\{d \in D : t \in d\}|} \right) \tag{3.2}$$

where $f_d(t)$, $\sum_t f_d(t)$, $|D|$ and $|\{d \in D : t \in d\}|$ are the frequency of term t in a document d, the total frequencies of all terms in document d, the total number of document in corpus and number of documents which have term t, respectively. The normalized term frequency (TF) is used to provide a balanced value to all documents that have a different number of words. If term t appears frequently in a document d_i and seldom occurs in other documents in D, the value of TF-IDF would be high where both TF and inverse document frequency (IDF) obtain high values. This indicates that term t is important to document d_i. Otherwise, if either the occurrence of term t is low in document d_i or term t is always appears in other documents in D, this would indicate that term t is not important to document d_i where the value of TF-IDF is low or '0'.

Figure 3.3 (a) and (b) demonstrates an example of normal email and spam email, respectively. Both email are attached with one or more URL links. As mentioned before, some of the URL links in the spam email will direct the user to harmful websites, thus, this type of spam email is categorized as malicious spam email. Table 3.1 shows the overall pre-processing steps which consist of tokenizing, lemmatizing, selecting important features and data representation. After going through the entire procedure above, these data are used as the input to the classifier model, where the example of RAN-LSH classifier calculation is shown in Table 2.2.

3.2.5 Outlier Detection

Although SPIKE can judge the maliciousness of spam emails, the analysis takes time, from a few minutes to even longer than ten minutes. Therefore, it is difficult to check all the collected double-bounce emails by SPIKE in real time. The outlier detection mechanism is introduced into RAN-LSH in order to reduce the number of spam emails to be checked by SPIKE. That is, only a new type of unknown spam emails (i.e., outlier) should be selected and sent to SPIKE for labeling. For this purpose, malicious spam email detection system is proposed by



[Memoir] 7-3 Early Publication; research paper titled "Distributed Denial of Services (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection"

10 Jun

From: Research Promortion Sec., Eng., KobeU

To: Siti Hajar Aminah

Dear Ms./Mr. ALI,

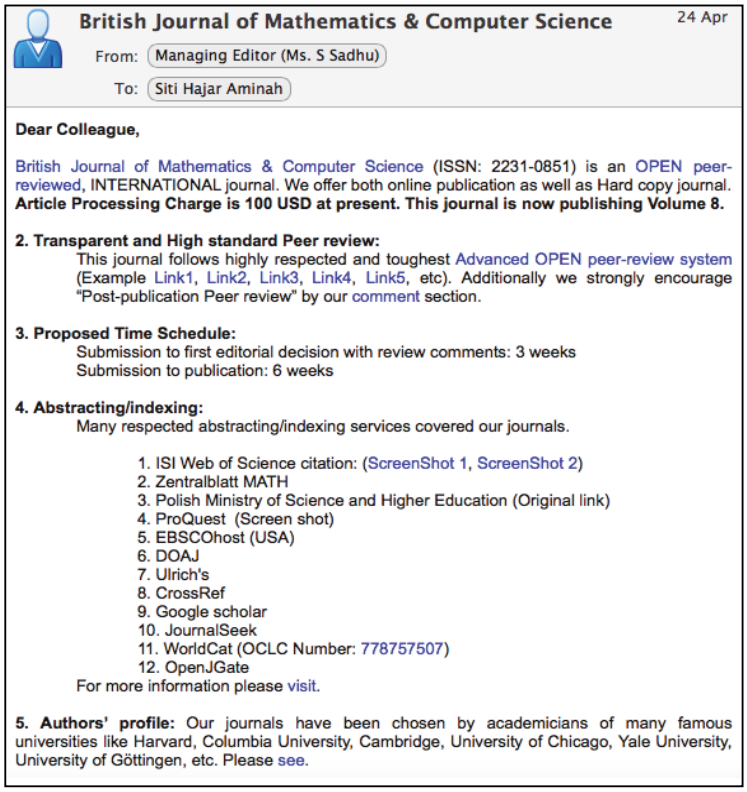
Thank you very much for your submission to Memoirs.
I would like to inform you that your paper has been published on the following website (early publication) prior to the official publication of Memoirs which is planned in March.

"Distributed Denial of Service (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection"

http://www.terrapub.co.jp/e-library/kobe-u_memoirs_en/

MASUDA Sumio,
Chair,
Editorial Committee of Memoirs of
the Graduate Schools of Engineering and System Informatics,
Kobe University

(a) Normal Email



British Journal of Mathematics & Computer Science

24 Apr

From: Managing Editor (Ms. S Sadhu)

To: Siti Hajar Aminah

Dear Colleague,

British Journal of Mathematics & Computer Science (ISSN: 2231-0851) is an OPEN peer-reviewed, INTERNATIONAL journal. We offer both online publication as well as Hard copy journal. **Article Processing Charge is 100 USD at present. This journal is now publishing Volume 8.**

2. Transparent and High standard Peer review:
This journal follows highly respected and toughest Advanced OPEN peer-review system (Example Link1, Link2, Link3, Link4, Link5, etc). Additionally we strongly encourage "Post-publication Peer review" by our comment section.

3. Proposed Time Schedule:
Submission to first editorial decision with review comments: 3 weeks
Submission to publication: 6 weeks

4. Abstracting/indexing:
Many respected abstracting/indexing services covered our journals.

1. ISI Web of Science citation: (ScreenShot 1, ScreenShot 2)
2. Zentralblatt MATH
3. Polish Ministry of Science and Higher Education (Original link)
4. ProQuest (Screen shot)
5. EBSCOhost (USA)
6. DOAJ
7. Ulrich's
8. CrossRef
9. Google scholar
10. JournalSeek
11. WorldCat (OCLC Number: 778757507)
12. OpenJGate

For more information please visit.

5. Authors' profile: Our journals have been chosen by academicians of many famous universities like Harvard, Columbia University, Cambridge, University of Chicago, Yale University, University of Göttingen, etc. Please see.

(b) Spam Email

Figure 3.3 Example of normal and spam email.

Table 3.1 Example of pre-processing for double-bounce email.

Example from Fig. 3.3(a): "Dear Ms./Mr. ALI, Thank you very much for your submission to Memoirs. I would like to inform you that your paper has been published on the following website (early publication) prior to the official publication of Memoirs which is planned in March. "Distributed Denial of Service (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection" http://www.terrapub.co.jp/e-library/kobe-u_memoirs_en/ MASUDA Sumio, Chair, Editorial Committee of Memoirs of the Graduate Schools of Engineering and System Informatics, Kobe University".	
Tokenize: (i) Break the sentences into single word. (ii) Remove stop words.	"dear", "ms", "mr", "ali", "thank", "very", "much", "submission", "memoirs", "like", "inform", "paper", "published", "following", "website", "early", "publication", "prior", "official", "publication", "memoirs", "planned", "march", "distributed", "dnial", "service", "ddos", "backscatter", "detection", "system", "resource", "allocating", "network", "data", "selection", "http", "www", "terrapub", "co", "jp", "e", "library", "kobe", "u", "memoirs", "en", "masuda", "sumio", "chair", "editorial", "committee", "of", "memoirs", "of", "the", "graduate", "schools", "of", "engineering", "and", "system", "informatics", "kobe", "university". Removed stop words "you", "for", "your", "to", "i", "would", "that", "has", "been", "on", "the", "of", "which", "is", "in", "using", "with"
Lemmatize: Transform the words into root word.	"submission" to "submit", "published" to "publish", "following" to "follow", "planned" to "plan", "distributed" to "distribute", "denial", to "deny", "detection" to "detect", "allocating" to "allocate", "editorial" to "editor", "schools" to "school", "engineering" to "engineer".
Feature selection using linear ℓ_1-SVM	Let say terms "submit", "website", "publish", "www", "screenshot", "link", "visit" and "see" are selected which produce the following matrix $f_{d,t} = \begin{bmatrix} 1 & 1 & 3 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 0 & 2 & 5 & 1 & 1 \end{bmatrix}$.
Feature representation: *TF-IDF*	(i) *IDF*: $IDF(submit) = \log\left(\frac{2}{1+2}\right) = -0.1761$, $IDF(website) = \log\left(\frac{2}{1+1}\right) = 0$. *IDF* for term "submit" is equal to "publish", whereas *IDF* for the remaining terms is equal to *IDF* for term "website". Thus, the idf is given by $f_{d,t} = \begin{bmatrix} -0.1761 & 0 & -0.1761 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. (ii) *TF-IDF*: $TF\text{-}IDF(d,t) = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{3}{6} & \frac{1}{6} & 0 & 0 & 0 & 0 \\ \frac{2}{15} & 0 & \frac{4}{15} & 0 & \frac{2}{15} & \frac{5}{15} & \frac{1}{15} & \frac{1}{15} \end{bmatrix} \times \begin{bmatrix} -0.1761 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.1761 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ $= \begin{bmatrix} -0.0294 & 0 & -0.0881 & 0 & 0 & 0 & 0 & 0 \\ -0.0235 & 0 & -0.0470 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. (iii) Input vector (training data): Let say from SPIKE, doc_1 is label as non-malicious and doc_2 as malicious. Training data $\boldsymbol{x}'_1 = (\boldsymbol{x}, \boldsymbol{d}) = ([-0.0294\ 0\ -0.0881\ 0\ 0\ 0\ 0\ 0], [0\ 1])$ and $\boldsymbol{x}'_2 = (\boldsymbol{x}, \boldsymbol{d}) = ([-0.0235\ 0\ -0.0470\ 0\ 0\ 0\ 0\ 0], [1\ 0])$. This system dataset would be $\boldsymbol{X}' =$ (**x**,class label) $= \begin{bmatrix} -0.0294 & 0 & -0.0881 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0235 & 0 & -0.0470 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

combining RAN-LSH classifier[57] and SPIKE, so that the learning time is accelerated compared to when using SPIKE alone. In this study, an outlier is detected based on the output margin θ_m, outlier threshold θ_o and the occurrence frequency threshold θ_N. The data with low output margins are considered as unknown emails for the current classifier and thus should be categorized as outlier. In addition, the number of similar data in each entry N_e is also important to decide whether the data is outlier or not. It is assumed that the data that do not frequently occurred (i.e., data allocated to an entry with small N_e) can also be categorized as outlier although the output margins are slightly higher than θ_m. The outlier flag F_e^O is calculated as follows (see Algorithm 9 Line 12):

$$F_e^O = \begin{cases} 0 & (F_e^M = 0) \\ 0 & (\theta_m \leq \Delta z(\bar{\boldsymbol{x}}_e) < \theta_o \ \ \& \ \ N_e < \theta_N). \\ 1 & (\text{otherwise}) \end{cases} \tag{3.3}$$

where $\bar{\boldsymbol{x}}'_e$ and $\tilde{\boldsymbol{x}}_i$ are the previous prototype which is registered in the entry e and training data that belong to sth subset of unique hash values $\tilde{\boldsymbol{x}}_i \in \tilde{\boldsymbol{X}}_s$, respectively. Meanwhile, N_e is the previous occurrence frequency of the entry e and n is the number of training data that has a similar hash value. Whenever a new training data is assigned to an entry, the occurrence frequency is increased by one. $\bar{\boldsymbol{x}}_e$ is regarded as the representative point of the subregion.

The algorithm of the outlier detection is summarized in Algorithm 9.

3.2.6 RAN-LSH Classifier

In this subsection, a brief explanation of the RAN-LSH classifier[57] that illustrated in Fig. 3.1(c) is given. RAN-LSH is the extended model of the RAN[12], where LSH is adopted to select essential training data and RBF bases for fast learning. There are three main components in RAN-LSH: hash table, data selection and classifier.

Algorithm 2 shows the overall learning procedures of RAN-LSH. During the initial learning phase (Lines 1-2), initial training data are used to obtain the most

suitable values of the following two important parameters: RBF width σ and the number of partitions P. In addition, initial data are also used to obtain an initial hash table and initial structure of the classifier. After that, the incremental learning is carried out whenever training data are given to learn (Lines 3-10). In LSH, similar data are allocated in the same hash entry with a high probability. Therefore, the number of hash entries determines the granularity of input space representation, and too large number of hash entries would result in both high computational and memory costs in the data selection. Therefore, it is important to design the hash functions such that a suitable number of hash entries are created.

In RAN-LSH, PCA is adopted to generate a proper number of hash functions by controlling the threshold of the accumulation ratio θ_a. Accumulation ratio $A(\boldsymbol{U}_l)$ is the ratio of input components in the approximated subspace over those in the whole input space[47]. Giving a proper value of θ_a based on a tolerant approximation error, a proper number of hash functions is automatically determined by selecting the number of partitions P via the cross-validation.

Let l be the subspace dimensions obtained by PCA. Then, the linear transformation is considered to define hash values in LSH shown by Eq. (2.1). Each projection vectors $v_i (i = 1, \cdots, l)$ is then divided into P partitions with equal size. θ_a controls the number of eigenvectors $\boldsymbol{U}_l$. To obtain a hash code, data is first projected on all the eigenvectors $\boldsymbol{u}_i$ and the hash code is obtained by combining the encoded values for the projections. When a large θ_a is adopted, the number of eigenvectors tends to become large and this would cause elongation of the length of a hash code.

As shown in Algorithm 2 (see Lines 3-10), the incremental learning of RAN-LSH is carried out not only for RAN-LSH classifier but also for the hash table. Algorithm 4 illustrates the steps to create and update the hash table which is used in RAN-LSH learning algorithm (Lines 1 (during *Initialization*), 5 and 9 in Algorithm 2). Each subregion is allocated to an entry in a hash table, where each

entry is composed of five items: hash value $\boldsymbol{h}(\bar{\boldsymbol{x}}_e)$, prototype $\bar{\boldsymbol{x}}_e$, margin flag F_e^M, outlier flag F_e^O and the occurrence frequency N_e (Fig. 3.1 : top right). The index or a hash value is used as a key to find a matched entry e of a similar item which has been registered previously in hash table (see the first condition in Algorithm 4 Line 6). Hash values $\boldsymbol{h}(\boldsymbol{x}) = \{H(v_1), \cdots, H(v_l)\}$ are a set of hash functions $H(v_i)$, which are given by Eq. (2.2).

The next item is prototype $\bar{\boldsymbol{x}}_e$. A prototype is the mean vector of all data allocated to each entry (see Algorithm 4 Lines 7 and 11) which is calculated as follows:

$$\bar{\boldsymbol{x}}_e = \frac{(\bar{\boldsymbol{x}}'_e * N_e) + \sum_{i=1}^{n} \left(\tilde{\boldsymbol{x}}_i\right)}{N_e + n}. \tag{3.4}$$

The third item is the margin flag F_e^M and it is calculated by Eq. (2.5) (see Algorithm 4 Line 12). The fourth item is the outlier flag F_e^O which has been discussed in subsection 3.2.5. The last item is the occurrence frequency N_e of similar data in an entry. Whenever a new training data is assigned to an entry, the occurrence frequency is increased by one.

When a large number of data are given simultaneously under an incremental learning environment, it is important to learn only essential data in a classifier model. Obviously, this is because the learning must be completed as quickly as possible; otherwise, the next data may be given before the learning is completed. In RAN-LSH, the data selection is conducted by using LSH. First, all N_d training data in a given training set are projected to l eigenvectors. Then, for each training data, the projection value is encoded into a hash code whose granularity is determined by the number of partitions P, and the obtained hash codes are transformed into a hash value.

If a matched entry with the same hash value is found and the margin flag F_e^M is '1', it means the classifier is well trained (Line 6 in Algorithm 4). Then, the mean vector is calculated and the occurrence frequency N_e is incremented by n. Otherwise (Line 10 in Algorithm 4), the output margin Δz in Eq. (1.5) is

calculated and the margin flag F_e^M and outlier flag F_e^O are updated by Eqs. (2.5) and (3.3), respectively. Note that the training data associated with the margin flag $F_e^M = 1$ are eliminated from the training set (see Algorithm 5 Lines 3-5). On the contrary, if the margin flag F_e^M is '0', it means a given data should be trained. After the learning phase, the margin flag F_e^M would be updated. Nevertheless, updating the margin flag of every prototype in the hash table would increase the learning time. As mentioned before, the prototype with $F_e^M = 1$ means the classifier is "well-trained" around the prototype. Thus, this prototype does not need to be updated. Meanwhile, prototype with $F_e^M = 0$ should be updated because there would probably be regions that have become "well-trained" after the learning phase (Line 9 in Algorithm 2 and Lines 10-16 in Algorithm 4).

LSH is also used to find RBF bases near to the training data (Lines 3-6 in Algorithm 6). In RAN-LSH, only the connection weights connected to the selected RBF bases are updated in the following procedures. Firstly, the hash values of RBF bases $\boldsymbol{h}(\boldsymbol{c}_j)$ are retrieved from a hash table. Next, the LSH distance d_j for each jth RBFs is calculated by Eq. (2.6).

Then, only RBF bases whose LSH distance is less than a threshold θ_p are selected for a learning purpose. This is because it is considered that if the LSH distance is large, the RBF output would become very small and the weight update could be negligible. Finally, the selected RBF bases are used to solve the linear equation in Eq. (1.3)[11].

3.3 Performance Evaluation

3.3.1 Experimental Setup

The detection performance is evaluated under incremental learning settings to study the following effects: (1) the effect of threshold parameters and (2) the effectiveness of daily updates.

In the former experiment, the effects of the following three threshold parameters to the performance is investigated: threshold of accumulation ratio θ_a, that of

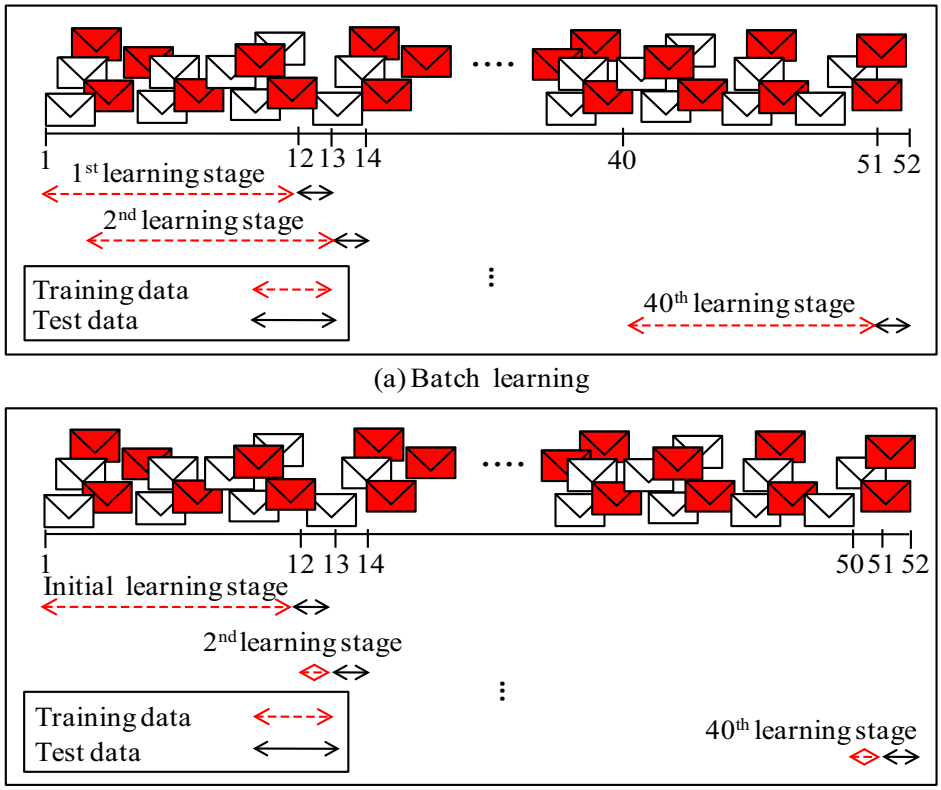


Figure 3.4 Comparison of learning scheme between batch learning and incremental learning.

output margin θ_m, and that of tolerant distance θ_p. In the latter experiment, the effect of incremental learning is studied through comparison with the batch learning scheme. Figure 3.4 illustrates how labeled spam emails are trained in (a) batch learning scheme and (b) incremental learning scheme. In the batch learning scheme, the conventional RBF network (RBFN) is adopted (i.e., RBFN usually used as batch learning[15]) as a classifier and a sliding window is introduced to define a data set to be trained every day. In this experiment, the time-window size is preliminarily determined as 12 days via the cross-validation using the spam emails collected during a different period. Therefore, as seen in Fig. 3.4(a), the first learning stage is carried out on Day 12 using a set of spam emails collected from Day 1 to Day 12, and the data set from Day 13 is used to test the performance. Then, the time-window is shifted by one day at the second

learning stage; that is, a set of spam emails collected from Day 2 to Day 13 is used for training, and the data set from Day 14 is used to test. Note that RBFN is retrained with a set of 12-day spam emails at every learning stage in a batch mode. On the other hand, in the incremental learning scheme, batch learning is first applied to an initial data set, which is composed of spam emails collected in the first 12 days. After that, this initial detection model is updated daily using one day training data and the system is tested using the next day data. A set of spam emails collected is forwarded to SPIKE to get the labels on maliciousness, and the pairs of a spam email and its class label are used as a training set on the next day. The number of collected spam emails is different every day. Their maximum, minimum, and average numbers are 756, 26, and 207, respectively.

In this detection system, three parameters are determined empirically. The parameters are accumulation ratio θ_a, output margin θ_m and tolerant distance θ_p. In the following experiment, the parameters are set to: $\theta_a = 0.9$, $\theta_m = 0.2$, and $\theta_p = 2$. Meanwhile, the error threshold ε is set to 0.5. The other parameters, RBF width σ, the number of partitions P, errors penalty of SVM C, outlier threshold θ_o, occurrence frequency threshold θ_N and time-window size are determined in the initial learning phase through the cross-validation. There are 20,448 double-bounce emails with 8,334 malicious spam emails and 12,114 non-malicious spam emails used in this study that were collected from 1st March 2013 to 10th May 2013.

It is important for the proposed system to correctly classify not only positive examples (malicious spam emails) but also negative ones (non-malicious spam emails). Therefore, in this study, three evaluations are used which are: recall rate, precision rate and F1 measure rate. The actual class labels are the class labels given by SPIKE, whilst the prediction labels are obtained by the detection system. The recall rate and the precision rate measure the ability of the detection system to classify the malicious spam emails (positive samples) correctly which takes into account different types of error. On the other hand, the F1 measure

is the harmonic mean of recall and precision rate. The recall rate considers type II error (i.e., a malicious spam email is wrongly classified as non-malicious spam email), whereas the precision rate considers type I error (i.e., a non-malicious spam email is wrongly classified as malicious spam email). If the malicious spam email detection system obtained a low recall rate, the users are exposed to the danger of malware infection because some users may click an URL that leads to malicious websites. Such a misclassification must be avoided in any cases. In the second situation, there is low risk of the malware attack. The system only gives strict conditions where most of the non-malicious spam emails are categorized as malicious spam emails. Therefore, to design a good malicious spam email detection system, it is crucial to have at least high percentage of recall rate to reduce the risk of malware attack. However, if the malicious spam email detection system is able to obtain high rate of both recall and precision, it can be concluded that the developed system is nearly a perfect detection system.

3.3.2 Effects of Threshold Parameters

First, let us examine the threshold parameters and their effect to the detection system. Here, the influences of θ_a, θ_m and θ_p are studied so that the parameters are optimized to ensure fast learning property of the detection system while having low misclassification rate. The first parameter is the threshold of accumulation ratio θ_a. If θ_a is set to a large value, an eigenspace to define hash functions has high-dimensions. Thus, the length of a hash code becomes long, resulting in the enlargement of a hash table. Therefore the searching of similar data would require a longer time since there are many hash values registered in hash table. The next parameter is the output margin threshold θ_m. This parameter controls the amount of selected data to be learned by RAN-LSH. As the value of θ_m is set to a higher value, the representation of the "not well-learned" region would become wider and the number of selected data is increased in the incremental learning phase, resulting in slower learning. The third important parameter is

the tolerant distance θ_p which determines the distance of near RBF bases. By updating the weights of only near RBF bases (i.e., using small value of θ_p), the time needed to solve the linear equation using Singular Value Decomposition (SVD) is shorten, thus the learning time would be accelerated.

To determine an appropriate value of each parameter, the cross-validation is performed for the initial training set, and the obtained parameter values are fixed over the incremental phase. Table 3.2 (a) and (b) show the F1 measure and the learning time, respectively, using several combination values of the accumulation ratio θ_a and output margin θ_m. As seen in Table 3.2(a), the highest F1 measure is obtained when $\theta_a = 0.9$ and $\theta_m = 0.2$. For output margin $\theta_m \leq 0.2$, F1 measure does not differ much from the F1 measure with output margin $\theta_m = 0.2$. This result is not surprising because high value of output margins represents that the data are "well-learned" or correctly classified. By adding these data, it is expected that the classification rate would not improve although the number of selected data and RBF bases created are increased. As a result for learning using more training data shown by a higher θ_m, the learning time would also increase. It is assumed that the data which are located on the border of different classes should have network outputs in the range of '0.4' to '0.6'. Therefore, the output margins θ_m can be estimated to be between '0' to '0.2' where the data with output margins in these range are important to be learned to reduce the misclassification rate.

On the other hand, Table 3.3 demonstrates the suitable value of the tolerant distance θ_p using appropriate value of θ_a and θ_m obtained previously which are '0.9' and '0.2', respectively. Tolerant distance θ_p also give influence to the classification rate and the speed of the detection system by controlling the distance which defines the area of near RBF bases. As we can see, the suitable value for the tolerant distance is 2. It means that only the RBF centers that differ from the given training data at two projection vector v_i are used to update the weight. If the θ_p is too small, it indicates that the area of selected RBF bases is not enough

Table 3.2 The evaluation using several values of accumulation ratio θ_a and output margin threshold θ_m for the spam email detection system. The performance measures are: (a) the F1 measure, and (b) initial learning time.

(a) F1 measure [%]

θ_a	θ_m				
	0.05	0.1	0.2	0.3	0.4
0.5	82.8	87.0	88.4	88.2	88.1
0.7	86.6	87.3	90.5	87.8	88.2
0.9	87.0	87.7	90.8	88.0	87.9

(b) Initial learning time [sec.]

θ_a	θ_m				
	0.05	0.1	0.2	0.3	0.4
0.5	126.5	128.6	130.1	130.3	131.2
0.7	128.0	129.7	130.3	130.9	132.6
0.9	128.6	129.8	132.1	132.7	139.8

Table 3.3 The performance using different values of tolerant distance θ_p.

Evaluation	θ_p			
	0	1	2	5
F1 measure [%]	88.5	88.8	89.3	89.3
Initial learning time [sec.]	138.2	139.1	140.1	144.9

to approximate the weights correctly. Whereas for θ_p that is too large, it would be similar to the approach of updating weights using all RBF bases. Thus, the size of RBF outputs $\mathbf{\Phi}$ in Eq. (1.3) would be bigger and therefore, the decomposition steps using SVD would require a longer time. Even though the results show the evaluation performance during initial learning, it is expected a similar result from the incremental learning phase. It is because parameter θ_a, θ_m and θ_p are also required during the incremental learning phase. For the next experiment, the values of θ_a, θ_m and θ_p are set to be '0.2', '0.9' and '2', respectively.

3.3.3 Effectiveness of Incremental Learning

All learning parts in the detection system including pre-processing and classifier module are very crucial which give effect to the performance result. In this experiment, the performance of the proposed online detection system is compared

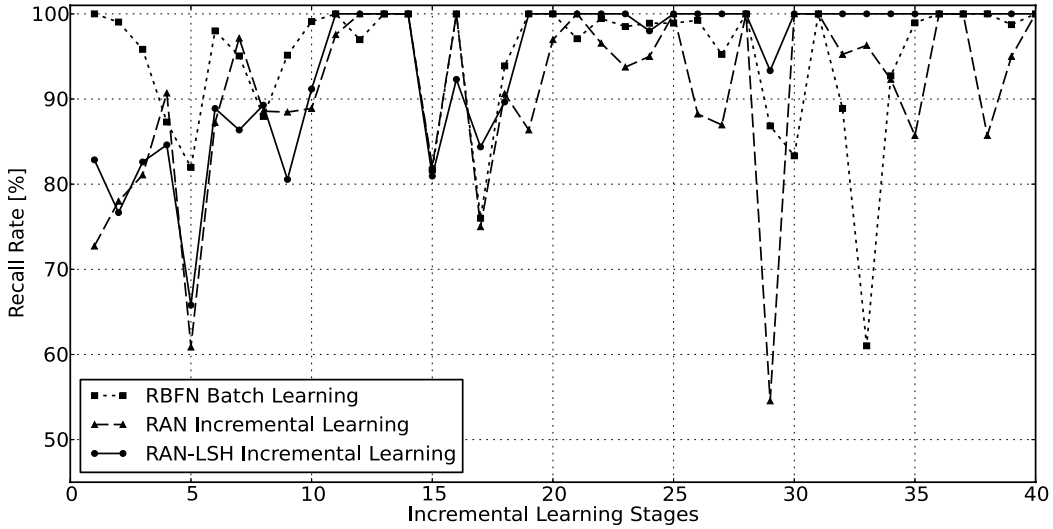


Figure 3.5 Transitions of recall rates in the malicious spam email detection system with three learning schemes.

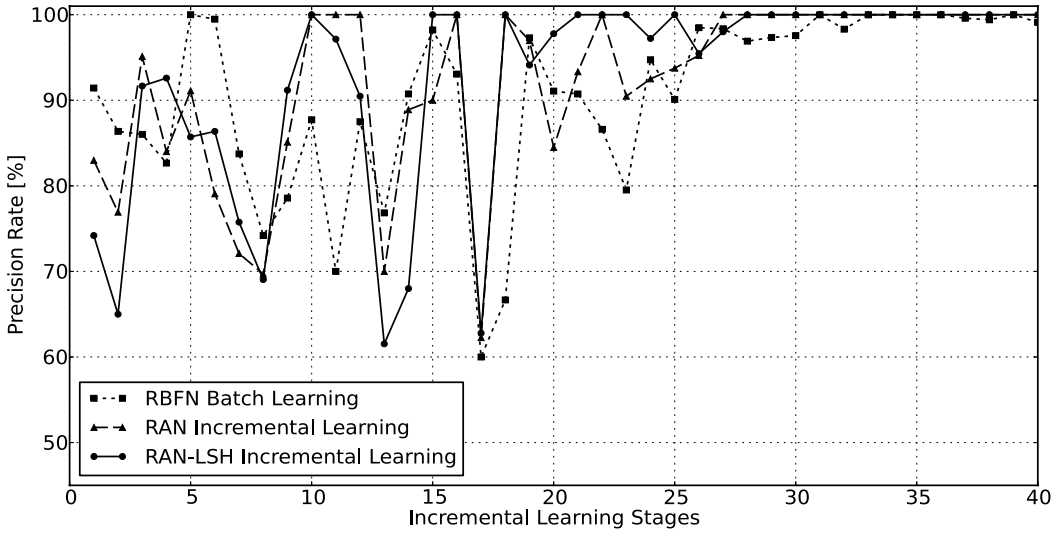


Figure 3.6 Transitions of precision in the malicious spam email detection system with three learning schemes.

with different learning scheme and classifier model to see the competency of the proposed method. Figure 3.5 and Fig. 3.6 show the recall rate and precision rate for the detection system with the following three combinations of classifiers and learning schemes: RBFN (batch learning), RAN (incremental learning), and RAN-LSH (incremental learning) (see Fig. 3.4). The batch learning is carried

Table 3.4 Overall performance of malicious spam email detection system.

Performance evaluation	RBFN batch	RAN incremental	RAN-LSH incremental
Recall Rate [%]	94.6	89.4	90.7
Precision Rate [%]	90.7	90.8	87.2
F1 Measure [%]	92.2	89.3	87.7
Learning Time [sec.]	4532.7	432.0	9.4

out using 12-days of training data and it is retrained incrementally. While for the incremental learning, the classifier is updated incrementally using 1-day of training data. As seen in Fig. 3.5 and Fig. 3.6, the proposed one-pass learning of the detection system is capable to learn and carry out the classification task effectively since the proposed system obtained almost the same classification rate as the memory-based learning approach (i.e., batch learning). In fact, the proposed method does not need large memory size to store the training data compared to the memory-based learning. In this study, 12-days length of window size is used for the batch learning to learn incrementally, whereas for incremental learning, only 1-day data set is used as training data. Besides that, the performance of conventional classifier RAN is compared using the same incremental learning scheme. My previous study in[57] shows that RAN-LSH can learn fast. As seen in Fig. 3.5 and Fig. 3.6, the detection system obtains a comparable result against the conventional classifier model.

3.3.4 Overall Performance of Malicious Spam Email Detection System

The overall performance is evaluated by averaging over the whole incremental learning phase. The recall rate, precision rate and F1-measure for the three learning models are summarized in Table 3.4. From Table 3.4, it can be seen that the proposed detection system can learn 482 times faster than the conventional RBFN model, and can learn 46 times faster than RAN. This is because the detection system can find a set of similar data in a given training set very quickly

using LSH; thus, only a data set falling in an untrained region are selected to learn and the others are discarded. In addition, only near RBF centers are used to update the connection weights. The recall rate and F1 measure of RBFN are higher than those of the other incremental models because this model keeps a large number of data for training; where this requires high computational costs and large memory. Since the classification rate of recall rate, precision rate and F1 measure do not differ much from the other model, it can be concluded that the proposed system is able to update efficiently and able to give class label of the incoming emails within a short time.

3.4 Conclusions

A malicious spam email detection system using BoW features is proposed, where the classifier adopts LSH to select essential data and near RBF bases. Two types of essential data are used: (1) the data located close to a class boundary, and (2) the data located outside of the learned region (i.e., outlier). The proposed scheme provides desirable learning characteristics as an autonomous malicious spam email detection system and able to adapt to new trends of malicious emails quickly. In addition, the detection system is quite fast compared with SPIKE which often needs a long time to complete the maliciousness analysis. By using the proposed system, it is possible to give proper alerts to users quickly based on up to date information. Since the learning is quite fast and the detection performance is comparable to the conventional models, it can be concluded that the proposed system is suitable to be implemented in an email client software on the user side.

Currently, the proposed detection system has no pruning function for RBF bases. Therefore, as the learning is continued for a long time, the number of RBF could be increased excessively, and this causes longer learning time. Then, in the worst scenario, the learning may not converge before new training data are given. To avoid such a disastrous situation, a proper number of RBF bases

should always be maintained by introducing an online pruning mechanism into RAN-LSH. Besides that, the detection system uses selected features from initial learning training data. As future work, an adaptive hash table is inteded to be constructed to adapt to the changes of feature vectors from the recent BoW without forgetting the previous knowledge. It is expected that the detection system would be more stable and robust to the new malicious spam email attacks.

Chapter 4

Distributed Denial of Service (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection

4.1 Introduction

As many people greatly rely on the Internet these days, potential threats to have cyber attacks are increasing. One of these serious threats is DDoS attack. DDoS attack has been used by cyber criminals to cause damage virtually to the targeted organization especially organizations which rely on Internet such as news sites, political parties, financial institutions, gaming cites and online businesses. DDoS attack can cause material losses (i.e., finance and resources bandwidth), ruin the reputation of the targeted company or organization and decreased the consumers confidence. By saturating the targeted bandwidth with fake requests, this attack can cause the legitimate users to experience a slowed down internet services when requesting connection to targeted websites, or even in worst cases, the targeted websites would suffer a total blockage for a minutes, several hours or even a week. Recently, the worst attack in history was recorded on 20th November 2014, when 500 gigabits per second of junk traffic were pounded at Apple Daily and Pop Vote which are two independent news site at Hong Kong[64)]. In the previous year, 300 gigabits per seconds of DDoS attack had been recorded attacking DNS system operated by Spamhaus which caused a slowed down Internet around the world[65)]. Given the devastating effects of DDoS attack on our cyber infrastructure, it is very important to detect the DDoS attack as quickly as possible before the Internet

traffic is totally down.

Backscatter is the after-effect of DDoS attack when the source Internet Protocol (IP) addresses are spoofed by the attacker. Backscatter happens when the target server sends unsolicited response packets to the spoofed IP addresses. Some of these random IP addresses are directed to the darknet (i.e., a set of unused IPs). By spoofing the source IP addresses, the attackers not only can amplify the DDoS attack, but also are able to conceal their identity. According to the reports on the Cloudflare website[66], 92% of DoS attacks using Transmission Control Protocol (TCP) at Cloudflare are through Port 80 (Hypertext Transfer Protocol (HTTP)), and 97% of the Denial of Services (DoS) attacks using User Datagram Protocol (UDP) are through Port 53 (Domain Name System (DNS)). Basically, it is expected that the companies and organizations with service-based websites would have a trend of attack similar to Cloudflare. It is because web servers often use TCP Port 80 (80/TCP), whereas DNS servers usually use Port 53 (53/UDP).

The most common DDoS attack is using "flood" attack in which a large number of packets containing useless information are sent by this flood through various internet protocols such as TCP and UDP[67]. The main purpose of the flood attack is to occupy the server resources so that the server is unavailable to the legitimate user. To send DDoS attack through 80/TCP, firstly, the attacker would request a connection by sending packets with SYN flag set to '1' to the World Wide Web (www) of the targeted server using a spoofed source IP address. Then, the targeted server would send a response packet with SYN-ACK flags to the source IP addresses. The targeted server is under the DDoS attack when the attacker sends a SYN flood to the targeted server using botnet and consumes the available resources by refusing to respond the SYN-ACK flags. Since the available resources are consumed by these overwhelming connection requests and waiting period for reply packets with ACK flag, the targeted server would denies any request from the legitimate users. Another frequent attack is using 53/UDP in which the attacker manipulates the operation of the DNS server to amplify the DDoS

attack. DNS server is used to search the corresponding IP address of the requested internet domain name. Internet Control Message Protocol (ICMP) "Destination Unreachable" message would be generated the moment after the system had noticed inactive ports. Thus, the additional packets containing these messages would cause traffic congestion at the targeted server when a large number of UDP packets are sent to these inactive ports.

Both attacks from labeled packets (80/TCP and 53/UDP) are easy to be detected based on the flags in these packets. However, for other protocols with different port numbers, usually an expert is hired to analyze the characteristics of packet traffic manually. This method is time consuming and is costly because a professional needs to be hired. Therefore, an autonomous DDoS backscatter detection for all protocols and port numbers is proposed using RAN-LSH classifier. The RAN classifier is embedded with the well-known LSH technique which is used to select essential data. The LSH method indeed has proven to be able to quickly find similar objects in a large database[4, 5, 6, 7, 8].

Although extensive research has been carried out on DDoS attack detection using machine learning model[68, 69, 70, 71], to my knowledge, no attempt was done to explore the potential of data selection to speed up the learning time of the DDoS attack detection system. To address this gap, RAN-LSH classifier is used which embedded with data selection to learn only essential data. Learning irrelevant data would not only lead to a slow learning time[58] but also can cause the deterioration in detection performance of a classifier. Oyang et. al.[59] suggest that the training samples that are located near to the boundaries between different classes would carry more crucial information. By calculating the difference between the highest network output and the second highest network output, the training data is judged whether are located near the class boundaries or not. With the expectation that if the difference between them is large, it means that the given data is "well-learned", whereas the small difference indicates that the data is "not well-learned". It is assumed that the "not well-learned" data is located

at the class boundaries and is essential to be learned. In this study, an output margin is defined by the difference between the two highest network outputs. By combining the margin-based data selection with LSH technique, the learning time is expected to be accelerated.

To evaluate this detection system, darknet packets collected by NICT are used in this study. Darknet is a group of IP addresses which is not connected to any active services such as computer, printer or router. Thus, it is unusual for darknet to receive any packets since darknet is not capable to start any communication with other server. Based on this fact, darknet is an effective medium for collecting data set of DDoS attack because a large number of true positive data (i.e., DDoS backscatter packets) can be collected with only small number of false positive data (i.e., DDoS non-backscatter packets). There are at least two possibilities of false positive data, which are from misconfigurated IP setting and the scan activity by botnet to find available IP addresses to be spoofed. Since NICT has the largest network monitoring system in Japan that assembled 140 thousands unused Internet Protocol version 4 (IPv4)[72], a good data set is expected to be collected using this resources.

The rest of this chapter is organized as follows: Section 4.2 describes the overall architecture of the proposed DDoS detection system. Section 4.3 validates the effectiveness of the detection system to detect the DDoS attack from different protocols and port numbers, other than labeled packets. Finally, conclusions and future work are discussed in Section 4.4.

4.2 The Proposed DDoS Backscatter Detection System

4.2.1 System Architecture

Figure 4.1 shows the overall architecture of DDoS backscatter detection system. IP addresses can be divided into the following two categories: livenet and darknet. Livenet consists of a group of IP addresses that are connected to actual hosts and devices (i.e., desktop, laptop and printer). While the remaining IP addresses

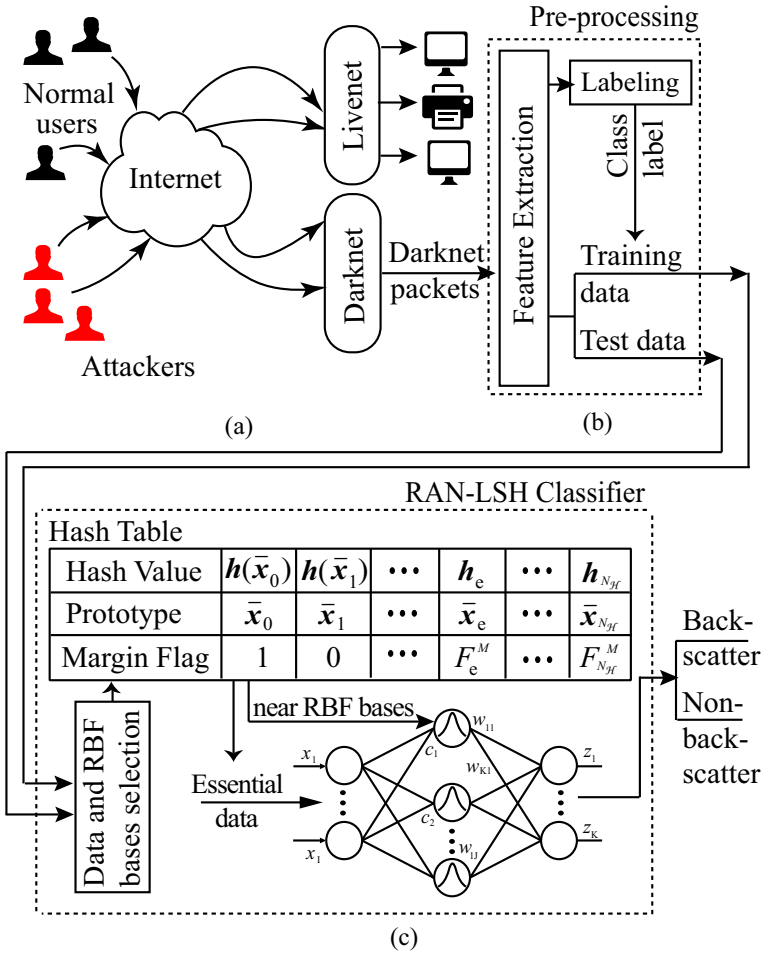


Figure 4.1 The architecture of DDoS backscatter detection system.

other than livenet are included in darknet category. In general, there should not be any legitimate packets sent to a darknet. However, there is a possibility of configuration errors during data transmission, which mistakenly end up to reach darknet. Nevertheless, there are users who intentionally use darknet to perform cyber criminal activities such as DDoS attack. In this study, darknet packets collected by NICT is used as data set. There are two main modules of the detection system, which are pre-processing and classifier. Firstly, the raw darknet packets are passed to the pre-processing module where the packets are transformed into feature vectors. Then, these feature vectors are given to the classifier for learning and testing. Algorithm 10 shows the overall procedures of

the detection system which consists of the following two learning phases: initial learning (Lines 1-3) and incremental learning (Lines 4-9).

4.2.2 Pre-processing

To make sure that the classifier of the detection system can learn the decision boundary of different classes (i.e., DDoS backscatter or non-backscatter), only important information are extracted from raw data packets. Algorithm 11 shows the pre-processing procedures to create the data set of the DDoS backscatter. First and foremost, the darknet packets are sorted according to the source IP addresses (Line 2). Next, 17 features are extracted from the first one-minute packets as one data (Line 5). Here, a new data is defined as two sequential packets which are separated by one hour of time interval between them (Lines 3-4). For training data, the feature vectors consist of 17 features and the corresponding class label $(\boldsymbol{x}_i, \boldsymbol{d}_i)$, whereas the test data do not contain the class label in the feature vectors $\boldsymbol{x}_i$. The 17 features which are extracted from raw data packets are as follow:

(1) Number of packets.

(2) Average for the difference of the time between packets.

(3) Variance for the difference of the time between packets.

(4) Number of Source Port.

(5) Average of the numbers of packets to each Source Port.

(6) Variance of the numbers of packets to each Source Port.

(7) Number of protocol.

(8) Number of destination IP addresses.

(9) Average of the numbers of packets to each Destination IP.

(10) Variance of the numbers of packets to each Destination IP.

(11) Average for the difference of Destination IP between packets.

(12) Variance for the difference of Destination IP between packets.

(13) Number of Destination ports.

(14) Average of the numbers of packets to each Destination port.

(15) Variance of the numbers of packets to each Destination port.

(16) Average of payloads (the cargo of a data transmission).

(17) Variance of payloads.

Besides extracting features, the training data would also be given a class label in the pre-processing module. Figure 4.2 illustrates an example of one DDoS backscatter data and one non-backscatter data in which the darknet packets are labeled based on the traffic features. As we can see, the figure has four different axes which are time and source port number at the first half in the left side, whereas destination IP and destination port number at the second half in the right side. The source is referred to target server, while destination is referred to the darknet. The straight line from a point at target server side to a point at darknet side corresponds to the duration time to sent the packet. Each lines are colored with different color to differentiate the internet protocol and TCP control flag which are used by each packets. Yellow, purple, green, red and black represent SYN-ACK packet in TCP, RST packet in TCP, ACK packet in TCP, UDP packet and ICMP packet, respectively. From this figure, a notable rules to differentiate backscatter with non-backscatter data, are obtained. For example, from Fig. 4.2(a), it can be seen that a large amount of packets is sent in a short time to a various destination IP address, which represent a typical behavior of DDoS attack. On the other hand, Fig. 4.2(b) shows that there is no DDoS attack because the time interval between two sequence packets is longer and involves only a small number of packet, compared to Fig. 4.2(a).

The steps to assign a class label to the labeled packets are shown by Algorithm 12. If the source host sends more than 40 packets per minute and its response flag F^R is equals to '1' (Line 11), then the feature vector for the one-minute darknet traffic is classified as DDoS backscatter. To set the response flag as $F^R = 1$, the data from 80/TCP and 53/UDP (labeled packets) need to fulfill the conditions

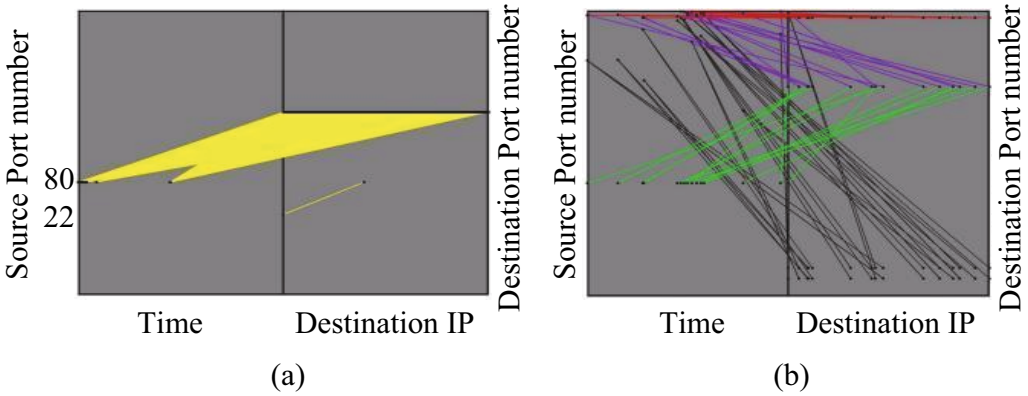


Figure 4.2 The example of darknet packets activity based on traffic features for: (a) DDoS backscatter and (b) DDoS non-backscatter.[69)]

in Line 3 and Line 7, respectively. The remaining data are labeled as non-DDoS backscatter.

Table 4.1 shows the overall procedures of darknet packets' pre-processing using example in Fig. 4.3. The pre-processing procedures consist of feature extraction and labeling. The darknet packets data that have been converted to feature vectors $\boldsymbol{X} = (\boldsymbol{x}, class\ label)$, would be passed to RAN-LSH to be learned as input. Firstly, the essential data are selected using LSH before passing to RAN classifier in which the weights update is carried out using near RBF bases. Table 2.2 shows an example of learning data using RAN-LSH classifier.

4.2.3 RAN-LSH Classifier

RBFN[73)] has at least three desirable properties: (1) the architecture of the model is simple and it can easily be modified from batch learning algorithm to incremental learning algorithm[74)], (2) the ability to approximate any function if a sufficient number of hidden units are given[14)], and (3) the capability of handling the classification problem of more than two classes in a single run[59)]. In this study, the incremental model of RBFN so-called RAN[12)] is used.

Algorithm 2 shows the learning algorithm of RAN-LSH which has been proposed in[57)]. This model is the extended model of RAN which adopted the LSH-

Table 4.1 Example of pre-processing and data representation for darknet packet.

Feature extraction	Refer to Fig. 4.3. (1) Number of packets. # of darknet packets in a minute: 5 packets. (2) Average for the difference of the time between packets. $\frac{t_1+t_2+t_3+t_4}{4} = a_1$. (3) Variance for the difference of the time between packets. std$\left(\frac{t_1+t_2+t_3+t_4}{4}\right) = b_1$. (4) Number of Source Port. Source port: Port 80 and Port 53. # Port= 2. (5) Average of the numbers of packets to each Source Port. 2 packets from Port 80 and 3 packets from Port 53. Average = $\left(\frac{2+3}{2}\right) = 2.5$ (6) Variance of the numbers of packets to each Source Port. std$\left(\frac{2+3}{2}\right) = 0.707$ (7) Number of protocol. Protocol such as TCP, UDP and ICMP. # of protocol is 2; TCP and UDP. (8) Number of destination IP addresses. # of darknet IP addresses=3; 1.2.3.4, 2.3.6.1 and 1.4.6.1. (9) Average of the numbers of packets to each Destination IP. 2 packets to 1.2.3.4, 2 packets to 2.3.6.1 and 1 packet to 1.4.6.1. Average = $\left(\frac{2+2+1}{3}\right) = 1.67$ (10) Variance of the numbers of packets to each Destination IP. std$\left(\frac{2+2+1}{3}\right) = 0.577$ (11) Average for the difference of Destination IP between packets. Firstly, the IP will be converted into long IP address. For example 133.3.9.1=133x256x256x256+3x256x256+9x256+1=2231568641=c_1 (12) Variance for the difference of Destination IP between packets=d_1. (13) Number of Destination ports. Destination port: Port 440 and Port 80. # Port= 2. (14) Average of the numbers of packets to each Destination port. 3 packets to Port 80 and 2 packets from Port 440. Average = $\left(\frac{3+2}{2}\right) = 2.5$ (15) Variance of the numbers of packets to each Destination port. std$\left(\frac{3+2}{2}\right) = 0.707$ (16) Average of payloads (the cargo of a data transmission). $\frac{payload_1+payload_2+payload_3+payload_4+payload_5}{5} = e_1$. (17) Variance of payloads. std$\left(\frac{payload_1+payload_2+payload_3+payload_4+payload_5}{5}\right) = f_1$.
Labeling	Refer to Fig. 4.2 and Section 4.2.2.
Data representation	$\boldsymbol{X}'$ = ($\boldsymbol{x}$,class label)= $\begin{bmatrix} 5 & a_1 & b_1 & 2 & 2.50 & 0.71 & 2 & 3 & 1.67 & 0.58 & c_1 & d_1 & 2 & 2.50 & 0.71 & e_1 & f_1 & 0 \\ 3 & a_2 & b_2 & 1 & 3.00 & 0.00 & 1 & 2 & 1.50 & 0.71 & c_2 & d_2 & 1 & 3.00 & 0.00 & e_2 & f_2 & 0 \end{bmatrix}$

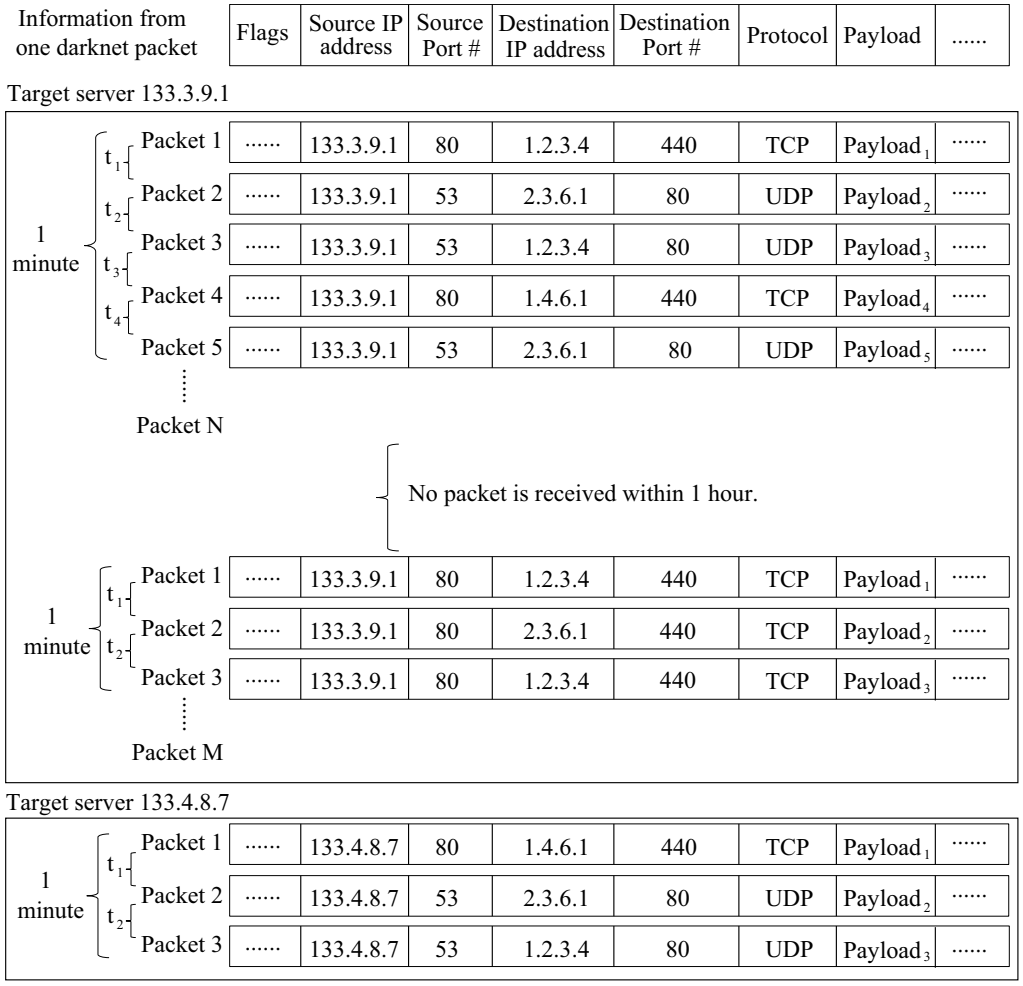


Figure 4.3 Example of darknet packet for pre-processing.

based data selection to accelerate the learning time. The RBF centers of RAN-LSH are not trained but selected based on the error between the prediction class label obtained by the classifier model using Eq. (1.2) and actual class label of the training data $\boldsymbol{d}$. Low error means that the existing RBF centers are "well-learned". On the other hand, high error value indicates that the RBF centers are "not well-learned" and thus, it is necessary to add the data as a new RBF center in the hidden layer node (Algorithm 1 Lines 5 and 17). Besides the RBF centers selection, another important element in RAN classifier is updating the connection weights. To shorten the learning time, instead of using all RBF centers to update the weights, only local RBF centers which are RBF centers that are located near

to the training data are used (Algorithm 2 Line 10).

RAN-LSH model is base on the idea that the data which are located near to each other would probably have almost similar value of RBF outputs and network outputs. Therefore, data that located near to each other are grouped into a same group which is called as region. LSH technique consists of two main steps: firstly, to group similar data into a same group, and secondly, to store them in a table called hash table where hash values are used as the index of the hash table. With the former step, the data are projected into several one-dimensional of projection vectors and each of the projection vectors is divided into equal size of partition or called as prtition. With the latter step, the hash values which correspond to a buckets are comprise of a hash function for every projection vectors. When two different data are always assigned in the same partition while the directions of the projected vectors are changing multiple numbers of times, it means that the data are most probably located close to each other. Nevertheless, assigning to many projection vectors would leads to the construction of a bigger size of hash table that requires not only larger memory but also longer processing time. Therefore, it is important to choose a proper number of hash functions. PCA can generate a proper number of hash functions by controlling the threshold of the accumulation ratio θ_a.

A sub-region is defined by a series of assigned partition which are divided based on the l projection values. Each sub-regions is stored as an entry in hash table. A hash table is composed of the following three items: hash value $\boldsymbol{h}_e$, prototype $\bar{\boldsymbol{x}}_e$ and margin flag F_e^M. The index or the hash values $\boldsymbol{h}_e$ are used as a key to find a matching entry e of a similar item which has been registered previously in hash table (Figure 4.1: top right). Hash values $\boldsymbol{h}$ in Eq. (2.3) are a series of hash function $H(v_i)$ given by Eq. (2.2).

The prototype $\bar{\boldsymbol{x}}_e$ is the average value of data in each entry of the hash table. If the hash values have not been registered in the hash table, the margin flag is

calculated as follow:

$$F_e^M = \begin{cases} 1 & (\Delta z(\boldsymbol{x}_e) > \theta_m) \ \ \& \ \ (z_{k'}(\boldsymbol{x}_e) > \theta_z) \\ 0 & (\text{otherwise}) \end{cases} \tag{4.1}$$

where the output margin Δz is given by the difference between the highest network output and second highest network output which is calculated using Eq. (1.5). If the flag of the matched prototype is '1', it means the classifier is "well-trained" around the prototype; therefore, there is no need to train a given data. On the contrary, if the flag is '0', it means a given data should be trained. After the learning phase, the margin flag F^M should be updated. Nevertheless, updating the margin flag of every prototype in the hash table would increase the learning time. As mentioned before, the prototype with $F^M = 1$ means the classifier is well trained around the prototype. Thus, this prototype does not need to be updated. Meanwhile, prototype with $F^M = 0$ should be updated because there would probably be regions that have become "well-trained" after the learning phase.

The LSH is also used to find near RBF bases to the training data. These RBF bases are used to update the connection weights. Firstly, the hash values of RBF bases are retrieved from the constructed hash table. Next, the LSH distance d_j^* for each jth RBFs is calculated using Eq. (2.6). Then the RBF centers with LSH distances that are less or equal to θ_p would be selected to solve the linear equation in Eq. (1.3).

4.3 Performance Evaluation

4.3.1 Experimental Setup

To evaluate the performance of the detection system with data selection mechanism, the following two experiments on is carried out: (1) the influence of the threshold parameters for output margin θ_m and network output θ_z and (2) the effectiveness of incremental learning.

The first experiment is carried out to see the effects of different values θ_m and θ_z using initial training data. In the second experiment, the performance of the three classifier models are compared: RBFN, RAN and RAN-LSH. RBFN is learned with data within a 90-day time window in a batch learning mode, and the time window is shifted at every 1 day for an online learning purpose. For instance, the first learning stage is carried out using data Day 277 until Day 366 as training data and data Day 367 as test data. During the next learning stage, the time-window is shifted by one, as such for the second learning stage, Day 278 until Day 367 is used for training, whereas the performance are tested using data Day 368. On the other hand, RAN and RAN-LSH are trained with data of the first 90 days, and then they are trained incrementally with a chunk of data given everyday. The first learning stage is similar to those in batch learning. The difference is the model is updated only with the data given in a day (i.e., the past data are not kept and used for a learning purpose). For example, in the second learning stage, Day 367 and Day 368 are used as training data and test data, respectively. The learning steps are continued using the next sequence of data until all data have been learned by the model. Since the training data are given based on window-size, the sizes of a data chunk are different at every learning stage. The number of data in a 90-day time window ranges from 8,513 to 13,558, and 11,123 on average. Whereas, the minimum, maximum and average numbers of data collected in a day are 0, 436 and 41, respectively.

In this detection system, four parameters are determined empirically which are accumulation ratio θ_a, output margin θ_m, network output θ_z and tolerant distance θ_p. In the first experiment, the effect of output margin θ_m and network output θ_z is investigated regarding the detection rate and learning time, so that the data selection mechanism is carried out effectively. In the following experiment, the parameters are set to: $\theta_a = 0.9$, $\theta_m = 0.01$, $\theta_z = 0.6$ and $\theta_p = 2$. Meanwhile, the error threshold ε and δ are set to 0.5 and $\sqrt{2}\sigma$, respectively. The other parameters, RBF width σ and the number of partitions P are determined in

the initial learning phase through the cross-validation. There are 9,968 data of labeled packets (9,404 backscatter and 564 non-backscatter) collected from 1st January 2013 to 31st December 2013 (1 year). For the test purpose, 5,933 data of unlabeled packets (2,464 backscatter and 3,469 non-backscatter) are collected from 1st January 2014 to 20th January 2014 (20 days).

Besides measuring the learning time, the performance of the detection system is also evaluated based on recall rate, precision rate and F1 measure. The recall rate measures the ratio correctly classified as DDoS backscatter (True Positive) out of expected DDoS backscatter determined by the classifier. Meanwhile, the precision rate measures the True Positive sample out the total of data with DDoS backscatter class label. The F1 measure is the harmonic mean of both recall and precision rates. For this detection system, it is more crucial to have at least high percentage of recall rate to reduce the risk of misclassifying DDoS attack, where a low recall rate indicates that the detection system has mistakenly classified many backscatter data as non-backscatter. Therefore, in this situation, a right action to salvage a targeted host could be delayed, resulting in a significant loss. However, having too low precision rate is also inappropriate because the detection system tends to have classified most of the data as backscatter.

4.3.2 Parameter Tuning for Data Selection

For RAN-LSH classifier which uses LSH data selection, two important parameters have been identified to affect the detection rate and learning time, namely the thresholds for output margin θ_m and network output θ_z. Table 4.2 shows the result of parameter tuning for θ_m and θ_z using three months of training data and 20-days of test data regarding three evaluation: (a) number of data selected, (b) F1 measure, and (c) learning time. Equation (2.5) indicates that the higher the threshold values of θ_m and θ_z are, the more stringent the requirements to determine the data as "well-learned". Thus, if high values are used for both parameters, it is expected that only a few data would have margin flag $F^M = 1$.

Table 4.2 Parameter tuning of data selection using three months training data and 20 days test data. The performance are measured regarding the (a) total number of selected data, (b) F1 measure [%] and (c) learning time [sec.]

(a) Total number of selected data

θ_m	θ_z		
	0.5	0.6	0.75
0.01	2704	3839	6690
0.05	3115	3839	6745
0.1	3319	3839	6891
0.2	3769	3844	6891

(b) F1 Measure [%]

θ_m	θ_z		
	0.5	0.6	0.75
0.01	89.4	91.2	93.1
0.05	89.5	91.3	93.1
0.1	90.1	91.3	93.1
0.2	91.1	91.3	93.1

(c) Learning time [sec.]

θ_m	θ_z		
	0.5	0.6	0.75
0.01	1122.5	5008.1	46071.2
0.05	1915.0	5021.5	46464.9
0.1	2510.4	5284.4	47926.5
0.2	4316.8	5528.8	48184.4

Consequently, the learning time is expected to be increased because many data should be selected to learn the detection system (i.e., data with margin flag $F^M = 0$). To accelerate the learning time, the number of selected data should be small by tuning both θ_m and θ_z.

By depending on only the output margin θ_m alone to determine whether the data are located in "well-learned" region or not, it is not enough to reduce the number of selected data effectively. Table 4.2(a) shows that the number of selected data is high even though a small value of θ_m is used (i.e., $\theta_m = 0.01$) for different values of θ_z. Almost twice the number of selected data is able to be reduced if another threshold is used namely θ_z. To ensure that the selected data provides a high detection rate, the F1 measure of θ_z value should not differ much compared to the other value of θ_z. For RAN-LSH model with $\theta_z = 0.6$ and $\theta_m = 0.01$,

Table 4.3 Comparison of three different model: RBFN, RAN and RAN-LSH, using different learning scheme. RBFN uses batch learning, whereas RAN and RAN-LSH use incremental learning setting. The performance are measured regarding (a) the recall rate [%], (b) precision rate [%], (c) F1 measure [%], and (d) learning time [s].

Evaluation	RBFN (batch-retrained)	RAN (incremental-with update)	RAN-LSH (incremental-without update)	RAN-LSH (incremental-with update)
Recall rate [%]	98.4	98.1	98.1	97.8
Precision rate [%]	96.2	97.2	65.9	97.4
F1 measure [%]	97.3	97.6	77.8	97.5
Learning time [s]	58135.5	207.9	3.3	46.2

almost 10 times of learning time can be shortened and about 0.9% of difference at F1 measure, compared to the model with $\theta_z = 0.75$ and $\theta_m = 0.2$. Taking the tradeoff between short learning time and high detection rate into account (see Table 4.2(b) and 4.2(c)), $\theta_z = 0.6$ and $\theta_m = 0.01$ are the most suitable for reducing the learning time of the detection system.

4.3.3 Performance of DDoS Detection System

Table 4.3 shows the performance comparison between three different classifiers: RBFN, RAN and RAN-LSH regarding recall rate, precision rate, F1 measure and learning time. RBFN uses batch learning, while RAN and RAN-LSH use incremental learning, in which the learning modes are explained in Section 4.3.1. The results obtained are expected and not surprising. The detection system using RBFN classifier obtains the highest recall rate compared to other classifiers. This is because the RBFN classifier learns using a large number of training data (data collected during the 90-day window size). Since this classifier is retrained in every learning stage, this classifier requires longer learning time than RAN and RAN-LSH. By learning incrementally, RAN and RAN-LSH classifier updated the weights of the classifier using only the recent training data. This can save a lot of time. Moreover, using RAN classifier with LSH-based data selection

(RAN-LSH), the detection system is capable to learn faster than RAN, because only essential data from the training data are selected to be learned, instead learning all given training data using RAN classifier. Based on the result shown in Table 4.3, to give class labels to 5,581 test data, RBFN and RAN requires approximately 10.4 (sec.) and 0.04 (sec.), respectively in average for each test data. As for RAN-LSH, only 0.008 (sec.) is required which is 130 times faster than RBFN classifier and five times faster than the RAN classifier. Since the recall rate is not significantly different, the detection system with RAN-LSH classifier is expected to detect DDoS backscatter faster, and hence allowing the moderator of the targeted server to take precautious action on the suspicious IP addresses before the Internet traffic of the targeted server is effected by the DDoS attack. Besides that, the effectiveness of updating RAN-LSH classifier in every learning stage is also examined. It is obvious that without updating the RAN-LSH classifier using the recent training data, the overall performance is dropped where we can see that the precision rate and the F1 measure decreased about 20%. Based on this result, it gives an evidence that the incremental learning of the detection system is carried out effectively.

4.4 Conclusions

In this chapter, a fast DDoS backscatter detection system for unlabelled packets is proposed using RAN-LSH classifier. It has been shown that the proposed detection system is capable to perform well, comparable to the detection system using conventional classifiers (RBFN and RAN). The current findings add to a growing body of literature on LSH, in which LSH is used to find similar data. However, in this study, LSH is used to select essential data to learn the detection system. The proposed detection system provides a desirable characteristic as a DDoS backscatter detection system. At every minute, darknet packets are transformed into a feature vector and it is provided to the RAN-LSH classifier to detect DDoS backscatter. The time to classify needs less than 0.01 (sec.).

The proposed system has some limitations when deployed under a real environment. First, the labelling of darknet packet features is not easy except for the packets of 80/TCP and 53/UDP. Basically, the labelling must be carried out by expert surveyors through visual observation. Therefore, only unreliable decision by a classifier should be presented to a surveyor for a labelling purpose. However, this function is not implemented in the proposed detection system at this preset. Second, there are a few parameters that need to be tuned during initial learning in order to achieve high detection rate. The parameters are RBF width σ, the number of partitions P, error threshold ε, distance threshold δ, accumulation ratio θ_a, tolerant distance θ_p, output margin θ_m and the highest network output θ_z.

The above issues are left as the future work.

Chapter 5

Conclusions

In this dissertation, the study of large-scale stream data is chosen as the main theme since this area are becoming compulsary for the researchers to be prepared of flood of stream data in near future. The large-scale stream data usually involves most application including the cybersecurity. Moreover, this field needs a rapid enhancement to support the rapid changes which made by the cybercriminals when a new campaign is announced. As mentioned before, there are at least three major issues to be solved in large-scale stream data which the system that has been developed need to be able:
(1) to monitor the incoming data non-stop,
(2) to handle a growing number of data, and
(3) to learn fast.
To equipped a proposed system with those three criteria, a machine learning is one of a good choice which is more effective compare to using human resources.

In general, learning all training data (i.e., batch learning) is not a good strategy under online learning environments, especially when a large number of data are given as a sequence. To enhance the adaptability to such dynamic environments, the learning should be carried out with essential data that are selected in an online fashion. There are two types of essential data for a learning purpose. One is the data located close to a class boundary, while the other is the data located in the outside of learned region (i.e., outlier). In order to ensure fast and accurate learning, the data selection mechanism should be introduced into a classifier model to find such essential data from a given chunk of data.

The first type of essential data are selected using LSH[4, 5, 6, 7, 8]. In RAN-LSH, whether a data exists in "well-trained region" or not is determined by calculating a hash value. If the margin flag in a matched entry is '1', it means a "well-trained region", and '0' otherwise. On the other hand, the second type of essential data are selected by the outlier detection, which relies on the output margin of the RAN-LSH classifier. Thus, the objective of this dissertation is to develop a model or system that can learned the stream data in high speed especially for cybersecurity application, so that the after-effect of cybercriminals attacks from both malicious spam email and DDoS attack can be avoided by discarding harmful email or data packets from reaching the user-end.

Another major issue in cybersecurity is to obtain a large number of cybersecurity dataset with a class label. It is usually difficult to obtain such data, because to collect these data from any organization or company, the researcher need to follow a strict guideline to preserve the confidentiality and privacy of the data. Besides that, it is also hard to assign a class label to each of the acquired data. To solve the first issue, the dataset of malicious spam email and DDoS backscatter attack in this study are collected from a server that gathered double-bounce email and packets from darknet, respectively. Since the data could not be sent back to the sender (i.e, the sender spoofed the email address or destination IP address), these data are quite suspicious and have a higher possibility to hold a malicious intention. Therefore, by using double bounce email and darknet packets, it is expected that the proportion of positive examples would be higher compare to conventional method. This give an advantage when the labeling process which is carried out based on rule-based from the pool of positive examples. The labeling for malicious spam email is carried out using SPIKE, whereas for darknet packets, the data are assigned a class label from analysis of traffic features.

5.1 Contributions of This Dissertation

In Chapter 2, a new incremental learning algorithm of RBFN is proposed called RAN-LSH to speed-up the learning of large-scale data sequences. Since the time to calculate the connection weight of RBFN is propotional to the number of training data, a huge amount of time is required to complete the incremental learning if many training data are used. Therefore, by removing similar training data, the number of data to be trained can be reduced and contributes to fast incremental learning. In order to search similar data, LSH partitioning is adopted to group similar data in the same bucket and choose data with margin flag equals to "0". In addition, by reusing the hash table from LSH method to choose near RBF bases, a faster processing time is obtained.

In Chapter 3, an autonomous malicious spam email detection system is proposed in which RAN-LSH is adopted as a classifier model and an outlier detection mechanism is implemented in RAN-LSH. The proposed malicious spam email detection system has desirable characteristics as an online cyber-security system, because the detection system can quickly adapt to new trends of malicious spam emails with reliable detection. These characteristics are realized by introducing incremental learning, outlier detection, and automatic malicious spam email analyzer called SPIKE in the proposed detection system. Since the maliciousness of most spam emails are judged by RAN-LSH, the number of times to execute SPIKE is very limited. Therefore, the incremental learning for new malicious spam emails is also conducted very quickly. This enables the proposed detection system to operate in real-time under a practical environments.

In Chapter 4, a fast DDoS backscatter detection system is proposed to detect DDoS attack from labeled and unlabeled packets using RAN-LSH classifier. Basically, it is not easy to detect DDoS backscatter for unlabeled packets. However, it has been shown in this study that the proposed detection system is capable to detect DDoS backscatter for not only labeled packets, but also unlabeled packets. To achieve the goal, the RAN-LSH classifier need to learn the general rule of

DDoS backscatter in the 80/TCP and 53/UDP packets. After that, the updated RAN-LSH classifier is used to detect both labeled and unlabeled packets. The detection system has been proven capable to detect the DDoS backscatter in a short time, which is shown by the experimental result, where the time to classify a data would require less than 0.01 (sec.).

5.2 Future Work

There still remain several open problems in the proposed RAN-LSH classifier. Although the learning time is reduced, the accuracy is also a little bit lower than the conventional margin-based model RAN-MRG. One of the reasons is that the hash functions of LSH are only generated in the initial learning and are not adapted incrementally. One way to solve this problem is that the proposed RAN-LSH can be extended not only by updating hash functions incrementally but also by increasing the number of hash functions. This can be attained by introducing Incremental PCA (IPCA)[47] into the proposed RAN-LSH. However, there is one problem for this. After updating/increasing hash functions, the hash table should also be updated properly without unexpected forgetting of the previous knowledge. If this problem is solved, it is expected that the idea of fast processing for large-scale stream data using LSH would work well. As another problem, the number of partition P should also be set properly in the LSH-based data selection. As mentioned in 2.3, P is determined only from the initial training data and it is fixed during the incremental learning. Since the data distributions are always changed over time, P should be adapted to the change in the data distributions.

As the incremental learning is carried out for a long time, it is expected that the number of RBF could also increased excessively, which will cause longer learning time. To overcome this problem, a pruning function for RBF bases is intended to be added in future. Besides that, an adaptive hash table to adapt to the changes of feature vectors from the recent BoW without forgetting the previous

knowledge is planned to be included in the proposed system. It is expected that the detection system would be more stable and robust to the new malicious spam email attacks.

An automatic parameter tuning is planned to be developed to achieve high detection rate. The parameters are RBF width σ, the number of partitions P, error threshold ε, distance threshold δ, accumulation ratio θ_a, tolerant distance θ_p, output margin θ_m, outlier threshold θ_o, occurance frequency θ_N and the highest network output θ_z.

The above issues are left as the future work.

Bibliography

1) Guha, S., Koudas, N., and Shim, K., "Data Streams and Histograms," *ACM Symposium on Theory of Computing*, pp. 471–475, 2001

2) Sun, N., and Guo, Y., "A Modified Incremental Learning Approach for Data Stream Classification," *Sixth International Conference on Internet Computing for Science and Engineering*, Henan, pp. 122–125, 2012

3) Melville, P., and Mooney, R. J., "Diverse Ensembles for Active Learning," *Proc. 21th International Conf. on Machine Learning*, Banff, CA, pp. 584–591, 2004

4) Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S., "Locality-Sensitive Hashing Scheme Based on p-stable Distributions," *Proceedings of Symposium on Computational Geometry (SoCG'04)*, pp. 253–262, 2004

5) Andoni, A., and Indyk, P., "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *Communications of the ACM*, Vol. 51, Issue 1, pp. 117–122, 2008

6) Gu, X., Zhang, Y., Zhang, L., Zhang, D., and Li, J., "An Improved Method of Locality Sensitive Hashing for Indexing Large-Scale and High-Dimensional Features," *Signal Processing*, Vol. 93, Issue 8, pp. 2244–2255, 2013

7) Lee, K. M., and Lee, K. M ., "Similar Pair Identification Using Locality-Sensitive Hashing Technique," *Proceedings of Joint 6th International Conference on Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS)*, pp. 2117–2119, 2012

8) Shen, H., Li, T., Li, Z., and Ching, F., "Locality Sensitive Hashing Based Searching Scheme for a Massive Database," *Proceedings of IEEE SoutheastCon'08*, pp. 123–128, 2008

9) Scheper, C., Cantor, S. and Maughan, D., "PREDICT: A Trusted Framework for Sharing Data for Cyber Security Research," *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGER'11)*, New York, NY, USA, pp. 105–106, 2011

10) The White House Cyberspace, "Cyberspace Policy Review: Assuring a Trusted and Resilient Information and Communications Infrastructure," pp. 1–38, 2009 http://www.whitehouse.gov/assets/documents/Cyberspace_Policy_Review_final.pdf

11) Haykin, S., "Neural Networks: A Comprehensive Foundation," Prentice Hall, 1999

12) Platt J., "A Resource-Allocating Network for Function Interpolation," *Neural Computation*, Vol. 3, Issue 2, pp. 213–225, 1991

13) Kobayashi, M., Zamani, A., Ozawa, S., and Abe, S., "Reducing Computational in Incremental Learning for Feedforward Neural Network with Long-Term Memory," *Proceedings International Joint Conference on Neural Networks*, pp. 1989–1994, 2001

14) Okamoto, K., Ozawa, S., and Abe, S., 'A Fast Incremental Learning Algorithm of RBF Networks with Long-Term Memory," *Proceedings of International Joint Conference on Neural Networks*, pp. 102–107, 2003

15) Ozawa, S., Tabuchi, T., Nakasaka, S., and Roy, A., "An Autonomous Incremental Learning Algorithm for Radial Basis Function Networks," *Journal of Intelligent Learning Systems and Applications*, Vol. 2, pp. 179–189, 2010

16) Slaney, M., and Casey, M., "Locality-Sensitive Hashing for Finding Nearest Neighbors [Lecture Notes]," *Signal Processing Magazine*, IEEE, Vol. 25, Issue 2, pp. 128–131, 2008

17) Ozawa, S., Toh, S. L., Abe, S., Pang, S., and Kasabov, N., "Incremental Learning of Feature Space and Classifier for Face Recognition," *Neural Networks*, Vol. 18, Issue 5-6, pp. 575–584, 2005

18) Ali, S. H. A., Ozawa, S., Nakazato, J., Ban, T., and Shimamura, J., "An Autonomous Online Malicious Spam Email Detection System Using Extended RBF Network," *Proc. Int. Joint Conf. on Neural Networks (IJCNN'2015)*, Killarney, Ireland, 2015

19) Alazab, M., Venkatraman, S., Watters, P. and Alazab, M., "Information Security Governance: The Art of Detecting Hidden Malware," *IT Security Governance Innovations: Theory and Research*, ed: IGI Global, pp. 293–315, 2013

20) Symantec Corporation, "Internet Security Thread Report (ISTR)", pp. 1–119, 2015

21) Karspersky Lab, "Spam and Phishing Statistics Report Q1-2014," 2014

22) Deshmukh, P., Shelar, M. and Kulkarni, N., "Detecting of Targeted Malicious Email," *2014 IEEE Global Conference on Wireless Computing and Networking (GCWCN)*, pp. 199–202, 2014

23) Alazab, M., Layton, R. ,Broadhurst, R. and Bouhours, B., "Malicious Spam Emails Developments and Authorship Attribution," *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, pp. 58–68, 2013

24) Song, J., Inoue, D., Eto, M., Kim, H. C., and Nakao, K., "An Empirical Study of Spam : Analyzing Spam Sending Systems and Malicious Web Servers," *2010 10th Annual International Symposium on Applications and the Internet*, pp. 257–260, 2010

25) Anderson, D. S., Fleizach, C., Savage, S., Voelker, G. M., "Spamscatter: Characterizing Internet Scam Hosting Infrastructure," *Proceedings of the USENIX Security Symposium*, Boston, 2007

26) Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I., "Spamming Botnets: Signatures and Characteristics," *ACM SIGCOMM Computer Communication Review*, Vol. 38, Issue 4, 2008

27) John, J. P., Moshchuk, A., Gribble, S. D., and Krishnamurthy, A., "Studying

Spamming Botnets Using Botlab," *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pp. 291–306, 2009

28) Chen, Y., Ma, X., and Wu, X., "DDoS Detection Algorithm Based on Preprocessing Network Traffic Predicted Method and Chaos Theory," *IEEE Communications Letters*, Vol. 17, Issue 5, pp. 1052–1054, 2013

29) CERT, "Overview of Attack Trends," http://www.cert.org/archive/pdf/attack_trends.pdf, Apr. 8, 2002

30) Li, L. and Lee, G., "DDoS Attack Detection and Wavelets," *Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN)*, pp. 421–427, 2003

31) Zhang, J., Qin, Z., Ou, L., Jiang, P., Liu, J. R., and Liu, A. L., "An Advanced Entropy-Based DDoS Detection Scheme," *20IO International Conference on Information, Networking and Automation (ICINA)*, pp. 68–71, 2010

32) Ali, S. H. A., Ozawa, S., Nakazato, J., Ban, T., and Shimamura, J., "An Online Malicious Spam Email Detection System Using Resource Allocating Network with Locality Sensitive Hashing," *Journal of Intelligent Learning Systems and Applications (JILSA)*, Vol. 7, pp. 42–57, 2015

33) Ali, S. H. A., Furutani, N., Ozawa, S., Nakazato, J., Ban, T., and Shimamura, J., "Distributed Denial of Service (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection," *Memoirs of Graduate Schools of Engineering and System Informatics, Kobe University*, Vol. 7, pp. 000-000, 2015

34) Attar, V., Sinha, P., and Wankhade, K., "A Fast and Light Classifier for Data Streams," *Evolving Systems*, Vol. 1, Issue 3, pp. 199–207, 2010

35) Polikar, R., Udpa, L., Udpa, S. S., and Honavar, V., "Learn++: An Incremental Learning Algorithms for Supervised Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 31, pp. 497–508, 2001

36) Cauwenberghs, G., and Poggio, T., "Incremental and Decremental Support

Vector Machine Learning," *Advances Neural Information Processing Systems (NIPS'2000)*, Vol. 13, 2000

37) Diehl, C. P., and Cauwenberghs, G., "SVM Incremental Learning, Adaptation and Optimization," *Proceedings of the 2003 International Joint Conference on Neural Networks (IJCNN'03)*, pp. 2685–2690, 2003

38) Spüler, M., Rosenstiel, W., Bogdan, M., "Adaptive SVM-Based Classification Increases Performance of a MEG-Based Brain-Computer Interface (BCI)," *Proceedings of the 22nd International Conference on Artificial Neural Networks*, pp. 669–676, 2012

39) Subramanian, K., Suresh, S., and Sundararajan, N., "A Metacognitive Neuro-Fuzzy Inference System (McFIS) for Sequential Classification Problems," *IEEE Transactions on Fuzzy Systems*, Vol. 21, Issue 6, pp. 1080–1095, 2013

40) Lin, H. T., Lin, C. J., and Weng, R. C., "A Note on Platt's Probabilistic Outputs for Support Vector Machines," *Technical Report, Department of Computer Science, National Taiwan University*, 2003

41) Wu, T. F., Lin, C. J., and Weng, R. C., "Probability Estimates for Multi-Class Classification by Pairwise Coupling," *Journal of Machine Learning Research*, Vol. 5, pp. 975–1005, 2004

42) Dan, R., and Kevin, S., "Margin-Bbased Active Learning for Structured Output Spaces," *Proceedings ECML'06*, pp. 413–424, 2006

43) Chellapilla, K., Mityagin, A., and Charles, D. X., "GigaHash: Scalable Minimal Perfect Hashing for Billions of URLs," *Proceedings of the 16th International Conference on World Wide Web*, 2007

44) Bache, K., and Lichman, M., "UCI Machine Learning Repository http://archive.ics.uci.edu/ml Accessed 01 January 2013.

45) Davies, D. L., and Bouldin, D. W., "A Cluster Separation Measure," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 1, Issue 2, pp. 224–

227, 1979

46) Hsu, C. W., and Lin, C. J., "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, Vol. 13, Issue 2, pp. 415–425, 2002

47) Ozawa, S., Pang, S., and Kasabov, N., "Incremental Learning of Chunk Data for On-line Pattern Classification Systems," *IEEE Trans. on Neural Networks*, Vol. 19, Issue 6, pp. 1061–1074, 2008

48) Vuong, T. P., and Gan, D., "A Targeted Malicious Email (TME) Attack Tool," *6th International Conference on Cybercrime, Forensics, Education and Training (CFET)*, Christ Church Canterbury, 2012

49) Nagarjuna, B. V. R. R., and Sujatha, V., "An Innovative Approach for Detecting Targeted Malicious E-mail," *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, Vol. 2, Issue 7, pp. 422–428, 2013

50) Symantec Corporation, "Internet Security Threat Report 2014," Vol. 19, pp. 1–98, 2014

51) Hurcombe, J., "Malicious Links: Spammers Change Malware Delivery Tactics," 2014 http://www.symantec.com/connect/blogs/malicious-links-spammers-change-malware-delivery-tactics

52) Amin, R. M., Ryan, J., and van Dorp, J., "Detecting Targeted Malicious Email Using Persistent Threat and Recipient Oriented Features," *IEEE Security and Privacy Magazine*, Vol. 99, pp. 1–12, 2011

53) Hadnagy, C., "Social Engineering: The Art of Human Hacking," *Wiley*, Indianapolis, 2011

54) Jungsuk, S., "Clustering and Feature Selection Methods for Analyzing Spam Based Attacks," *Journal of the National Institute of Information and Communications Technology*, Vol. 58, Issue 3/4, pp. 35–50, 2011

55) Criddle, L., "What are Bots, Botnets and Zombies?"

http://www.webroot.com/nz/en/home/resources/tips/pc-security/security-what-are-bots-botnets-and-zombies

56) Nazirova, S., "Survey on Spam Filtering Techniques," *Communications and Network*, Vol. 3, pp. 153–160, 2011

57) Ali, S. H. A., Fukase, K., and Ozawa, S., "A Neural Network Model for Large-Scale Stream Data Learning Using Locally Sensitive Hashing," *Neural Information Processing Lecture Notes in Computer Science*, pp. 369–376, 2013

58) Langley, P., "Selection of Relevant Features in Machine Learning," *Proceedings of the AAAI Fall Symposium on Relevance*, LA, pp. 140–144, 1994

59) Oyang Y. J., Hwang, S. C., Ou, Y. Y., Chen, C. Y., and Chen, Z. W., "Data Classification with Radial Basis Function Networks Based on a Novel Kernel Density Estimation Algorithm," *IEEE Trans Neural Networks*, Vol. 16, Issue 1, pp. 225–236, 2005

60) Dai, Y., Tada, S., Ban, T., Nakazato, J., Shimamura, J., and Ozawa, S., "Detecting Malicious Spam Mails: An Online Machine Learning Approach," *Neural Information Processing Lecture Notes in Computer Science*, Vol. 8836, pp. 365–372, 2014

61) Cortes, C. and Vapnik, V., "Support-Vector Networks," *Machine Learning*, Vol. 20, Issue 3, pp. 273–297, 1995

62) Brank, J., Grobelnik, M., Milič-frayling, N., and Mladenič, D., "Feature Selection Using Linear Support Vector Machines," *Proceedings of the 3rd Int. Conf. on Data Mining Methods and Databases for Engineering, Finance, and Other Fields*, pp. 84–89, 2002

63) Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gatford, M., "Okapi at TREC-3," *Proceedings of the Third NIST Text Retrieval Conference (TREC3)*, Washington, DC, pp. 109–126, 1996 NIST Special Publication 500-225.

64) Olson, P., "The Largest Cyber Attack in History Has Been Hitting Hong

Kong Sites," 2014 http://www.forbes.com/sites/parmyolson/2014/11/20/the-largest-cyber-attack-in-history-has-been-hitting-hong-kong-sites/

65) Lee, D., "Global Internet Slows After 'Biggest Attack in History'," 2013 http://www.bbc.com/news/technology-21954636,

66) Graham-Cumming, J., "The Wednesday Witching Hour: CloudFlare DoS Statistics," 2012 https://blog.cloudflare.com/the-wednesday-witching-hour-cloudflare-dos-st/

67) Jackson, D., "Understanding and Combining DDoS Attacks: A Threat Analysis," 2011 http://www.secureworks.com/assets/pdf-store/articles/Understanding_and_Combating_DDoS_Attacks.pdf

68) Xu, X., Wei, D., and Zhang, Y., "Improved Detection Approach for Distributed Denial of Service Attack Based on SVM," *Third Pacific-Asia Conference on Circuits, Communications and System (PACCS'11)*, pp. 1–3, 2011

69) Furutani, N., Ban, T., Nakazato, J., Shimamura, J., Kitazono, J., and Ozawa, S., "Detection od DDoS Backscatter Based on Traffic Features of Darknet TCP Packets," *Ninth Asia Joint Conference on Information Security (ASIA JCIS)*, pp. 39–43, 2014

70) Xu, X., Wei, D., and Zhang, Y., "Improved Detection Approach for Distributed Denial of Service Attack Based on SVM," *2011 Third Pacific-Asia Conference on Circuits, Communications and System (PACCS)*, pp. 1–3, 2011

71) Saied, A., Overill, R. E., and Radzik, T., "Artificial Neural Networks in the Detection of Known and Unknown DDoS Attacks: Proof-of-Concept," *Proceedings of PAAMS 2014 International Workshops*, Salamanca, Spain, pp. 309–320, 2014

72) H. Hishinuma, "Information Security by NICT and the Government of Japan," 2011 http://www.bic-trust.eu/files/2011/12/slides13.pdf

73) Broomhead, D. S., and Lowe, D., "Radial Basis Functions, Multi-variable Functional Interpolation and Adaptive Networks (Technical Report)," *Royal*

Signal and Radar Establishment (RSRE), Vol. 4148, pp. 1–39, 1988

74) Poggio, T. and Girosi, F, "Networks for Approximation and Learning," *IEEE Trans. on Neural Networks*, Vol. 78, Issue 9, pp. 1481–1497, 1990

Appendix A

Algorithms From Chapter 1

A.1 Algorithm 1: Modified RAN Learning

Algorithm 1 Modified RAN Learning

Require: Data $\boldsymbol{X}' = \{(\boldsymbol{x}_i, \boldsymbol{d}_i, \boldsymbol{h}(\boldsymbol{x}_i))\}_{i=1}^{N}$, RAN, hash table $\mathcal{H}$ and parameter θ_p.
Ensure: RAN.

for all $(\boldsymbol{x}_i, \boldsymbol{d}_i, \boldsymbol{h}(\boldsymbol{x}_i)) \in \boldsymbol{X}'$ **do**
  Calculate the outputs $\boldsymbol{z}$ for $\boldsymbol{x}_i$ by Eq. (1.2) and find the nearest center $\boldsymbol{c}^*$ to $\boldsymbol{x}_i$.
  Calculate the error: $E = \|\boldsymbol{z}(\boldsymbol{x}) - \boldsymbol{d}\|/K$
  if $E > \varepsilon$ & $\|\boldsymbol{x} - \boldsymbol{c}^*\| > \delta$ **then**
    Add an RBF unit (i.e., $J \leftarrow J + 1$) and set an RBF center, and the connection weights: $\boldsymbol{c}_J = \boldsymbol{x_i}$, and $\boldsymbol{w}_J = \boldsymbol{d}_i - \boldsymbol{z}$.
    Get an index set $\mathcal{C}$ of RBFs position in hash table as such $\boldsymbol{h}(\boldsymbol{c}_J)$ is the eth entry in the hash table $\mathcal{H}_{\{\mathcal{C}_J = e\}}$: $\boldsymbol{h}(\boldsymbol{c}_J) = \boldsymbol{h}(\bar{\boldsymbol{x}}_e)$.
  else
    Do *LSH-based RBF Selection* for training data $(\boldsymbol{x}_i, \boldsymbol{d}_i, \boldsymbol{h}(\boldsymbol{x}_i))$, an index set $\mathcal{C}$ of RBFs position in hash table and an index set $\mathcal{R}$ of selected RBFs.
    for all $\{(\boldsymbol{c}_j, \boldsymbol{d})\}_{j \in \mathcal{R}}$ **do**
      Calculate the outputs $\boldsymbol{\Phi}$ for selected RBF centers $\boldsymbol{c}_j$ $(j \in \mathcal{R})$ and a training data $\boldsymbol{x}_i$ by Eq. (1.1).
      Decompose $\boldsymbol{\Phi}$ using SVD and obtain the matrices $\boldsymbol{V}$, $\boldsymbol{H}$, $\boldsymbol{U}$.
      Update connection weights by Eq. (1.4)
    end for
    Calculate the outputs $\boldsymbol{z}$ for $\boldsymbol{x}_i$ by Eq. (1.2).
    Calculate the error: $E = \|\boldsymbol{z}(\boldsymbol{x}) - \boldsymbol{d}\|/K$
    if $E > \varepsilon$ **then**
      Add a hidden unit and the connection weights: $\boldsymbol{c}_J = \boldsymbol{x_i}$, and $\boldsymbol{w}_J = \boldsymbol{d}_i - \boldsymbol{z}$.
      Get an index set $\mathcal{C}$: $\mathcal{C}_J = e$.
    end if
  end if
end for

Appendix B

Algorithms From Chapter 2

B.1 Algorithm 2: RAN-LSH Learning Algorithm

Algorithm 2 RAN-LSH Learning Algorithm

Require: A set of N_d training data $\boldsymbol{X}' = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_d}$, parameters: accumulation ratio θ_a and output margin θ_m and tolerant distance θ_p.

1: Do *Initialization* using initial training data $\boldsymbol{X}_0 = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_0}$. Get proper parameters RBF width σ^2, number o partitions P, an initial hash table $\mathcal{H}$ and additional outlier detection parameters in Chapter 3: output margin for outlier detection θ_o and occurance frequency threshold θ_N.
2: Do *RAN Learning* to construct initial RAN.
3: **loop**
4: **Input:** A chunk of N_d training data $\boldsymbol{X}' = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=N_0+1}^{N_d}$.
5: Do *Update Hash Table* to get hash values $\boldsymbol{h}(\boldsymbol{x}_i)$.
6: Define $\hat{\boldsymbol{X}}' = \{(\boldsymbol{x}_i, \boldsymbol{d}_i, \boldsymbol{h}(\boldsymbol{x}_i))\}_{i=N_0+1}^{N_d}$.
7: Do *LSH-Based Data Selection* and get a set of selected data $\hat{\boldsymbol{X}}'$.
8: Do *Modified RAN Learning* for $\hat{\boldsymbol{X}}'$.
9: Do *Update Hash Table*.
10: **end loop**

B.2 Algorithm 3: Initialization

Algorithm 3 Initialization

Require: Initial data set $\boldsymbol{X}_0 = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_0}$, parameters θ_a, θ_m and θ_p.
Ensure: Hash table $\mathcal{H}$, parameters σ^2 and P.

1: Do the cross validation for $\boldsymbol{X}_0$ to determine the parameters: the RBF width σ^2 and the number of partitions P.
2: Do PCA for $\boldsymbol{X}_0$, and obtain l eigenvectors $\boldsymbol{U}_l$ such that the accumulation ratio is larger than θ_a.
3: **for** j=1 to l **do**
4: Calculate the jth projection values v_{ij} for all $(\boldsymbol{x}_i, \boldsymbol{d}_i) \in \boldsymbol{X}_0$ by Eq. (2.1).
5: Calculate projection vector $\boldsymbol{V}$ and for each $v \in \{v_1, \cdots, v_l\}$, find v^+ and v^-.
6: **end for**
7: Do *Update Hash Table* to create initial hash table.

B.3 Algorithm 4: Update Hash Table

Algorithm 4 Update Hash Table

Require: Data $\boldsymbol{X}' = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N}$ and hash table $\mathcal{H} = \{\boldsymbol{h}_e, \bar{\boldsymbol{x}}_e, F_e^M, F_e^{O\,a}, N_e\}_{e=1}^{N_\mathcal{H}}$.
Ensure: $\mathcal{H}$.
1: Calculate the hash values $\boldsymbol{h}(\boldsymbol{x}_i)$ for all $\boldsymbol{x}_i \in \boldsymbol{X}'$ by Equations (2.2) - (2.3).
2: Find $s(1, \cdots, S)$ subsets of unique hash values from $\{\boldsymbol{h}(\boldsymbol{x}_i)\}_{i=1}^{N}$ so that $\boldsymbol{h}_s \subset \{\boldsymbol{h}(\boldsymbol{x}_i)\}_{i=1}^{N}$.
3: Sort all $\boldsymbol{x}_i \in \boldsymbol{X}'$ into the corresponding subset $\boldsymbol{h}_s$ where $\boldsymbol{h}(\boldsymbol{x}_i) = \boldsymbol{h}_s$.
4: Define all data in each subset as $\tilde{\boldsymbol{X}}_s$.
5: **for** $s = 1$ to S **do**
6: **if** $\boldsymbol{h}_s$ is registered in $\mathcal{H}$ (i.e., $\boldsymbol{h}_s = \boldsymbol{h}_e$) & $F_e^M = 1$ **then**
7: Calculate the mean vector using Eq. $(2.4)^b$ or Eq. $(3.4)^c$ as a prototype $\bar{\boldsymbol{x}}_e$ where $n = |\tilde{\boldsymbol{X}}_s|$.
8: Set $N_e^c \leftarrow N_e + n$.
9: **print**
10: **else**
11: Calculate the mean vector using Eq. $(2.4)^b$ or Eq. $(3.4)^c$ as $\bar{\boldsymbol{x}}_e$.
12: Calculate the output margin Δz for $\bar{\boldsymbol{x}}_e$ by Eq. (1.5) and set the margin flag F_e^M by Eq. $(2.5)^d$ or Eq. $(4.1)^e$ and outlier flag F_e^O by Equations (3.3).
13: Set $N_e^c \leftarrow N_e + n$.
14: Register $\{\boldsymbol{h}_e, \bar{\boldsymbol{x}}_e, F_e^M, F_e^O, N_e\}_{e=N_\mathcal{H}+1}$ in $\mathcal{H}$.
15: Set $N_\mathcal{H} \leftarrow N_\mathcal{H} + 1$.
16: **end if**
17: **end for**

^{*a*} Note that outlier flag F^o is used only in malicious spam email detection system.

^{*b*} This equation is used in Chapter 2 and 4.

^{*c*} This equation is used in Chapter 3.

^{*d*} This equation is used in Chapter 2 and 3.

^{*e*} This equation is used in Chapter 4.

B.4 Algorithm 5: LSH-Based Data Selection

Algorithm 5 LSH-Based Data Selection

Require: $\hat{\boldsymbol{X}}' = \{(\boldsymbol{x}_i, \boldsymbol{d}_i, \boldsymbol{h}(\boldsymbol{x}_i))\}_{i=N_0+1}^{N_{d'}}$ and hash table $\mathcal{H}$.
Ensure: $\hat{\boldsymbol{X}}' = \{(\boldsymbol{x}_i, \boldsymbol{d}_i, \boldsymbol{h}(\boldsymbol{x}_i))\}_{i=1}^{N_{d'}}$.
1: **for** all $\boldsymbol{x}_i \in \hat{\boldsymbol{X}}'$ **do**
2: Find $\boldsymbol{h}(\boldsymbol{x}_i)$ in $\mathcal{H}$.
3: **if** $F_e^M = 1$ for a matched entry **then**
4: Remove $\boldsymbol{x}_i$ from $\hat{\boldsymbol{X}}'$ and $N_{d'} \leftarrow N_{d'} - 1$.
5: **end if**
6: **end for**

B.5 Algorithm 6: LSH-Based RBF Selection

Algorithm 6 LSH-Based RBF Selection

Require: Training data $(\boldsymbol{x}_i, \boldsymbol{d}_i, \boldsymbol{h}(\boldsymbol{x}_i))$, an index set $\mathcal{C}$ of RBFs position in hash table, an index set $\mathcal{R}$ of selected RBFs, hash table $\mathcal{H}$, and parameter θ_p.

Ensure: $\mathcal{R}$

1: **for** $j = 1$ to J **do**
2: Get hash value of kth entry from hash table $\mathcal{H}_{\{k=\mathcal{C}_j\}}$: $\boldsymbol{h}(\boldsymbol{c}_j) = \boldsymbol{h}(\hat{\boldsymbol{x}}_k)$ for *some* hash function.
3: Calculate LSH distance d_j^* between hash values of the retrieved RBF centers $\boldsymbol{h}(\boldsymbol{c}_j)$ and a given training data $\boldsymbol{h}(\boldsymbol{x}_i)$ using Eq. (2.6).
4: **if** $d_j^* \leq \theta_p$ **then**
5: Define a set $\mathcal{R}$ of selected RBF centers (i.e $\mathcal{R} \leftarrow j$).
6: **end if**
7: **end for**

Appendix C

Algorithms From Chapter 3

C.1 Algorithm 7: Malicious Spam Email Detection System

Algorithm 7 Malicious Spam Email Detection System

Require: Double bounce email.

Ensure: The class label of double bounce email.

1: Do *Pre-processing* to generate feature vectors of training data $\boldsymbol{X'} = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_{d'}}$ from double bounce email.
2: Do the cross-validation for N_0 initial training data $\boldsymbol{X}_0 = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_0}$ to determine the RBF width σ, the number of partitions P, error threshold ε and distance threshold δ. Determined the other four parameters empirically; accumulation ratio θ_a, output margin θ_m, network output θ_z and tolerant distance θ_p.
3: Do *RAN-LSH Learning* to construct initial RAN.
4: **loop**
5: **Input:** New double bounce email.
6: Do *Pre-processing* to generate feature vectors of test data $\boldsymbol{X''} = \{(\boldsymbol{x}_i\}_{i=1}^{N_{d''}}$ from double bounce email.
7: Test $\boldsymbol{X''}$ using the constructed RAN and obtain the class label of the test data $\boldsymbol{X''} = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}$.
8: Do *RAN-LSH Learning* to learn new training data $\boldsymbol{X'} \leftarrow \boldsymbol{X''}$.
9: **end loop**

C.2 Algorithm 8: Pre-processing of Malicious Spam Email Detection System

Algorithm 8 Pre-processing of Malicious Spam Email Detection

Require: Double bounce email.

Ensure: Malicious spam email data set: training data $\boldsymbol{X'} = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_{d'}}$ or test data $\boldsymbol{X''} = \{\boldsymbol{x}_i\}_{i=1}^{N_{d''}}$.

1: **for all** double bounce email **do**
2: Tokenization and lemmatization.
3: Selects N_f features using ℓ_1-SVM feature selection.
4: Feature representation using normalized TF-IDF.
5: **end for**

C.3 Algorithm 9: Outlier Detection

Algorithm 9 Outlier Detection

Require: Test data $\boldsymbol{X''} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, hash table $\mathcal{H}$ and SPIKE.
Ensure: Class label $\boldsymbol{d}_i$

```
loop
    Calculate the hash value h(x_i) by Eqs. (2.1) - (2.3).
    Find h(x_i) in H.
    if F_e^O = 0 then
        Assign the data as outliers.
        Get class label d_i using SPIKE.
    else
        Get class label d_i using the spam email detection system.
    end if
end loop
```

Appendix D

Algorithms From Chapter 4

D.1 Algorithm 10: DDoS Backscatter Detection

Algorithm 10 DDoS Backscatter Detection

Require: Darknet packets from 80/TCP and 53/UDP.

Ensure: The class label of given darknet packets.

1: Do *Pre-processing* to generate feature vectors of training data $\boldsymbol{X'} = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_{d'}}$ from darknet packets.
2: Do the cross-validation for N_0 initial training data $\boldsymbol{X}_0 = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_0}$ to determine the RBF width σ, the number of partitions P, error threshold ε and distance threshold δ. Determined the other four parameters empirically; accumulation ratio θ_a, output margin θ_m, network output θ_z and tolerant distance θ_p.
3: Do *RAN-LSH Learning* to construct initial RAN.
4: **loop**
5: **Input:** New darknet packets from other than 80/TCP and 53/UDP.
6: Do *Pre-processing* to generate feature vectors of test data $\boldsymbol{X''} = \{(\boldsymbol{x}_i\}_{i=1}^{N_{d''}}$ from darknet packets.
7: Test $\boldsymbol{X''}$ using the constructed RAN and obtain the class label of the test data $\boldsymbol{X''} = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}$.
8: Do *RAN-LSH Learning* to learn new training data $\boldsymbol{X'} \leftarrow \boldsymbol{X''}$.
9: **end loop**

D.2 Algorithm 11: Pre-processing of DDoS Backscatter Detection

Algorithm 11 Pre-processing of DDoS Backscatter Detection

Require: Darknet packets.
Ensure: DDoS backscatter data set: training data $\boldsymbol{X'} = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N_{d'}}$ or test data $\boldsymbol{X''} = \{\boldsymbol{x}_i\}_{i=1}^{N_{d''}}$.

for all darknet packets **do**
  Sort the darknet packet by the source IP addresses.
  if time interval between two sequence of packets $\geq$ one hour **then**
    Define a new data set $i \leftarrow i + 1$.
    Extract 17 features from the packets of the first one minute time-window.
    if the darknet packets are training data **then**
      Do *Labeling*
      Create ith feature vector $\{\boldsymbol{x}_i, \boldsymbol{d}_i\}$.
    else if the darknet packets are test data **then**
      Create ith feature vector $\{\boldsymbol{x}_i\}$.
    end if
  end if
end for

D.3 Algorithm 12: Labeling (for TCP Port 80 and UDP Port 53)

Algorithm 12 Labeling (for 80/TCP and 53/UDP)

Require: Input data containing 17 features.
Ensure: Class label $\boldsymbol{d}_i$.

loop
  if the packets are from 80/TCP **then**
    if (SYN=1 and ACK=1) $\|$ (RST=1 and ACK=1) $\|$ (RST=1) **then**
      Response flag $F^R = 1$.
    end if
  else if the packets are from 53/UDP **then**
    if the Query/Response flag QR=1 (domain name contains http, www, com) **then**
      Response flag $F^R = 1$.
    end if
  end if
  if Number of packets sent in one minute $\geq$ 40 $F^R = 1$ **then**
    Label the packet as DDoS backscatter $\boldsymbol{d}_i = \{10\}$.
  else
    Label the packet as DDoS non-backscatter $\boldsymbol{d}_i = \{01\}$.
  end if
end loop

Appendix E

Thesis Outlines Mind Mapping

E.1 Mind Mapping of Chapter 1

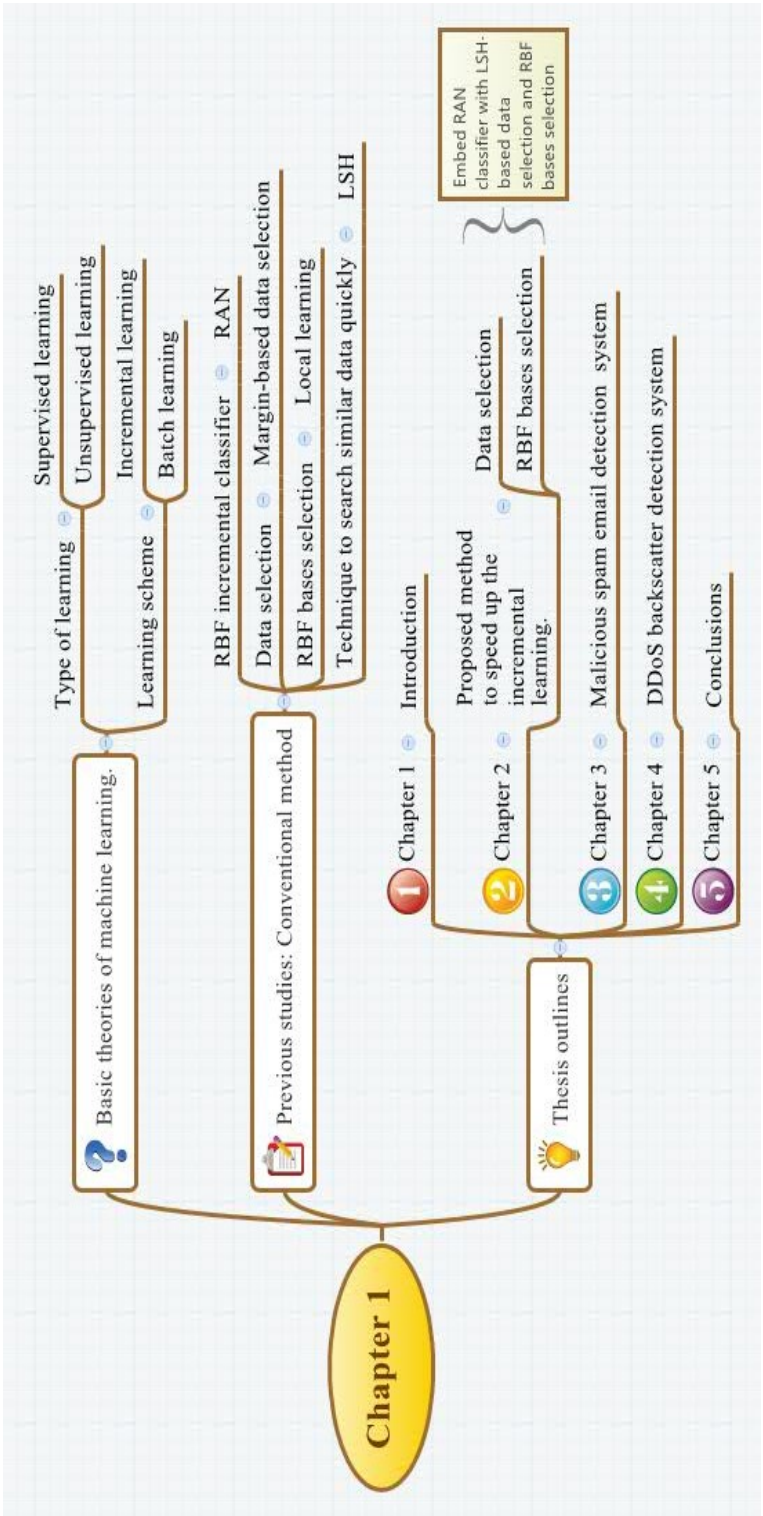


Figure E.1 Outline of Chapter 1.

E.2 Mind Mapping of Chapter 2

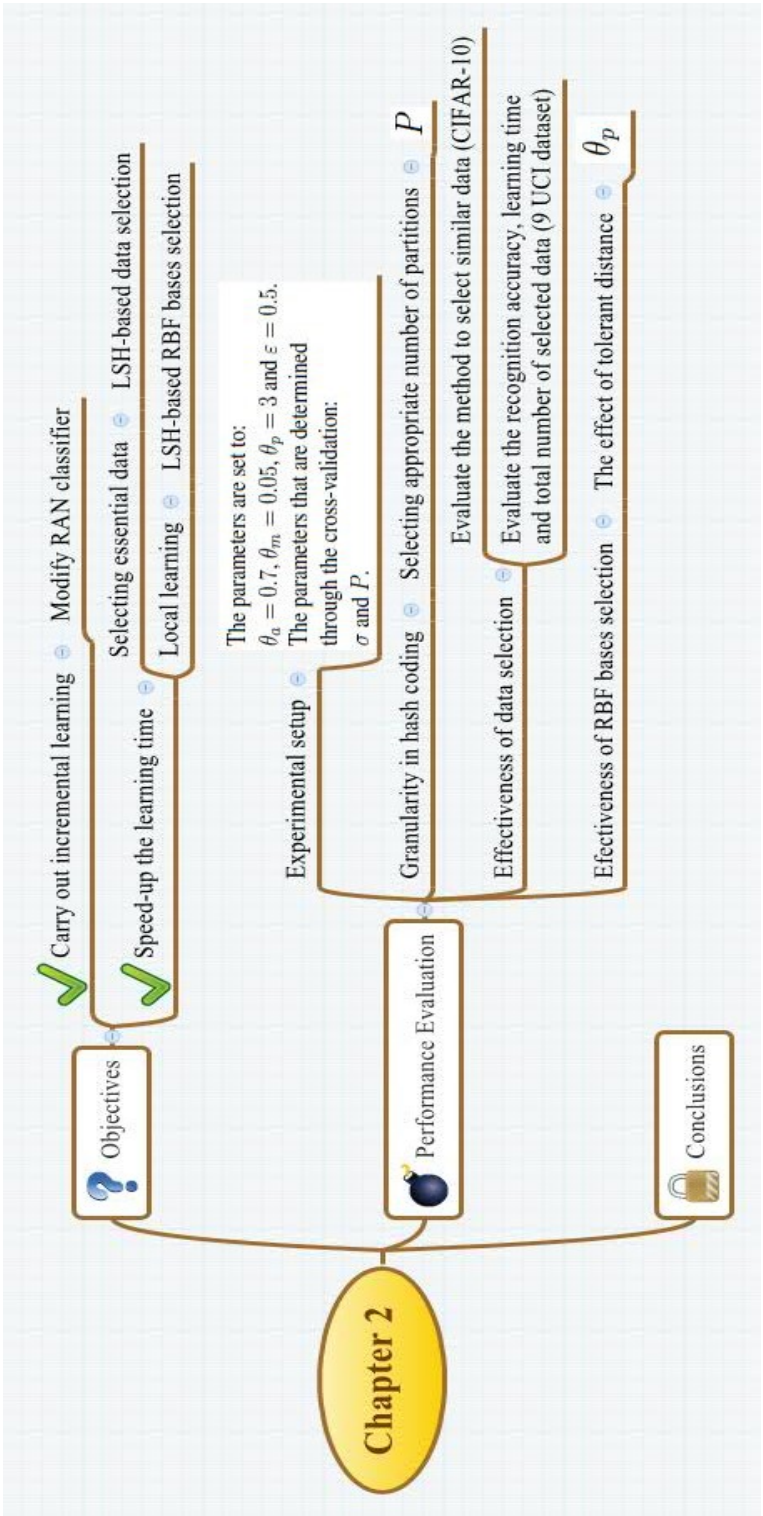


Figure E.2 Outline of Chapter 2.

E.3 Mind Mapping of Chapter 3

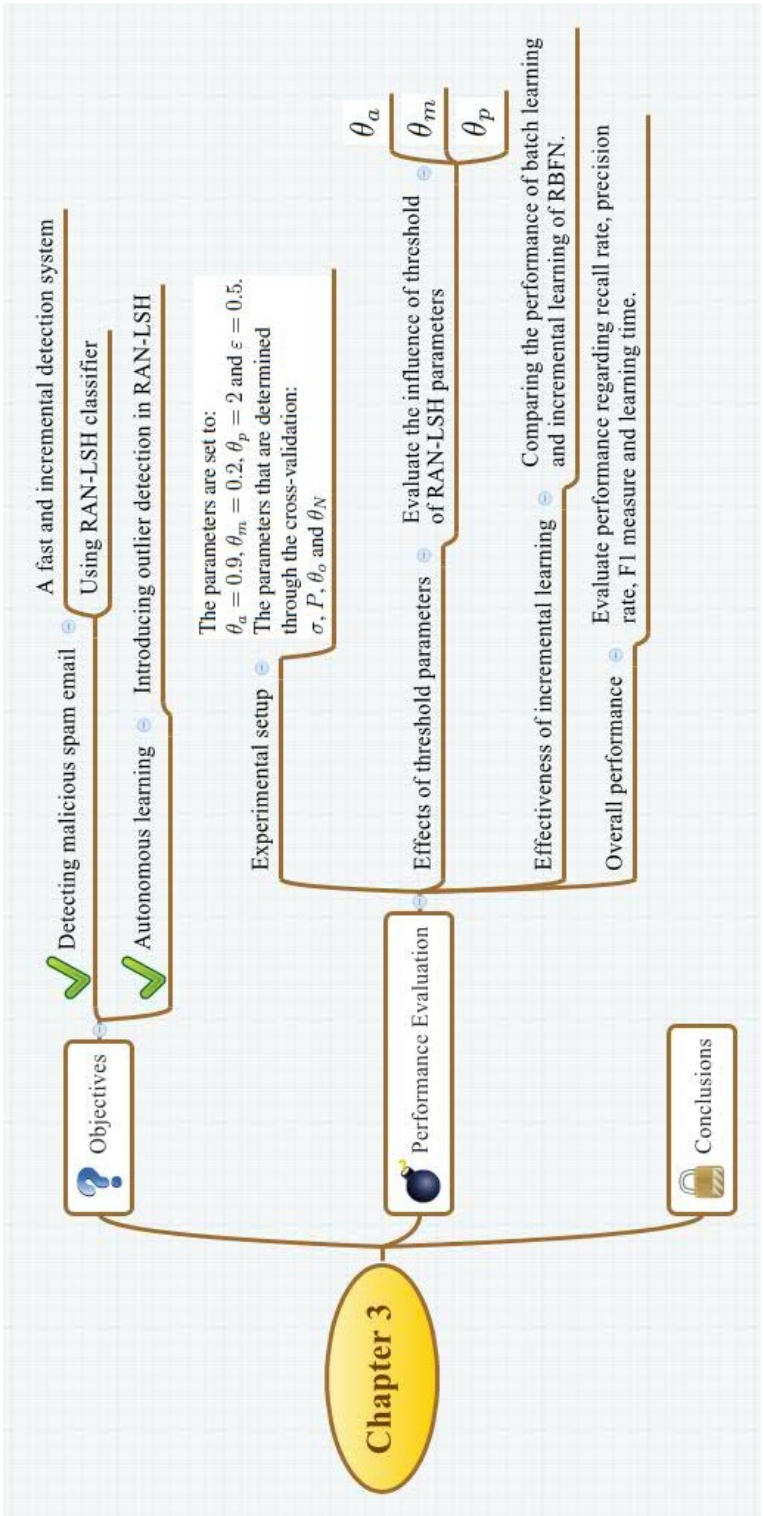


Figure E.3 Outline of Chapter 3.

E.4 Mind Mapping of Chapter 4

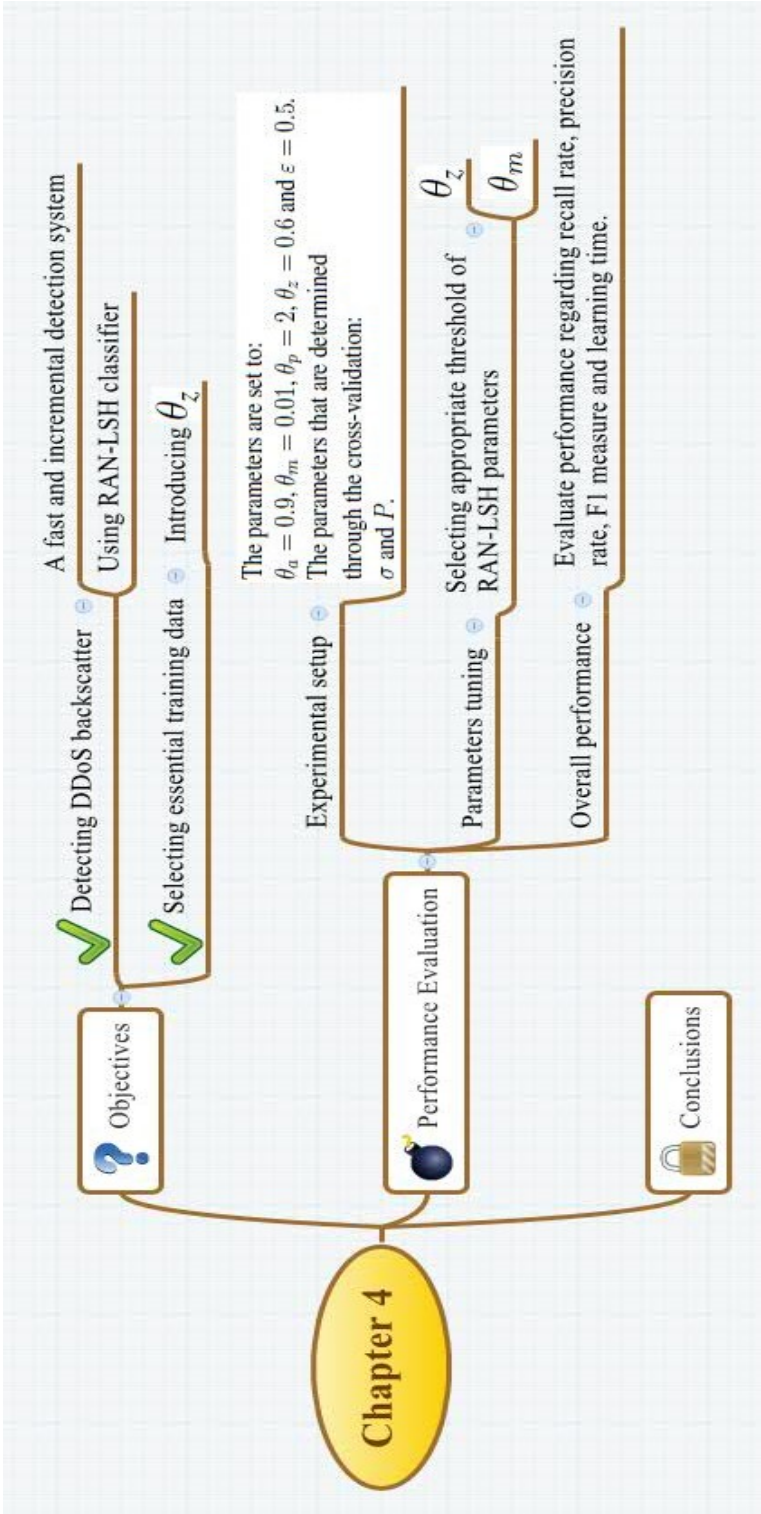


Figure E.4 Outline of Chapter 4.

List of Publications

Journal Papers

1) S. H. A. Ali, K. Fukase and S. Ozawa, "A Fast Online Learning Algorithm of Radial Basis Function Network with Locality Sensitive Hashing," Evolving Systems, Springer (in review) [Chapter 2]

2) S. H. A. Ali, S. Ozawa, J. Nakazato, T. Ban, and J. Shimamura, "An Online Malicious Spam Email Detection System Using Resource Allocating Network with Locality Sensitive Hashing," *Journal of Intelligent Learning Systems and Applications (JILSA)*, Vol. 7, pp.42-57, 2015 [Chapter 3]

3) S. H. A. Ali, N. Furutani, S. Ozawa, J. Nakazato, T. Ban, and J. Shimamura, "Distributed Denial of Service (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection," *Memoirs of Graduate Schools of Engineering and System Informatics, Kobe University*, Vol. 7, pp. 000-000, 2015 [Chapter 4]

International Conferences

1) S. H. A. Ali, K. Fukase and S. Ozawa, "A Neural Network Model for Large-Scale Stream Data Learning Using Locally Sensitive Hashing," *Neural Information Processing Lecture Notes in Computer Science, Springer (Proc. of 19th Int. Conf. on Neural Information Processing 2012 (ICONIP2013-Daegu, South Korea)*, pp. 369-376 (2013) [Chapter 2]

2) S. H. A. Ali, S. Ozawa, J. Nakazato, T. Ban, and J. Shimamura, "An Autonomous Online Malicious Spam Email Detection System Using Extended RBF Network," *Proc. Int. Joint Conf. on Neural Networks (IJCNN 2015), Killarney, Ireland*, 2015 [Chapter 3]

Domestic Conferences

1) K. Fukase, S. H. A. Ali and S. Ozawa, "A Study on Fast Learning for Large-Scale Stream Data," Proc. of the 57th Annual Conference of the Institute of Systems, Control and Information Engineers. SCI'13, 2013 [Chapter 2]

2) S. H. A. Ali and S. Ozawa, "A Radial Basis Function Network with Locality Sensitive Hashing for Large-Scale Stream Data," Proc. of the SICE System and Information Division Annual Conference, SSI'13, 2013 [Chapter 2]

3) S. H. A. Ali, K. Fukase and S. Ozawa, "A Neural Network Model for Incremental Learning of Large-Scale Stream Data," Proc. of the 58th Annual Conference of the Institute of Systems, Control, and Information Engineers (ISCIE), SCI'14, 2014 [Chapter 2]

Doctoral Dissertation, Kobe University
"Incremental Learning for Large-Scale Stream Data and its Application to Cybersecurity", 121 pages
Submitted on July, 10, 2015
The date of publication is printed in cover of repository version published in Kobe University Repository Kernel.

© Siti Hajar Aminah Binti Ali
All Right Reserved, 2015