Malware Motif Identification using Bio-inspired Data Mining

Yi Chen

A thesis submitted to

Auckland University of Technology

In partial Fulfilment of the requirements for the degree of

Master of Computer and Information Sciences (MCIS)

School of Computing and Mathematical Sciences

Auckland, New Zealand

2013

Declaration

..... [illegible] ....

Signature

Acknowledgements

I would like to thank all the people who have helped, supported and encouraged me during my masters study.

I express my sincerest thanks to my supervisors, Prof. Ajit Narayanan, Dr. Paul S. Pang and Dr. Ban Tao who have given me great assistance, support and guidance. Prof. Ajit Narayanan has always provided me with help and guidance so I could overcome my difficulties during my research. As a Master student at Auckland University of Technology, I feel so grateful to have undertaken my research under the supervision of Prof. Ajit Narayanan.

Four papers have been published from the research presented in this thesis.

1. Chen, Y., Narayanan, A., Pang, S. and Tao, B. Malicious. Malicious software detection using multiple sequence alignment and data mining.. *Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications*. Fukuoka, Japan, March 2012. (Section 8.3.1 Experiment I, case 1)
2. Chen, Y., Narayanan, A., Pang, S. and Tao, B. Malicious. Multiple sequence alignment and artificial neural networks for malicious software detection. *Proceedings of 8th IEEE Conference on Natural Computation (ICNC'12)*, Chonqing, China, May, 2012. (Section 8.3.2 Experiment I, case 2)
3. Narayanan, A., Chen, Y., Pang, S. and Tao, B. The effects of different representations on malware motif identification. *Proceedings of 8th International Conference on Computational Intelligence and Security*, Guangzhou, China, November, 2012. (Section 8.4.2 Experiment II, case 6)
4. Narayanan, A., Chen, Y., Pang, S. and Tao, B. The effects of different representations on static structure analysis of computer malware signatures. *The Scientific World Journal, 2013*

I would also like to express thanks to Dr. Ban Tao at the National Institute of Communication and Technology (NICT), Japan, for providing the original idea of this project.

I would also like to acknowledge Auckland University of Technology for providing me with such a good study environment and facilities.

And last, I would like to express my friend and gratitude to my family for their understanding, support and encouragement during my research study project at AUT.

Yi Chen
Auckland
24-Jan-13

Abstract

The application of data mining techniques into biological data is well established. The aim of this thesis is to explore the effects of giving amino acid representation to problematic machine learning data and to evaluate the benefits of supplementing traditional data mining techniques with bioinformatics tools, techniques and databases. The focus of the research is on methods for identifying patterns in computer malware signatures typically used in current anti-viral software. In total, 60 computer viruses and 60 worm signatures were converted into amino acid representations and then aligned to produce fixed length sequences as input to data mining techniques for classification and prediction. Standard protein databases and modellers were also used to give a biological interpretation, and to find biological analogues of the polypeptide representations of the malware signatures. Protein modelling of the consensuses produced through sequence alignment and meta-signatures extracted from data mining provides novel ways of looking at malware signatures and their possible structure and function. However, the results varied by the method of biological representation used and further work is needed to determine the advantages and disadvantages of different methods for representing data as artificial polypeptide sequences.

Keywords: malware; sequence alignment; viral signatures, ClustalW, T-Coffee

Table of Contents

List of Tables

List of Figures

Glossary

A

Artificial neural network (ANNs) is computational models inspired by animal central nervous systems (in particular the brain) that are capable of machine learning and pattern recognition.

Anti-Viral Software (AVS) is software used to prevent, detect and remove malware (of all descriptions), such as: computer viruses, malicious BHOs, hijackers, ransomware, keyloggers, backdoors, rootkits, trojan horses, worms, malicious LSPs, dialers, fraudtools, adware and spyware.

C

ClustalW with Blosum(CB) is a sequence alignment, produced by ClustalW using Blosum matrix

ClustalW with Gonnet(CG) is a sequence alignment, produced by ClustalW uisng Gonnet matrix

ClustalW with Identity(CI) is a sequence alignment, produced by ClustalW using Identity

H

Hidden Markov Models(HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states.

I

Internet Relay Chat(IRC) is a protocol for live interactive Internet text messaging (chat) or synchronous conferencing

J

J48 an open source Java implementation of the C4.5 decision tree algorithm

L

LAD Tree Logical Analysis of Data is the method for classification proposed in optimization literature

M

Machine Learning(ML), a branch of artificial intelligence, concerns the construction and study of systems that can learn from data.

Multi-Layer Perceptron(MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs

Motif3D is a simple wireframe protein structure viewer that has been designed specifically for use with the PRINTS database.

N

Naive Bayes is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions

O

OneR short for "One Rule", is a simple, yet accurate, classification algorithm that generates one rule for each predictor in the data, then selects the rule with the smallest total error as its "one rule".

P

Prism aim to reduce modular classification rules directly from the training set

Protein Data Bank(PDB) is a repository for the three-dimensional structural data of large biological molecules, such as proteins and nucleic acids

S

Scientific Research Methodology (SRM) is a body of techniques for investigating phenomena, acquiring new knowledge, or correcting and integrating previous knowledge

Sum of Squared Error(SSE) is a measure of the discrepancy between the data and an estimation model

T

T-Coffee Blosum(TB) is a multiple sequence alignment software using a progressive approach and Blosum matrix

W

Waikato Environment for Knowledge Analysis(WEKA) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand

Chapter 1 INTRODUCTION

1.1 BACKGROUND

Malware is the generic term given to any program or code intended to cause disruption or gain access to unauthorized information and resources. It only became wide-spread when PCs were connected to company intranets and the Internet in the mid-1990s. Such is the growth of malware now that Symantec reported over 5.5 billion malware attacks in 2011, 81% increase from 2010 (Symantic Internet Security Threat Report, 2011).

A computer virus requires some actions to be undertaken before it is active led, such as being attached to a program by the user, while a worm can propagate by itself. The aim of virus and worm writers is usually to cause some damage to computer systems, hence the term ‘malware’. The traditional method for dealing with viruses and worms (two of the most common types of malware) is to use anti-viral software (AVS) that looks for ‘signatures’, or patterns of bytes that appear in network packets.

Malware writers adopt a variety of sophisticated techniques for avoiding detection, such as self-modification (e.g. viral code is modified each time it infects a system) and metamorphosis (e.g. viral code is totally re-written each time it infects a system). By the time the new variants are identified and signatures are released, the infection may already have reached epidemic proportions (Strickland, 2011). One of the problems in applying automatic data mining techniques directly to malware code for identifying signatures is the variable length of the code. Most data mining and other machine learning techniques assume fixed length sequences, with a column representing measurements of the same variable across many samples (Xinguang, Miyi, Chunlai & Xin., 2009). There is surprisingly little work reporting on the application of machine learning techniques to malware signature detection, mainly due, one assumes, to the problem of dealing with variable length malware code to identify the signature of the virus or worm in question. Instead, the existing few work focus on anomalous behaviour detection (e.g. Rieck, Hols, Willems, Düssel & Laskov, 2008; Singhal & Raul, 2012) rather than on coding or signature analysis. There is a need for a greater understanding of signatures and how they can be generated and used more effectively to deal with the ever-growing threat of malware. Malware signature data mining is the focus of this thesis.

Sequence analysis is used in biology to understand the relationship between two or more sequences (multiple sequence alignment) of genetic information, DNA or amino acids (Mout, 2001). Databases of genetic information are processed by string alignment algorithms to better understand the relationship between species and also to determine the location of specific genes. In particular, sequence analysis and alignment can be used to identify conserved regions in biological data that identify common genes and shared ancestry as well as to identify drugs that will be applicable to more than one species. One advantage of the alignment methods is that biological sequences with variable length can be converted into fixed length sequences through appropriate insertion and deletion techniques. Powerful data mining algorithms that assume fixed length sequences or patterns can then be applied to identify critical features that help to determine whether a sequence is malware or not.

1.2 AIM OF THESIS

The aim of this thesis is to explore whether sequence alignment techniques currently used for identifying conserved regions in biological sequences can also be used for identifying malware signatures for effective identification and, ultimately for the removal of malware, where such malware is in the form of code with variable length. The use of such alignment techniques depends on representing non-biological data (malware signature / malware motif) as biological data, specifically, in amino acid form, if the full potential of sequence alignment is to be realized. The overall research question being addressed in this thesis is therefore whether non-biological data being represented as amino acid sequences confers any benefit over standard data mining.

This research question raises the issue of whether finding biological analogues of malware signature represented as biological sequences and aligned with bioinformatics multiple sequence alignment tools can shed new light on problems typically identified as belonging to classical machine learning and data mining. The second research question addressed in this thesis is whether the application of sequence alignment to malware signatures compromises or adds value to data mining. If substitution matrices are used during alignment, it makes sense to evaluate their effects in terms of mapping against biological analogues. That is, biological databases continue to expand at a fast rate, with fully mapped protein structures being added daily to the Protein Data Base (PDB) and Prosite. A wealth of scientific literature supports the proposed functions, structures and roles of nearly 90,000 protein structures. If benefits of data mining are found by using sequence alignment, the third research question asked in this thesis is whether the bio-informatics techniques for random sequencing value to data mining.

1.3 OVERVIEW OF RESEARCH METHOD

This thesis contains two sets of systems and methods which contribute to two sets of results. In the first set of systems and methods, standard alignment methods and machine learning techniques are applied to data represented as bio-sequences. Standard bioinformatics protein database searching and protein modeling techniques are applied in the second set. The combined results will lead to a discussion of the advantages and disadvantages of treating machine learning data as amino acid sequences.

In summary, the research issues being addressed in this thesis are: (a) the benefits and disadvantages of representing problematic data as bio-sequences followed by alignment and machine learning; and (b) the interpretability of the results. Our research methods can be summarized is as follows:

0. Represent two classes of data (virus and worm signatures) in amino acids;

a. For each class, align the sequences using an appropriate method and substitution matrix (result is variable length samples in each class);

b. Represent the gaps with an amino acid;

c. Align all samples of all classes together (results in longer sequences with fixed length);

d. Repeat (a) – (c) but with parameters changes for different experiments;

e. Apply machine learning (ML) techniques and compare with machine learning results using the original and unaligned data;

f. Evaluate the datasets with the best alignment and representation and extract rules;

g. Create random sequences;

h. Represent two classes of data (30 random sequences in each class) in amino acids;

i. Repeat (a) – (e) working on these random sequences;

j. Evaluate the datasets with the accuracies and extract rules;

k. Check the sequences against naturally occurring proteins.

More details are presented in chapters 5 and 6. Chapter 5 deals with steps 0-j and Chapter 6 deals with step k. Existing detection techniques and problems associated with such techniques will be introduced in Chapter 3. The research methodology to be adopted in our experiments will be described in Chapter 4. Chapter 5 will demonstrate the benefits of representing problematic data as bio-sequences followed by alignment and machine learning and the interpretability of the results. In Chapter 6, the motif of malware sequences will be compared with natural protein structures. The Chapter 7 will conclude the results and further work.

Chapter 2 Computer Malware Review

2.0 INTRODUCTION

Malware (the generic term given to any program or code intended to cause disruption or gain access to unauthorized information and resources) did not become prevalent until PCs were connected to company intranets and the Internet in the middle of 1990s. Some of the most famous malware has now entered computing folklore, including the Melissa virus (a Word macro spreading through emails and infecting about a quarter of million new computers a day in 1999), Code Red (a worm exploiting weaknesses in Windows 2000 and Windows NT that allowed remote access to a user's computer in 2001), SQL Slammer/Sapphire (a web server virus that caused more than $1b worth in damages among American banks and airline services in 2003), MyDoom (a backdoor virus that caused denial of service attacks and estimated to have infected 1 in 12 email messages in 2004) and Storm Worm (a 2006 backdoor virus that turned computers into bots or zombies under the control of a remote bot-herder) (Strickland, 2011). Such is the growth of malware that Symantec reported over 3 billion malware attacks in 2010, with 93% increase in web attacks (Symantec Internet Security Threat Report, 2011).

The traditional method for dealing with viruses and worms is to use anti-viral software (AVS) that looks for 'signatures', or patterns of bytes, in executable code. For signature-based AVS to work, a library of known signatures must be stored and maintained as new viruses are found and analysed. Most AVS systems offer hourly, daily and weekly updates. Malware writers adopt a variety of sophisticated techniques for avoiding detection, such as self-modification (e.g. viral code is modified each time it infects a system) and metamorphosis (e.g. viral code is totally re-written each time it infects a system) (Szor and Ferrie, 2011). By the time the new variants are identified and signatures are released, the infection may already have reached epidemic proportions. There is an urgent need to explore novel methods resident within a host computer that can quickly identify possible new variants of viruses for which there is no stored viral signature before they can do any harm.

2.1 TYPES OF MALWARE

There are many kinds of malware, all of which can cause different types of damages

2.1.1 COMPUTER VIRUS

A computer virus is a computer program that can replicate itself and spread from one computer to another without user’s authorization (Dmitry and Solomon, 1995). It tends to affect the normal operation of an infected computer. For example, boot virus is the earliest computer virus found on a PC. It mainly infects the boot sector of the floppy disk and hard disk boot sector or master boot record (Landesman, 2010). Macro virus is a kind of computer viruses which storage in the document or template files. Once the document is open, the macro will be executed. The macro virus is activated and transferred to the computer, and resides in the Normal template. Later, all auto-saved documents are infected by this macro virus and when another user opens an infected document, the macro virus will be transferred to his or her computer (Microsoft, 2006). Script virus is usually written in JavaScript code malicious code. It will modify the information of Internet explorer, registration and so on. Boot viruses, macro viruses and script virus as use as a transmission mechanism similar to biological viruses and biological virus when they are (Mount, 2001).

2.1.2 WORMS

A worm is a common computer virus that replicates itself in order to spread to other computers. Unlike a computer virus, it does not need to attach itself to an existing program (Wikipedia, 2012). It uses network transmission mechanism of replication and transmission through the Internet, email, U-disk, mobile hard drives and other removable storage devices. Worms spread primarily to exploit system vulnerabilities through the network, email and other means of communication. Since a worm spreads using a variety of ways, its propagation speed can be very high. After infecting a computer, a worm can get access to other computer IP addresses and then send a copy of itself to these computers (Jiangmin anti-virus warning center, 2007). Worms also use stored address of mail clients in the computer’s address book to spread. A worm not only takes up memory resources but also affects other functions of the computer (Strickland, 2011).

2.1.3 SPYWARE

Spyware is a way to collect information about software users without the knowledge of the user. It can undermine user privacy and security of material. Spyware typically collects users and disseminates the user’s personal or sensitive information (Schuster, 2005).

2.1.4 ADWARE

Adware is the act of downloading and/or installing of advertising material without the user’s permission (Tulloch, In Koch & Haynes., 2003). The installation of advertising software often causes the system to run slowly, often resulting in system anomalies. Some adware and spyware may be integrated, such as key-loggers.

2.1.5 SCAREWARE

Internet Security bloggers use the term “Scareware” to describe software that produces frivolous and alarming or threatening notices that the user’s system has been attacked and suggests the installation of fake antivirus software to remove it. In fact, the software is non-functional or even malware itself (Leydon, 2009).

2.1.6 CRIMEWARE

Crime software is secretly installed on the computer for malicious software (Jakobsson, and Ramzan, 2008). Most of the crimeware are in fact Trojan horse software. These include a variety of Trojans with different functions. For example, Trojans are used to record the user’s keyboard (key-loggers), take screen shots of online banking sites and some download other malicious codes. Also, some Trojans allow hackers remote access to infected systems. However, they all share a common goal, which is to steal confidential information like passwords and personal information. Using such stolen information, cyber criminals can steal the user’s money for instance.

2.1.7 ROOTKITS

The NSA Glossary of Terms Used in Security and Intrusion Detection define rootkits as a hacker security tool that captures passwords and message traffic to and from a computer. It also means a collection of tools that allows a hacker to provide a backdoor into a system, collect information on other systems on the network, mask the fact that the system is compromised, and much more (Butler and Sparks, 2005). Rootkits are a classic example of Trojan Horse software. Rootkits are available for a wide range of operating systems.

2.1.8 BOTNET

The concept of Botnet has several components. "Bot" is short for robot and is a program control function that is hidden to achieve malicious objectives. A "zombie computer" is a computer connected to the Internet that has been compromised by a bot. A "botnet" is a collection of internet-connected programs or bots that communicate with each other to perform malicious tasks. "Control Server" refers to the control and communications to a central server based on Internet Relay Chat (IRC) protocol to control the botnet. Bots are also used to recruit other computers to the botnet so that they also become zombie computers. The most important feature of botnets is to send spam and/or participate in distributed denial of service (DDoS) attacks. Botnets have become wide spread (Ramneek, 2003).

2.2 FORMS OF MALWARE

Besides having different functions as discussed above, malware also comes in different forms.

2.2.1 ENCRYPTED MALWARE

One technique commonly used by malware writers is to ensure that large portions of the malware is encrypted, except for a small segment of code which is the key to decrypt the virus when it is executed (Strickland, 2011). Although the key features of malware may never change, the key used to encrypt and decrypt the malware will be different for every generation of the malware. In this way, when the malware is stored on the disk, it will always be encrypted. The only time the unencrypted form is visible would be when the malware is being executed.

Detection of such malware is still possible without having to decrypt the actual malware body. In most cases the malware code patterns of decryptors are sufficient for detection (Szor and Ferrie, 2011).

2.2.2 POLYMORPHIC MALWARE

Polymorphic Malware is encrypted malware (Parikka and Lang, 2007), ‘improved’ by changing the decryption header and the encrypted code. In a polymorphic virus the encrypted code is first decrypted and then executed (i.e. a copy is made). The encryptor and decryptor are mutated with each copy.

2.2.3 THE LATEST DEVELOPMENT

McAfee Threats Reported (May, 2012) notes that the first quarter of 2012 exhibited an increase in malware across all platforms. The report showed that in the first quarter, PC malware reached its highest levels in four years, as well as a steep increase in malware targeting the Android platform. Mac malware was also on the rise, indicating that total malware could reach 100 million within the year, though they had only detected 8 million new malware samples in the first quarter. This shows authors of malware are continuing their unrelenting development of new malware and computers are still unable to learn how to deal with malware without updates.

2.3 CONCLUSION

From all the different types of malware motifs currently known we select motifs of computer viruses and worms as the domain of our study. First, they are the largest and oldest groups found in the malware world. Second, computer viruses and worms have replication features, where a computer virus replicates by being attached to another program in order to spread to other computers but a worm does not need to be attached to an existing program to spread.

Chapter 3 Malware Detections

3.0 INTRODUCTION

As malware attacks become more frequent, attention has begun to shift from viruses and spyware protection to malware protection with programs developed to specifically combat them. By the end of 2011, McAfee Labs collected more than 75 million malware samples. This increase has resulted in 83 million pieces of malware samples by the end of the first quarter. United States currently houses the largest number of botnet control servers, where an average of 9,000 new malicious websites is recorded per day (McAfee, 2012).

In the previous chapter we introduced different types of malware and their characteristics. This chapter contains a review of previous studies on ASCII signature-based detection techniques used in dictionary-based AVS. In this review, behavioral detection techniques are investigated and traditional detection and behavioral detection methods are summarized. The weaknesses of both existing detection techniques are also identified and described, pointing the way to a new approach.

3.1 TRADITIONAL (DICTIONARY-BASED) ASCII SIGNATURE-BASED MALWARE DETECTION

ASCII is a character-encoding scheme based on the ordering of the English alphabet and other unreadable characters. Not only can text be presented as ASCII codes, but all digital data stored in computers, transmitted over computer networks, or manipulated by other digital devices can be also encoded. These can be treated as ASCII codes where the data is divided into 8-bit units, i.e. the so called digital bytes (Kumar, 2007).

The traditional method for malware detection is ASCII code analysis. It is also known as signature -based detection, which is the earliest technique used to defend against malware and still remains at the core of current antivirus software. These detection techniques scan objects such as programs or files and suspicious patterns are compared with the patterns of known malware signatures stored in a database (Landesman 2010). Such techniques are also called appearance detection or syntactic marker detection because of their reliance on patterns of code.

The problem of traditional detection

ASCII signature based analysis is known to be very effective for known viruses, in which signatures are acquired by analysing sample of infected programs. However, the increasing number of malware variants has resulted in the inability of commercial antivirus software to detect slight modifications to the original malware. Moreover, the cost in time taken by these techniques is often very high even for very simple malware, which makes them unsuitable for real time detection. Signature generation is typically a manual process requiring extensive code analysis. Moreover, there is an increasing tendency for malware writers to take advantage of obfuscation transformations for analysis through reverse engineering (Sharif, 2009).

3.2 BEHAVIOURAL DETECTION OF MALWARE

Behavioral detection differs from signature-based detection in that it identifies malware through anomalous actions performed by the malware rather than syntactic patterns of code. Consequently, a whole class of malware can be identified through semantic interpretation rather than a single piece of malware through a signature.

The problem of behavioral detection

Though behavior detection provides undeniable advantages for operational use, it uses a behavioral structure which identifies a whole class of malware. Unfortunately, current behavioral detection techniques return high false negative rates due to incomplete behavior structures or missing data (Tang and Chen, 2007).

3.3 SEQUENCE ANALYSIS ALIGNMENT

Sequence analysis is used in biology to understand the relationship between two or more sequences (multiple sequence alignment) of genetic information, such as DNA or amino acids (Mount, 2001).

There are databases of genetic information which are processed by string alignment algorithms to better understand the relationship between species and to also determine the location of specific genes (DNA sequencing). In particular, sequence analysis and alignment can be used to identify conserved regions in biological data that identify common genes and shared ancestry as well as to identify drugs that will be applicable to more than one species. The purpose of protein sequencing, on the other hand, is primarily to identify the functions of a sequence of amino acids and to compare functionality across different species (protein sequencing). As proteins are involved in virtually all cell functions, each protein within the body has a specific function. Some proteins are involved in structural support, while others are involved in bodily movement, or in defence against germs.

For example, hormonal proteins (Bailey, 2012) are messenger proteins which help to coordinate certain bodily activities. Examples include insulin, oxytocin, and somatotropin. Insulin regulates glucose metabolism by controlling the blood-sugar concentration. Oxytocin (fig. 3.1) stimulates contractions in females during childbirth. Somatotropin is a growth hormone that stimulates protein production in muscle cells.

Figure 3.1: Oxytocin (ball-and-stick) bound to its carrier protein neurophysin (ribbons) (Rose, Wu, Hsiao, Breslow and Wang,1996)

One particular DNA sequence can code a number of different proteins, depending on a number of aspects, such as alternative mRNA splicing and changes to proto-proteins (i.e. raw sequences of amino acids) through folding and cellular location of the proteins (Bailey, 2012).

In biology, conserved sequences (Fig, 3.2, Wikipedia, 2012) are sequences of nucleotides in genetic material or of amino acids in a polypeptide chain that has changed slightly or not at all during an evolutionary period of time. A region of an aligned track is considered to be a ‘conserved region’. It is widely believed that mutation in a “highly conserved” region leads to a non-viable life form, or a form that is eliminated through natural selection.

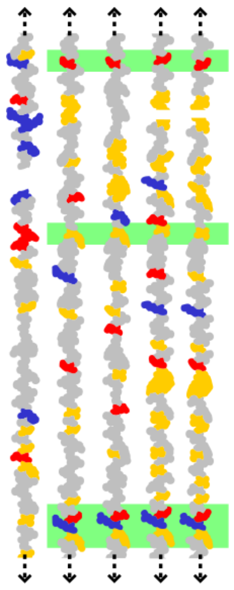

Figure 3.2: Residues conserved among various G protein coupled receptors are highlighted in green (Wikimedia, 2012)

One advantageous side-effect of alignment methods is that variable length biological sequences can be converted into fixed length sequences through appropriate insertion/gaps techniques.

Sequence alignment techniques are not confined to biological sequences and there have been applications of sequence alignment in linguishtics (Vaughan-Nichols, 2007) and marketing (Prinzie and Van den Poel, 2006). The first demonstration of the use of alignment for detecting computer viruses was reported in 2007 (McGhee, 2007) and used profile hidden Markov models (HMM) (Eddy, 1998) constructed form position specific information for scoring. However, only a subset of possible op-codes was used for alignment, resulting in the exclusion of many viral op-codes that were deemed not important. Also, the HMM did not work on a biological representation of the op-codes but on an alphabet that was neither DNA nor amino acid.

The alignment of multiple sequences in a given query set is often used of identifying conserved sequence regions across a group of sequences hypothesized to be evolutionarily related (wikipemedia, 2012). Typically a pairwise alignment is performed first and then information produced during pairwise alignment to perform a multiple alignment. At least one substitution matrix is used during the two phases. These substitution matrices reflect the effects of pre-defined biological relationships between residues when alignments are formed during the first and second phases. The alignment result will be produced depending on the substitution matrix, the alignment algorithm, the stage of alignment and the different amino acid representations.

3.4 RESEARCH QUESTIONS

The aim of this thesis is to explore the use of bio-informatics techniques (Chapter 5) to analyse malware. The overall approach will be divided into two parts. The first part involves creating a multiple sequence alignment for a predefined set of malware in the same family, and then aligning both families together to create a new sequence alignment. Such motifs (signatures) can also be consistent for a 'family' of viruses or worms that share parts of the code or have similar function and are essentially variants of each other.

The second part will use neural networks to test the effects of representing malware signatures as bio-sequences and the alignments. We also use a symbolic ML method to find the rules for classification. Then the whole methodology will be applied to random sequences to verify its effectiveness.

The second question addressed in this thesis is to establish whether motifs/meta signatures are accidental by-product of the representations used or are the evidence of a deeper and unpredicted aspect of applying bio-sequence techniques to artificial viruses and worms.

3.5 CONCLUSION

In this chapter we reviewed two of the most commonly used techniques for malware detection and the associated problems. There is a scope for a different approach to malware detection. The aim of this thesis is to explore a different signature-based approach that is based on biological representations of the signatures prior to data mining.

Research methods will be introduced in next chapter. The first part of experiments and results are described in Chapter 5.

Chapter 4 Research Methodology

4.0 INTRODUCTION

Chapter 2 reviewed the characteristics of a few different types of malware. Chapter 3 introduced the existing methods and technologies for malware detection and pointed out their weaknesses of these.

The aim of this chapter is to present that is applied in this research. The scientific research methodology (SRM) provides the most logical and constructive way to conduct experiments when there is little previous research on which to base the new research on question and experiments.

This chapter is organized as follows. After a general overview of SRM, research questions will be formulated.

The research method to be adopted in this thesis is the scientific research method and it is illustrated by the diagram in Fig. 4.1.

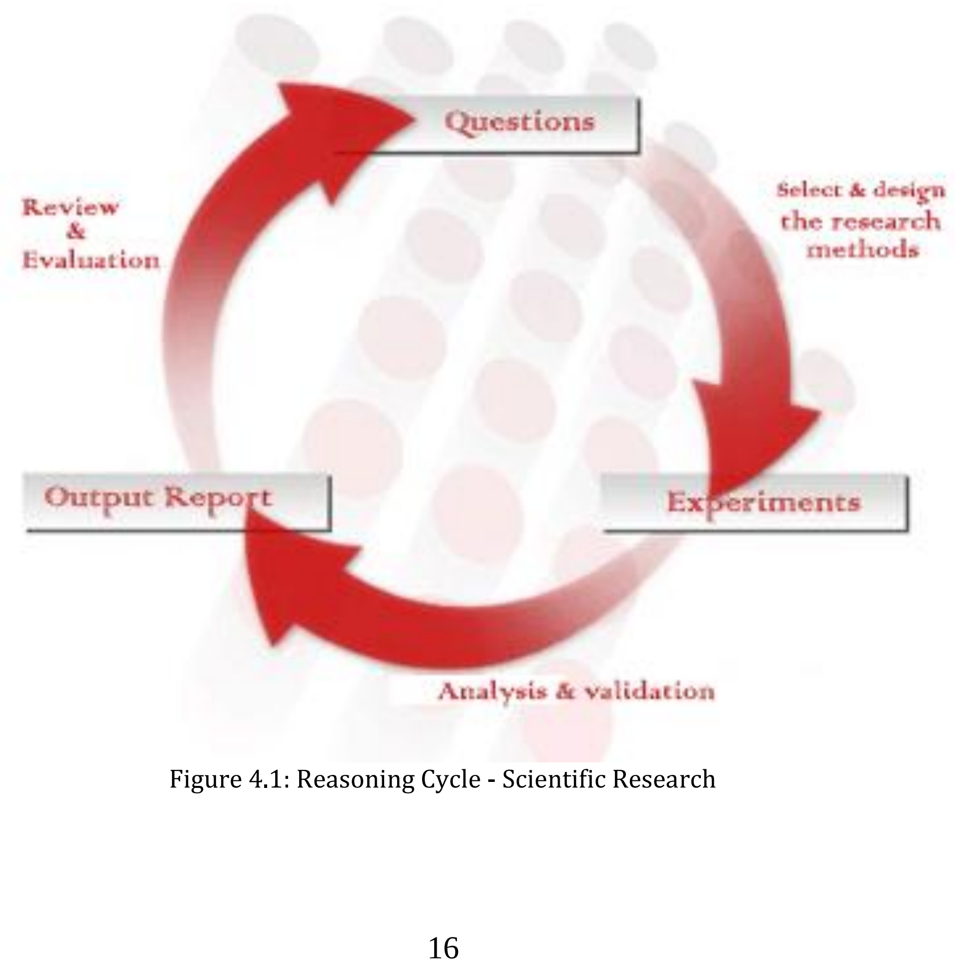

Figure 4.1: Reasoning Cycle - Scientific Research

4.1 RESEARCH QUESTIONS

In this section, the research method and research questions are introduced and discussed. This can be linked to the phase ‘Questions’ in the graph shown in figure 4.1.

We start work by gaining understand of the background to the research. To select a study domain, characteristics of different malware were introduced in Chapter 2. The traditional method for dealing with computer viruses and worms (two of most common types of malware are selected in this research) is to use anti-viral software that looks for ‘signatures’. Because of sophisticated techniques, the infection may already have reached epidemic proportions before new signatures are released. Also, the requirement for fixed-length sequences of the machine learning techniques causes high fault rate of behaviour detection. The few works that use machine learning methods focuses on anomalous behaviour detection (e.g. Rieck *et al*., 2008; Singhal and Raul, 2012) rather than on code or signature analysis.

In bioinformatics, sequence alignment techniques are used to identify conserved regions or motifs in biological data. There have been applications of sequence alignment in linguistics (Kondrak, 2002) and in marketing (Prinzie and Van den Poel, 2006).

The next step is to clarify the purpose / formulate the question

After conducting the literature review, a research question (Indicated as: “Questions” in fig.4.1) will be formulated as follows:

“Malware signature identification will be improved by using techniques from biological sequence analysis.”

The null hypothesis will be that no improvement is achieved when using the bioinformatics techniques reviewed in this thesis. The question can be further refined to: “Can we get any benefit or improved accuracy if we used bioinformatics data mining to identify malware signatures.”

4.2 SELECT & DESIGN THE RESEARCH METHODS:

The scientific method is a body of techniques for investigating phenomena, acquiring new knowledge, or correcting and integrating previous knowledge (Goldhaber and Nieto, 2010).

The scientific method is a way to ask and answer scientific questions by making observations and doing experiments. Common steps of the scientific methods are:

- Ask a question;
- Do background research ;
- Construct a hypothesis;
- Test hypothesis by doing an experiment;
- Analyze data and draw a conclusion;
- Communicate results.

The main challenge in this research is the lack of previous experimental work on the application of biological sequence techniques to malware signatures. There is no previous work on which to base a fully evidenced set of experimental methods in this thesis. The design process consists of a set of steps that start first with the identification of a problem or a need and lead to creating and developing a solution that solves the problem or meets the need. The steps of the methodology used in this thesis are to:

- Define the problem;
- Do background research;
- Design the method;
- Experiments (Create alternative solutions);
- Analysis and validation;

- ➢ Output report;
- ➢ Review and Evaluation (Do development work);
- ➢ Redefine the problem.

The task is to design some experiments and obtain results that can act as a reference for future researchers.

4.3 EXPERIMENTAL METHODS

The main problem is how biological sequence analysis can be applied to malware signature identification. The approach to be adopted in this thesis is as follow (Identified as: “Experiments” in fig. 4.1.):

0. Represent two classes of data (virus and worm signatures) as amino acids;

a. For each class, align the sequences using an appropriate method and substitution matrix (result is variable length samples in each class);

b. Represent the gaps with an amino acid;

c. Align all samples of all classes together (results in longer sequences with fixed length);

d. Repeat (a) – (c) but with changes of parameters changes for different experiments;

e. Apply ML techniques and compare with ML results using the original and unaligned data;

f. Evaluate the datasets with the best alignment and representation and extract rules.

The experiments will be carried out on two sets of virus signatures (30 and 60 viruses respectively) as well as on two sets of worm signatures (30 and 60 worms respectively). The results of these experiments will be considered for research revision and for further work. The actual choice of the bioinformatics sequence alignment technique to be adopted will be made after the review. However, it appears from preliminary work that local and global alignment methods should be adequate for testing the question in the first instance.

4.4 ANALYSIS OF RESULTS & VALIDATION:

The analysis and evaluation of our research is carried out by using protein analysis tools such as T-Coffee and ClustW with different matrices, data mining (such as Navie Bayes, J48, LADTree, OneR, Perceptron and Prism) and neural network techniques (Perceptron and JavaNNs).

For reporting the test results, the standard formula for accuracy using numbers of instances in each category is adopted. Details will be introduced in the next chapter.

At this stage we also present comments and conclusions for each case.

4.5 REVIEW & EVALUATION OF OUTPUT REPORTS

In this section, the experiments results are reported and evalutated. This can be link to the phase "Output Report" in the graph shown in fig. 4.1.

The differences in classification rules resulting from the first cycle will raise a common question: would random sequences get the same benefit if we used the same methods? If the answer is yes, it will mean that our methodology may have a big problem. As D.H Lehmer stated in 1951: "A random sequence is a vague notion…in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests traditional with statisticians." (Philip, 2006) It means we ought to find no rules in multiple random sequences.

A set of new experiments will be added:

g. Create random sequences;

h. Represent two classes of data (30 random sequences in each class) as amino acids;

i. Repeat (a) – (c) working on these random sequences

j. Evaluate the datasets with the accuracies and extract rules which will be introduced in next chapter.

Further explorations of alignment techniques will be briefly described in the Chapter 5 of the part I experiment.

4.6 REDEFINED QUESTIONS:

The differences in classification accuracy resulting from the first cycle will raise the following question (return to: “Questions” in fig. 4.1): Does finding biological analogues of malware signature represented as biological sequences and aligned with bioinformatics multiple sequence alignment tools shed a new light on problems typically identified as belonging to classical ML and data mining?

The accuracy of classification in the results from these representations raises the following questions: Does some fundamental sharing of structural information exist between natural infection agents and artificial infection agents? If the answer is yes, maybe we can find a new method to detect the malware by nature treatment or vice versa.

4.7 EXPERIMENTAL METHODS:

Answering these questions will be attempted by experiments with 60 viruses and 60 worms. The results of these experiments will be considered for future work. The actual choice of bioinformatics sequence alignment technique to be adopted will be made after the review (return to: “Experiments” in fig. 4.1).

Follow by the experiments steps (from section 4.3 to section 4.6)

k. Check the sequences against naturally occurring proteins.

Chapter 6 deals with step k.

4.8 OUTPUT REPORT

The analysis and evaluation is this research are carried out by using nature protein structure analysis tools (multiple alignment, such as ClustalW and T-coffee with matrix), 3D view (modif3D), data mining and neural network techniques. At this stage we also formulate comments and conclusions in each case (return to: “Output Report” in fig. 3.).

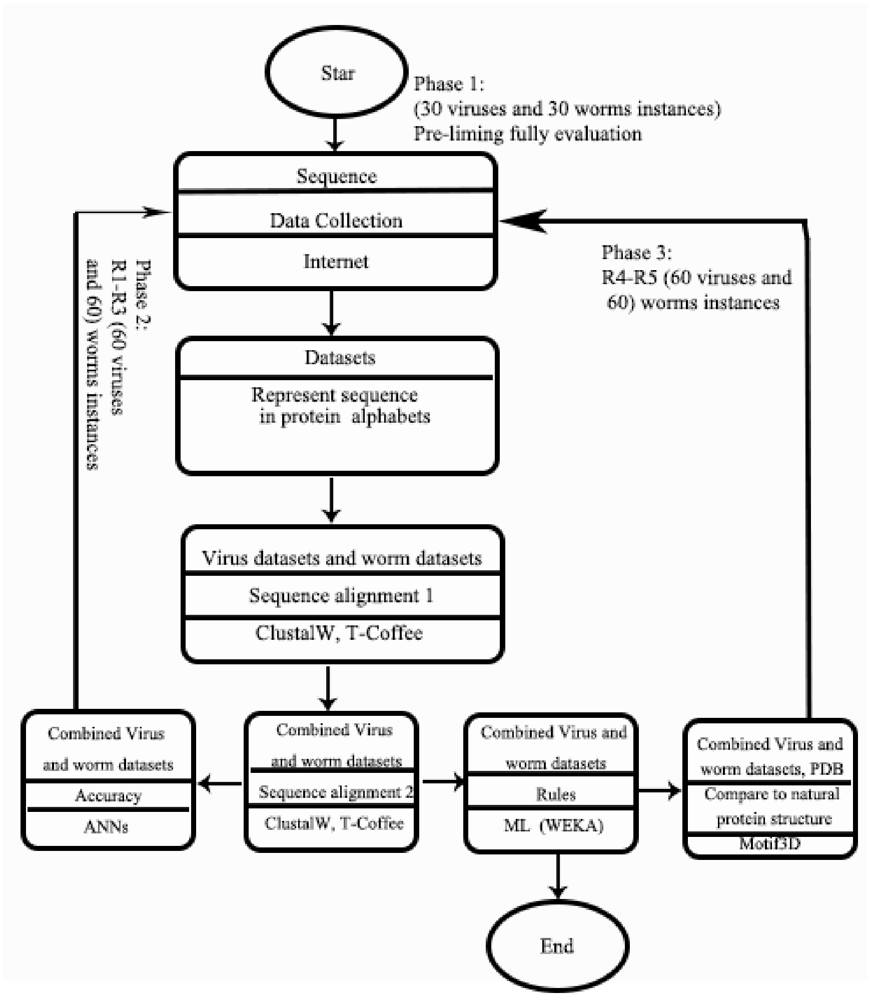
Star
Phase 1:
(30 viruses and 30 worms instances)
Pre-liming fully evaluation
Sequence
Data Collection
Internet
Phase 2:
R1-R3 (60 viruses and 60) worms instances
Phase 3:
R4-R5 (60 viruses and 60) worms instances
Datasets
Represent sequence in protein alphabets
Virus datasets and worm datasets
Sequence alignment 1
ClustalW, T-Coffee
Combined Virus and worm datasets
Accuracy
ANNs
Combined Virus and worm datasets
Sequence alignment 2
ClustalW, T-Coffee
Combined Virus and worm datasets
Rules
ML (WEKA)
Combined Virus and worm datasets, PDB
Compare to natural protein structure
Motif3D
End

Figure 4.2 Usage Methodology Layers of detail steps.

4.9 CONCLUSION

This chapter presents a scientific research methodology in the form of sequence analysis. It is the best possible way to focus the research process and organize the research by formulating and defining a research problem and draw conclusions that reflecting the real world.

The next chapter presents the details of research methods and experimental results were followed by. Chapter 6 would analyse the malware structure by 3D view. Last conclusion and further work would show on Chapter 7.

Chapter 5 System and Methods I

5.0 INTRODUCTION

Viruses can be written in any programming language before being compiled. Viral source code libraries exist on the Internet for experimental use and source code will not be used in this thesis. Instead, in line with viral signature detection, the compiled viruses, presented in hexadecimal code, are used here (VX Heavens, 2011). For instance, the first part of the virus 1C.Tanga.a computer virus has the hexadecimal coding:

8e5ef1aec91259d70c5e62cdfe42c36e

and the worm Bat.Agent.bo has the hexadecimal coding:

fb56373bde388174126fecf9143eeff2aae6b7486224c8fd213918abc38393357fa4fc670
.

The types of sequences analyzed in this thesis will be finite ordered list of symbols from an alphabet with a finite set of available symbols.

In the field of bioinformatics science, sequence analysis is used to understand the relationship between two or more sequences of genetic information, such as DNA or amino acids (Mount, 2001). DNA is a molecule which encodes the genetic instructions and along with RNA and proteins. Genetic information is encoded as a sequence of nucleotides (guanine, adenine, thymine and cytosine) recorded using the letters G, A, T and C (Wikipedia, 2012). On the other hand, Amino acids are the structural units that make up proteins (Wikipedia, 2012). Currently, about 500 amino acids are known and can be classified in many ways. Twenty-two amino acids are naturally incorporated into polypeptides, in which 20 are encoded by the universal genetic code (Creighton and Thomas, 1993).

Hence the method chosen to represent computer virus sequences is the protein alphabet rather than the DNA alphabet.

5.1 Hexadecimal-CODE REPRESENTATIONS

In first three experiments, we will use table 5.1 of amino acid alphabet to convert hexadecimal cods of 60 datasets. In all cases ‘Z’ is used to represent gaps introduced in the first alignment of worms and viruses separately, and ‘Y’ for gaps in the second alignment of worms and viruses jointly.

Table 5.1: Representation of hexadecimal code (bold) in amino acid (residue) alphabet in 60 datasets

Hexadecimal code	**Amino Acid Alphabet**
1	A
2	C
3	D
4	E
5	F
6	G
7	H
8	I
9	K
0	L
a	M
b	N
c	P
d	Q
e	R
f	S
-	Y
-	Z

Three different representations of the hexadecimal code in the amino acid alphabet were tried for alignment purposes (Table 5.2) which will be used in the rest of experiments. The first representation (R1, 2nd column of Table 5.2) uses the same order of hexadecimal to amino acid residues as Table 5.1 except gap representation. The second (R2, 3rd column of Table 5.2) reverses this order and the third (R3, 4th column of Table 5.2) uses a shift of one amino acid residue after the initial residue. In all cases 'W' is used to represent gaps introduced in the first alignment of worms and viruses separately, and 'Y' for gaps in the second alignment of worms and viruses jointly.

Table 5.2: Three different representations of hexadecimal code (bold) in amino acid (residue) alphabet in 120 datasets

Hexadecimal code	**R1**	**R2**	**R3**
1	A	S	A
2	C	R	D
3	D	Q	E
4	E	P	F
5	F	N	G
6	G	M	H
7	H	L	I
8	I	K	K
9	K	I	L
0	L	H	M
a	M	G	N
b	N	F	P
c	P	E	Q
d	Q	D	R
e	R	C	S
f	S	A	C
-	Y	Y	Y
-	W	W	W

Given the rewriting capabilities of malware and their many different variants, the research question is whether multiple alignment followed by data mining will find the 'conserved regions' of the hexadecimal code that can be interpreted as signatures, where the signatures themselves denote common function, structure or ancestry. Previous work on pairwise alignment of computer viruses used the 20-character amino acid alphabet to represent residues and 10 digits (0-9) to map the op code of viruses into sequence patterns, and sent the resulting sequences as input to an HMM (McGhee, 2007). This allowed a maximum of 30 op codes to be represented for alignment, resulting in several op codes being excluded from analysis. Previous work on real viruses, on the other hand, used a moving window approach of nine residues to produce several hundred sequences of fixed length from one viral protease sequence for input to a neural network (Narayana, 2002).

The approach taken here is to represent each of the hexadecimal characters by its 'amino acid' character (Table 5.1 and Table 5.2) for alignment purposes.

(a) virus.1C.Tanga.a:
8e5ef1aec91259d70c5e62cdfe42c36e
IRFRSAMRPKACFKQHLPFRGCPQSRECPDGR

(b) Bat.Agent.bo the hexadecimal coding:

fb56373bde388174126fecf9143eeff2aae6b7486224c8fd213918SNFGDHDNQRD
IIAHEACGSRPSKAEDRRSSCMMRGNHEIGCCEPISQCADKAI

In the above, the computer virus virus.1C.Tanga.a in hexadecimal code form is converts to full op-code sequence.

5.2 CHOOSING TOOLS OF ALIGNMENTS

In this thesis, T-Coffee (Notredame et al., 2000) and ClustalW (Thompson *et al.*, 1994) from the European Bioinformatics Institute are used in all experiments for aligning the signatures represented in the four ways presented in Table 5.1 and Table 5.2.

T-Coffee is widely used due to its ability to align sequences without needing a predefined substitution matrix, such as BLOSUM (Henikoff and Henikoff, 1992) that reflects highly conserved regions of existing protein families. Instead, T-Coffee, if so specified, can construct position-specific scoring schemes using the actual sequences input for alignment. However, due to the way that T-Coffee works (local pairwise alignment is undertaken first before a global alignment is performed) it is possible for different representations of the same set of sequences to produce different alignment results. This is because T-Coffee builds up a primary weighting scheme during the pairwise alignment based on ClustalW, then FASTA (Lipman and Pearson, 1985), and both of these use default parameters that can affect the weighting adopted for the final multiple alignment depending on the representation used. ClustalW default parameters are used to set gap penalties (based on an evolutionary model where gaps represent insertion or deletion mutations) after calculating a distance matrix from similarity scores for every possible pair of sequences. FASTA default parameters are used to create distance matrices based on the closest sequence in a biological database of similar sequences.

To check for the possibility of the pre-defined biological relationships between residues impacting on the effect of different representations for alignment, ClustalW was first used with default parameters except for the choice of substitution matrix.

T-Coffee analyzes sequences two by two in the first step and produces a global multiple sequence alignment in the second step. Residue substitution matrices (e.g. alignment using probabilistic mutation rates of residues based on the actual observed alignments rather than external biologically-based mutation rates) are switched on (BLOSUM, or BLOcks (Henikoff and Henikoff, 1992) of amino acid SUbstituion Matrices) to derive sequence-specific substitutions.For fourth (section 5.6.4) and fifth (section 5.6.5) alignment experiments, T-Coffee was used with BLOSUM as the substitution matrix.

In order to reduce the impact of Protein Data Bank on alignment, we used ClustalW with four different substitution methods. The first set of alignment experiments used ClustalW with default parameters except for the choice of substitution matrices, which were set to unitary matrices (1 for a positive self-match, 0 or negative for a mismatch) for both phases of alignment. This results in gaps being introduced as part of the alignment but not substitutions. The second set of alignment experiments with ClustalW switched on the substitution matrix BLOSUM (Henikoff and Henikoff, 1992) that reflects highly conserved regions of existing protein families. The third set of experiments with ClustalW used another substitution matrix, Gonnet (Gonnet *et al*., 1992). Generally, Gonnet matrices represent evolutionary substitution and therefore distance information gained from analyzing all protein sequences known in 1992, whereas BLOSUM matrices use conservation and similarity of function information within the sequences being analyzed. BLOSUM can lead to alignments where amino acids are blocked in appearance in comparison to other alignment techniques that can disperse the amino acids more widely in the alignment. It is well known that different substitution matrices produce different results on the same set for sequences reflecting different purposes (Hara *et al*., 2010).

FASTA format (Lipman and Pearson, 1985) was used as the input method for all alignments.

5.3 Research Methods

In this section, the details of research methods are introduced and discussed.

5.3.1. Step (0): Represent two classes of data in amino acids.

- Convert the 30 worm signatures into amino acid representation datasets using Table 5.1. Call this dataset ‘W30’;
- Covert the 30 worm signatures into three amino acid representation datasets using Table 2. Call these datasets W30R1-W30R3 (for ‘worm representation 1’ - ‘worm representation 3’ in dataset W30);
- Repeat for the virus signatures (‘V30’ and ‘V30R1’ – ‘V30R3’). Each sample in the dataset is an artificial polypeptide sequence consisting of amino acid letters;

- Covert the 60 worm signatures into three amino acid representation datasets using Table 5.2. Call these datasets W60R1-W60R3 (‘W’ for worm; ‘’60 is the number of instances and R1-R3 map the R1-R3 representations in table 5.2);
- Repeat for the 60 virus signatures (‘V60R1’ – ‘V60R3’). Each sample in the dataset is an artificial polypeptide sequence consisting of amino acid letters.

5.3.2. Step (a) Align the sequences of computer viruses and worms separately.

- Input all 30 ‘W30’ worm polypeptide sequences into T-Coffee without substitution matrix to form an initial set of aligned worm sequence (‘SAWT30’, for 30 single aligned worms using T-Coffee without substitution matrix). Code gaps as ‘Z’.

- Input all 30 ‘W30R1’ worm polypeptide sequences into T-Coffee without substitution matrix to form two initial sets of aligned worm sequences (SAWT30R1, for 30 single aligned worms using T-Coffee without substitution matrix’ and Representation ‘1’). Code gaps as ‘W’. Repeat for W30R2 and W30R3 worms, resulting in SAWT30R2 and SAWT30R3. Store the three consensuses produced by the multiple alignment as WCCI30R1-WCCI30R3, for 30 worms consensus using ClustalW with Identity matrix Representation 1 to 30 worms consensus ClustalW with Identity matrix Representation 3.

- Input all 60 ‘W60R1’ worm polypeptide sequences into ClustalW using the unitary matrix to form an initial set of aligned worm sequences (SAWCI60R1, for ’60 single aligned worms using ClustalW with Identity matrix and Representation 1’). Code gaps as ‘W’. Repeat for W60R2 and W60R3 worms, resulting in SAWCI60R2 and SAWCI60R3. Store the three consensuses produced by the multiple alignment as WCCI60R1-WCCI60R3, for ’60 worms consensus using ClustalW with Identity matrix Representation 1’ to ’60 worms consensus ClustalW with Identity matrix Representation 3’. We return to these consensus sequences in the second set of systems and methods (Chapter 6).

5.3.3. Step (b) Represent the gaps with an amino acid

- Input all 30 'V30' virus polypeptide sequences into T-Coffee without substitution matrix to form an initial set of aligned virus sequence ('SAVT30', for '30 single aligned viruses using T-Coffee without substitution matrix). Code gaps as 'Z'.

- In put all 30 'V30R1' virus polypeptide sequences into T-Coffee without substitution matrix to form two initial sets of aligned virus sequences (SAVT30R1, for '30 single aligned viruses using T-Coffee without substitution matrices and Representation '1'). Code gaps as 'W'. Repeat for V30R2 and V30R3 viruses, resulting in SAVT30R2 and SAVT30R3.

- Input all 60 V60R1 virus polypeptide sequences into ClustalW (unitary matrix) to form an initial set of aligned virus sequences. Code gaps as 'W'. Repeat for V60R2 and V60R3 viruses. This results in three single aligned datasets SAVCW60R1-SAVCW60R3. Store the three consensuses produced by ClustalW (VCCW60R1-VCCW60R3) for later use in a second set of experiments (Chapter 6).

5.3.4. Step (c) Align all samples together

- Combine the two corresponding single aligned sets SAVT30 (virus) and SAWT30 (worm) into one set put into T-Coffee to do double alignment. We will find the length of the virus set alignment will be different from the length of the worm set alignment. This is because there is no guarantee that T-coffee will make the same number of insertions (gaps) for both sets of sequences. Code all gaps introduced as this stage as 'Y'. Call this set 'DT60' for later use.

- Combine the two corresponding single aligned sets SAWT30R1 (worm) and SAVT30R1 (virus) into one set and input into T-Coffee to form a second, doubly aligned set of polypeptide sequences. Code all gaps introduced at this stage as ‘Y’. Repeat for R2 and R3 worm and virus. Call these datasets DT60R1, DT60R2 and DT60R3 where ‘T’ stands for ‘T-Coffee’, 60 is the number of total instances and R1-R3 are as described previously (Table 5.2).

- Combine the two single aligned sets SAWCW60R1 (worm) and SAVCW60R1 (virus) into one set and input into ClustalW (unitary matrix) to form a second, doubly aligned set of polypeptide sequences. Code all gaps introduced at this (double alignment) stage as ‘Y’. Repeat for R2 and R3 worm and virus singly aligned polypeptide sequences. This results in three datasets consisting of doubly multiply aligned worm and virus polypeptide sequences, with each data set containing sequences using R1, R2 and R3, respectively. Call these datasets DACI120R1, DACI120R2 and DACI120R3 where ‘DA’ is ‘doubly aligned’ (i.e. combined, doubly aligned set), ‘CI’ is ‘ClustalW with identity/unitary matrix’ and R1-R3 are as described previously (three different representations using Table 5.2).

5.3.5. Step (d) Repeat (a)-(c) but with the following changes:

In third alignment experiment:

(i) select 10 worm signatures and 10 virus signatures
(ii) select 20 worm signatures and 20 virus signatures
(iii) select 60 worm signatures and 60 virus signatures
(iv) select 70 worm signatures and 70 virus signatures

In fourth alignment experiment:

(i) switch on BLOSUM with T-Coffee for both phases of alignment;

In fifth alignment experiment:

(i) switch on Gonnet within ClustalW for both phases of alignment;
(ii) switch off Gonnet and switch on BLOSUM within ClustalW for both phases of alignment;

(iii) use T-Coffee and switch on BLOSUM at the final stage of multiple alignment.

All the results obtained from the above experiments are discussed in Section 5.6. This results in 12 more doubly aligned datasets: TB120R1-3, where ‘TB’ is T-Coffee with BLOSUM switched on; DACGR1-3, where ‘CG’ is ClustalW with Gonnet; DACBR1-3, where ‘CB’ is ClustalW with BLOSUM; and DATBR1-3, where ‘TB’ is T-Coffee with BLOSUM switched on. The templates DAxyR1-DAxyW3 will be used as follows: x=C(lustal) or T(-Coffee); y= I(dentity) or G(onnet) or B(LOSUM). These 19 datasets will be input to five different symbolic machine learning techniques using their amino acid representations or, in the case of the perceptron, their numeric equivalents as explained below.

5.4 STEP (E) – (F) MACHINE LEARNING ANALYSIS

5.4.1 Step (e) Apply ANNs techniques JavaNNs

ANNs is a mathematical model that is inspired by the structure and functional aspects of biological neural networks (Narayanan et al, 2002). A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation.

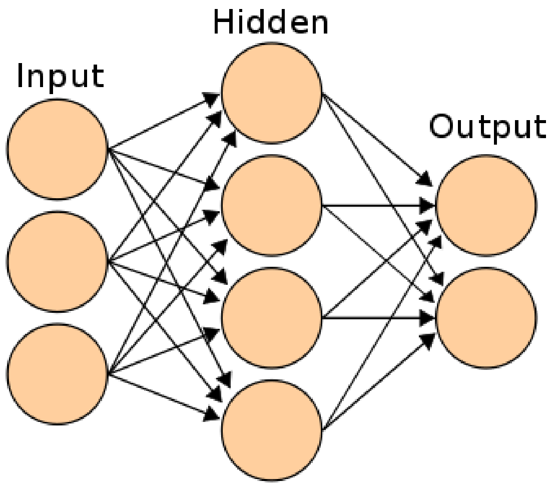

Figure .5.1: An artificial neural network is an interconnected group of nodes (Wikimedia, 2012)

These networks are similar to the biological neural networks in the sense that functions are performed collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which various units are assigned.

As neural networks only accept numeric format, for training neural networks, the amino acids were converted into their ASCII integer representations (Table 5.3) in the first experiment. For instance, the first four hexadecimal characters of the 1c.Tanga.a virus ‘8e5e’ will be ‘IRFR’ in amino acid representation and then ’73 82 70 82’ in ASCII represented.

Table 5.3: *Conversion of the 16 amino acid alphabet to ASCII integer representation. Y and W (two extra characters) are explained in the main text.*

A	C	D	E	F	G	H	I	K
65	67	68	69	70	71	72	73	75
L	M	N	P	Q	R	S	Y	W/Z
76	77	78	80	81	82	83	89	90

From the second experiment (results in section 5.6.2), all datasets (19 in total) are converted from their polypeptide alphabetic representation into numeric using the replacements specified in Table 5.4. To take into account the arbitrary nature of the conversion of amino acid residues to numerical values, a hidden layer of 72 units is introduced in the sixth experiment. The first and second experiments had shown that a 72-node hidden layer, in comparison to other architectures, was most effective. The numeric conversion was also used successfully in second experiment and is used again in the following three experiments. A two-layer network can also deal with the input values and their mapping to the output value in a non-linear way. WEKA perceptrons were used to implement the neural networks, which have as many input nodes as there are amino acids in the fixed length, doubly aligned polypeptide sequences. For WEKA, each residue position was given its own attribute name defined by its position in the doubly aligned sequence. Finally, for supervised machine learning, a ‘0’ is affixed to each virus sample and a ‘1’ to each worm sample.

Table 5.4: Conversion of the 16 amino acid alphabet to numeric form between 0 to 1 for input to perceptrons. Y and W (two extra characters) represent the gaps introduced during alignment

A	C	D	E	F	G	H	I	K
0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
L	M	N	P	Q	R	S	Y	W
0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95

5.4.2. Report accuracy results of JavaNNs.

Input the sequences into the data mining tool JavaNNS which is written around a simulation kernel to which user written activation functions, learning procedures and output functions can be added. It has support for arbitrary network topologies and the standard release contains support for a number of standard neural network architectures and training algorithms Each residue position was given its own attribute and the two predefined classes were ‘virus’ and ‘worm’. The machine learning task was therefore to determine whether the two stages of multiple alignments could lead to improved classification and separation between viruses and worms.

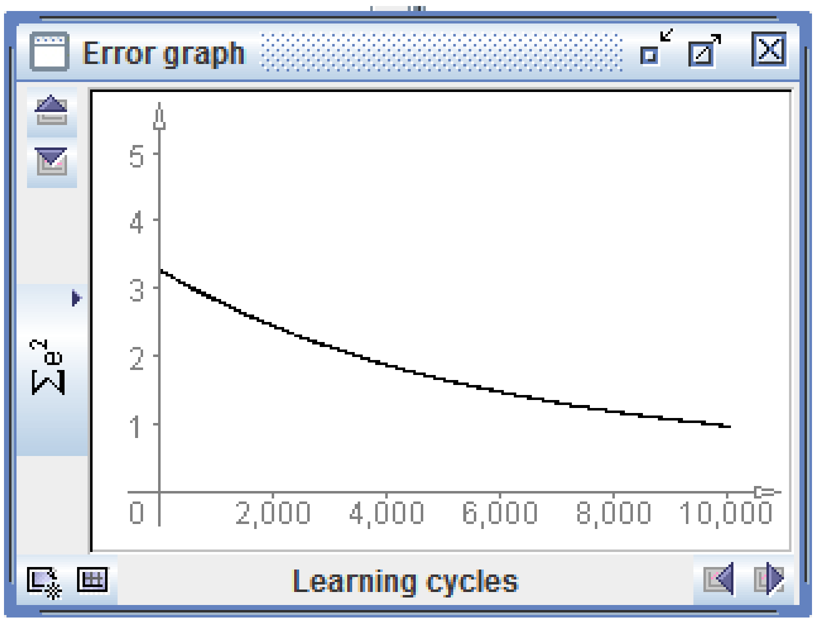

Figure 5.2: JavaNNS Error Graph which shown after learning, the error rate is decreasing.

Then test all three representations for generalizability and ability to withstand over-fitting across all machine learning techniques. The first condition of generalizability is implemented through a 66:34 training and testing regime, and the aim is to check how well the representations classify a relatively large proportion of unseen cases. The second condition, the ability to withstand over-fitting is implemented through a 10-fold training and cross-validation regime, with 90% of samples used for training and 10% for cross-validation. Over-fitting can occur if the trained classifier has problems classifying a relatively small proportion of unseen cases. Report overall accuracy results for each of the machine learning techniques across both test conditions.

5.4.3 Step (f) Apply machine learning techniques and compare with machine learning results using the original and unaligned data

Analyze all three representations for richness of knowledge extraction using PRISM (Cendrowska, 1987) available in WEKA and analyze the results for possible viral and worm meta-signatures. Whereas previously J48 (a rule extractor with pruning) had been used in the third experiment, here we report on the application of PRISM (a modular rule extractor) to help compare the results. PRISM was used with no testing to maximize knowledge extraction (i.e. all samples were used for rule extraction).

5.4.4 Report accuracy results and extract rules

Evaluate the datasets with the best alignment and representation, as given by the machine learning results, as if they are real biological sequences to check for any added value to supplement the machine learning results. The methods adopted for this stage are described in more detail in Chapter 6 below.

Aligning the 30 or 60 virus and 30 or 60 worm polypeptide sequences separately (a-b above) allows the conserved regions of the virus and worm sequences across families within each class to be independently extracted. For instance, after alignment by T-Coffee, we have (for the first parts of three viral polypeptide sequences using R1):

```
FIIDIDNGLFDSRPLEEFKGALEGEI...
GE-----SQMPSIDMPQF---PGLPS...
---------ILHSPMHQFRF-PRSQR...
                  :  :*
```

This shows that only F is aligned across all three sequences (‘*’), and M and Q across two sequences (“:”). The gaps ‘-‘ introduced at this stage are coded ‘W’. When these aligned virus sequences are themselves aligned with their worm sequence counterpart representations, the gaps introduced are coded ‘Y’ (step (c) above). Y and W have their own representation for perceptron input (Table 5.2).

Four symbolic machine learning algorithms were used in addition to the (numeric) perceptron: Naïve Bayes, J48, LAD Tree and OneR. These four ML techniques were chosen because of their intelligible output, long establishment in the ML area and variety from each other. Naïve Bayes is a simple version of Bayesian classifiers that only looks at the relationship between a particular feature and classification, i.e. no attention is paid to combinations of features. J48, as noted earlier, is an implementation of ID3 and C4.5 tree induction algorithms and uses information-theoretic formulae (Quinlan, 1993). The LAD tree algorithm is an approach using least absolute deviation (LAD) error to obtain regression trees that can handle discrete variables through recursive partitioning (Breiman *et al.*, 1983). OneR is a single rule classification algorithm that produces a tree of only one level, where the rule is a set of attributes associated with their majority class (Holte, 1993). For reporting the test results, the following standard formula for accuracy using numbers of samples in each category is adopted (virus is negative (N), worm is positive (P), true is T and false is F)*:*

$$\text{Accuracy}= \frac{\text{Number of true positives + number of true negatives}}{\text{Number of true positives + false positives + false negatives + true negatives}}$$

$$\text{Precision}= \frac{\text{Number of true positives}}{\text{Number of true positives + number of false positives}}$$

$$\text{Sensitivity}=\frac{\text{Number of true positives}}{\text{Number of true positives + number of false negatives}}$$

Table 5.5 the accuracy is the proportion of true results (wikimedia, 2012)

		Condition as determined by *Gold* standard		
		True	**False**	
Test outcome	**Positive**	**True positive**	**False positive**	→ <u>Positive predictive value</u> or Precision
	Negative	**False negative**	**True negative**	→ <u>Negative predictive value</u>
		↓ <u>Sensitivity or recall</u>	↓ <u>Specificity (or its complement, Fall-Out)</u>	Accuracy

5.5 STEP (G) – STEP (J) RANDOM SEQUENCE EXPERIMENTS

5.5.1 Step (g) Create random sequences

D.H Lehmer stated in 1951: "A random sequence is a vague notion…in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests traditional with statisticians." (Philip, 2006). In most cases, theorems relating the three paradigms (the frequency approach, the complexity/compressibility approach and the predictability approach) have been proven (Widmayer et al, 2002).

In order to create random sequences set of the 16 HEX numbers standard Excel function INT() and RAND() are used.

INT (RAND() * (17 – 1) + 1)

5.5.2 Step (h) Represent random sequences in amino acids.

- Convert the 30 random sequences into amino acid representation datasets using R1 in Table 5.2. Call this dataset 'R1RW30'(R1 map R1 representation of table 5.2; R means random; 30 is the instance number and W is class name);

- Repeat for the rest 30 random sequences ('R1RV30'). Each sample in the dataset is an artificial polypeptide sequence consisting of amino acid letters;

5.5.3 Step (i) Repeat (a)-(e) but with the following changes:

(i) Select 30 worm signatures and 30 virus signatures which length equal to 10. Call the datasets 'R1R10W30' and 'R1R10V30';

(ii) Select 30 worm signatures and 30 virus signatures which length equal to 20. Call the datasets 'R1R20W30' and 'R1R20V30';

(iii) Select 30 worm signatures and 30 virus signatures which length equal to 40. Call the datasets 'R1R40W30' and 'R1R40V30';

(iv) Select 30 worm signatures and 30 virus signatures which length equal to 60. Call the datasets 'R1R60W30' and 'R1R60V30';

(v) Select 30 worm signatures and 30 virus signatures which length equal to 72. Call the datasets 'R1R72W30' and 'R1R72V30'; and

(vi) Select 30 worm signatures and 30 virus signatures which length equal to 98. Call the datasets 'R1R98W30' and 'R1R98V30';

(iv) Switch on identity within ClustalW for both phases of alignment (CI);

These results in 6 more doubly aligned datasets: R1RxCI60, where ‘CI’ is ClustalW with Identity where x will be represented with different length of random sequences (i.e. x = 10, 20, 40, 60, 72 and 98). These six datasets will be input to Prism using their amino acid representation. The results of the experiments are discussed in section 5.6.6.

5.5.4 Step (j) Evaluate the datasets with the accuracies and extract rules.

Here we report on the application of PRISM (a modular rule extractor) to help compare the results. PRISM runs for generalizability and ability to withstand over-fitting across all machine learning techniques. The first condition that of generalizability is implemented through a 66:34 training and testing regime, and the aim is to check how well the representations classify a relatively large proportion of unseen cases. The second condition, the ability to withstand over-fitting is implemented through a 10-fold training and cross-validation regime, with 90% of samples used for training and 10% for cross-validation. Over-fitting can occur if the trained classifier has problems classifying a relatively small proportion of unseen cases.

5.6 Experiment results

In the previous, all steps in the proposed research methodology were described. This section will analyzes the effectiveness of representing problematic data as bio-sequences followed by alignment and machine learning from these experimental results.

The first set of experiments will use the signatures of 30 viruses and 30 worms. T-Coffee is used in this set of experiments for aligning the signatures represented in table 5.1. ANNs is one of the neural network techniques, which will be used for data mining in the first set of experiments. Because neural network only accept numeric format, for training neural networks, the amino acids are converted into their ASCII integer representations (table 5.3).

In order to find more effective architectures and representations, we also use T-Coffee and ANNs in the second set of experiments. All datasets are converted from their polypeptide alphabetic representation into numeric form using the encoding specified in Table 5.4.

The third set of experiments is to research the rules and features between instances of viruses and worms, we selected new10, 20, 30, 60 and 70 viruses and same numbers of samples of worms to repeat the experiment with T-Coffee.

The objectives of the fourth experiment are to explore the implications of adopting or not adopting predefined substitution matrices using different residue representations (table 5.2) for signatures as well as using a greater variety of machine learning tools for evaluating the effects of different representations for classification and prediction.

In the fifth set of experiment increased the samples from 60 (30 viruses and 30 worms) to 120 (60 viruses and 60 worms).

At the end, we will add a set of experiments with random sequences in order to validate the whole experimental methods.

5.6.1 EXPERIMENTAL RESULTS OBTAINED FROM T-COFFEE AND NEURAL NETWORK ON 30 VIRUSES AND 30 WORMS

A number of experiments were first conducted to find the best ANN architecture for learning to distinguish between the worm and the viral non-aligned (‘raw’) sequences to get a benchmark. Different learning rates and architectures, as well as different data input order, were tried. Ultimately, after much testing, the best architectures for the non-aligned sequences (Step 0) were found to be a unit with 72 inputs, 36-18 double hidden layer sets of units and 1 output unit architecture for fixed order input of sequences (three layers of connections), and a 72 input, 72 hidden and 1 output architecture (two layers) for random presentation of sequences during each learning cycle. Table 5.6 presents the results of the best five runs of non-aligned sequences fed to JavaNNS, with Sum of Squared Error (SSE) approaching 0 after

10000 cycles when all sequences were used as training samples (10 runs in total for each of the architectures with different random initializations of the ANN). The average SSE error is 16.6% (maximum 24.528%, minimum 2.843%) across the best runs of all five architectures. Since the best results depended on fixed order of presentation of samples, this was the method adopted for Step 1(e).

The multiple alignments of viruses (Step 2 (c)) resulted in fixed length sequences of 146 (including W gaps) and of worms 187 (Z gaps). When the two sets of aligned sequences were combined and re-aligned, the resulting multiple alignment led to fixed length sequences of 581 (Y gaps). The Y gap sequences were entered into JavaNNS using fixed order and all samples for training. A number of experiments with the architectures resulted in the identification of an architecture with two layers, 581 inputs, 290 hidden layers and one output. Table 5.6 presents the results of the best five runs and demonstrates that, when all samples are used for training with no testing, the non-aligned results (Table 5.6) are better in terms of convergence to 0 SSE.

Table 5.6: *Results Obtained from JavaNNS on original fixed length frames (72 residue characters), without alignment (training only, Step 0). η is the back-propagated maximum step width and RO indicates random order of presentation of samples. Each of architectures was run 1*

Structure (input x hidden x output)	**Parameter**			**Sum of squared error**	**Sum of square root error**
	η	**Cycles**	**RO**		
72x36x18x1	0.001	10000	no	0.25528	0.49696
72x72x1	0.0001	10000	yes	0.24416	0.491579
72x72x1	0.0001	10000	yes	0.23988	0.48778
72x72x1	0.0001	10000	no	0.06291	0.211503

72x72x1	**0.001**	**10000**	**no**	**0.02843**	**0.120466**

Table 5.7 shows the results of training and testing on the non-aligned sequences, for benchmark comparison purposes (Step 0 above). A leave-one-out strategy was adopted and the results are the average across all 60 neural networks. Table 5.7 provides the comparable figures for the doubly aligned sequences (Step 1(e)) and indicates a major improvement in training, a small drop in accuracy but also a small improvement in precision

Table 5.7: *Result obtained from JavaNNs when training-testing with unaligned sequences (60 neural networks) using a leave-one-out strategy, for benchmarking purposes (Step 0)*

Regime	Accuracy	Precision	Sensitivity
Training	80.00%	83.33%	78.125%
Testing	97.98%	97.78%	97.81%

Table 5.8: *Result obtained from JavaNNs when training-testing on doubly aligned sequences (60 neural networks) using a leave-one-out strategy (Step 1(c)).*

Regime	Accuracy	Precision	Sensitivity
Training	91.67%	100.00%	85.71%
Testing	96.67%	98.33%	95.21%

As can be seen, there is significant improvement in the training results but not the test results after alignment.

A number of further experiments were then conducted to try to understand the behaviour of the ANNs under different learning and testing conditions.

Table 5.9 provides the results of training only of the unaligned sequences (Step 0), but with a different η of 0.0005, showing a small improvement in training. Using the same value for the parameter (η:0.0005), Tables 5.10 and 5.11 show the results of adopting 90% training, 10% testing and 80% training, 20% testing strategies, respectively, for the unaligned sequences (rather than leave-one-out); all 11 runs in step 0 were performed for each regime and only the best 5 are reported in Tables 5.10 and 5.11. The overall average accuracy, precision and sensitivity across five runs for 90/10 is 91.12% (Table 5.10) and for 80/20 is 92.12% (Table 5.11). These figures provide a benchmark for training and testing aligned sequences (Step 5(d)) in Tables 5.12 and 5.13.

Table 5.9: *Results obtained from JavaNNS when training unaligned sequences, with η=0.0005 (Step 0).*

Regime	Accuracy	Precision	Sensitivity
Training	81.67%	86.67%	78.79%

Table 5.10: *Result obtained from JavaNNS when training unaligned sequences 90%*

Runs	Accuracy	Precision	Sensitivity
1	92.59%	92.59%	92.59%
2	92.59%	92.59%	92.59%

3	90.74%	92.59%	89.29%
4	90.74%	88.89%	92.31%
5	88.89%	88.89%	88.89%

Table 5.11: *Result obtained from JavaNNS when training unaligned sequences 80%*

Runs	Accuracy	Precision	Sensitivity
1	100.00%	100.00%	100.00%
2	95.83%	95.83%	95.83%
3	91.67%	91.67%	91.67%
4	89.58%	91.67%	88.00%
5	83.33%	83.33%	83.33%

The training plus testing regime whose results are presented in Tables 5.9, 5.10 and 5.11 (Step 0) was also repeated for the aligned sequences (Step 1(d)), and Table 5.12 provides the training only figures, indicating 100% accuracy. Tables 5.13 and 5.14 present the results to Tables 5.10 and 5.11 but for doubly aligned sequences, again indicating 100% accuracy. These results are better than the results for unaligned sequences presented in Table 5.10 and Table 5.11

Table 5.12: *Results of JavaNNS on doubly aligned sequences*

Runs	Accuracy	Precision	Sensitivity

Training	100.00%	100.00%	100.00%

Table 5.13: *Result of JavaNNS on doubly aligned sequences 90%*

Runs	Accuracy	Precision	Sensitivity
1	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%

Table 5.14: *Result of JavaNNS on doubly aligned sequences 80%*

Runs	Accuracy	Precision	Sensitivity
1	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%

In other words, changing from a leave one out testing strategy to a standard training-testing method using 80%-20% and 90%-10% training-test ratios has led to perfect classification. Comparison between the classification of the initial non-aligned signatures and that of doubly aligned signatures showed improvement: 91.6% average accuracy for unaligned, 100% average accuracy for doubly aligned sequences. It should be noted that a median output value was used to determine the class of a test sample. That is, the median output value obtained from the training set was used as a threshold below which a test sample was classified as 0 (worm) and above which a test sample was classified as 1 (virus). The median values for Tables 5.17, 5.18, 5.20 and 5.21 were 0.50334, 0.50334, 0.635515 and 0.609155 respectively.

5.6.2 EXPERIMENTAL RESULTS OBTAINED FROM T-COFFEE AND NEURAL NETWORK ON 30 VIRUSES AND 30 WORMS

In the first experiment, the doubly aligned signature sequences were in turn converted into decimal ASCII code (‘A’ became 66, ‘C’ 67…’Z’ 90) for input to a two-layer perceptron for checking accuracy of classification between worm and virus signatures. Instead of using ASCII, the 18 residues used for representing the signatures and the gaps introduced by multiple alignment were converted into numerical values for an ANNs through real numbers 0.1 to 0.95 in steps of 0.05 (table 5.4) For instance, the first four hexadecimal characters of the 1c.Tanga.a virus ‘8e5e’ would be ‘IRFR’ in amino acid representation and then ’0.45 0.8 0.3 0.8’ when encoded in numeric. The same parameters for the first set of experiments were used again. The aim of this experiment is to explore the implications of adopting different residue representations when forming alignment of signature and extracting motifs.

Table 5.16 provides the results of the best five runs and demonstrates that, when all samples are used for training with no testing, the non-aligned results (Table 5.15) are better in terms of convergence to 0 error. η is the back-propagated maximum step width and RO indicates random order of presentation of samples.

Each of architectures was run 10 times and the results of the best five runs are averaged. Bold font indicates the best overall architecture.

Table 5.15: *Results obtain from JavaNNS on all original worm and virus samples (72 residue characters), without alignment (training only, no testing).*

Structure	**Parameter**			**Sum of squared error**	**Sum of square root error**
	η	**Cycles**	**RO**		
72x1	0.002	10000	no	0.069199	0.22555
72x1	0.001	10000	yes	0.063733	0.216617
72x1	0.0001	10000	yes	0.054272	0.181655
72x1	0.002	10000	Yes	0.050805	0.199491
72x1	**0.001**	**10000**	**no**	**0.004711**	**0.052541**

The results obtained from step 5 are as presented in Table 5.16, where it can be seen that the sum of squared error decreased in comparison to results obtained for non-aligned (Table 5.15). The top five results obtained for doubly aligned sequences have an average error of 12.75%, roughly half the error of 24.27% for original fixed length sequences.

Table 5.16: *Results obtained from JavaNNS on doubly aligned sequences, training only.*

Structure	**Parameter**			**Sum of squared error**	**Sum of square root error**
	η	**Cycles**	**RO**		
581x1	0.002	10000	Yes	0.045381	0.172294
581x1	0.0005	10000	no	0.044893	0.17894
581x1	0.001	10000	Yes	0.016727	0.103227
581x1	0.001	10000	No	0.011038	0.082959
581x1	**0.002**	**10000**	**Yes**	**0.009438**	**0.073852**

Tables 5.15 and 5.16 provide results comparable to those in Tables 5.3 and Table 5.4 but for different numeric representation. Both sum of squared error and sum of square root error clear as shown that the numeric conversion (Table 5.4) was also used successfully and will be used in later experiment.

Finally, as part of Step 5, the doubly aligned sequences (each sample 581 characters long, hence 581 attributes plus a class value) were entered into J48 in WEKA using multiple folds with cross validation. Non-aligned viruses and worms input to WEKA to provide a benchmark (Step 0) led to the results presented in Table 5.17. Without any cross-validation, J48 correctly classified 85% of the sequences. Using 15%, 20%, and 30% cross-validation resulted in an average of 63% accuracy (Table 5.18) for the non-aligned sequences (Step 0).

Table 5.17: Results obtain from J48 in WEKA on original sequences, without alignment

Method	**Parameter**	**Correctly classified**	**Incorrectly classified**
J48	Training	85.00%	15.00%
J48	Cross-V 15	68.33%	31.67%
J48	Cross-V 20	60.00%	40.00%
J48	Cross-V 30	70.00%	30.00%

Comparison between the classification of the initial non-aligned signatures (table 5.17) and doubly aligned signatures (table 5.18) showed an improvement of 11.76% average accuracy after double alignment in Training only dataset. Also, there was a 1.67% improvement using 15% cross validation and 3.33% accuracy improvement using 20% cross validation.

Table 5.18: Results of J48 in WEKA on doubly aligned sequences

Method	**Parameter**	**Correctly classified**	**Incorrectly classified**
J48	Training	96.67%	3.33%
J48	Cross-V 5	78.33%	21.67%
J48	Cross-V 10	66.67%	33.33%
J48	Cross-V 15	70.00%	30.00%
J48	Cross-V 20	63.33%	36.67%

Table 5.19: Test with 60 proteins plus one random sequence

Method	Parameter	Correctly classified	Incorrectly classified
J48	Training	85%	15%
J48	Cross-V 15	63.33%	36.67%
J48	Cross-V 20	58.33%	41.67%
J48	Cross-V 30	66.67%	33.33%

With regard to whether the alphabetic coding makes a difference, we ran one experiment where we included a random sequence made up of any combination of amino acid sequences. When this random sequence was incorporated into the 60 signatures and doubly aligned with those signatures, the random sequence was the only sequence that J48 could not classify. Section 5.6.6 examines the relationship between random sequences and actual signatures in more detail.

One of the rule sets produced by WEKA is (where ‘V’ is virus and ‘W’ is worm):

If pos27 = C then V	If pos55 = A then W
If pos66 = Q then V	If pos261 = Q then W
If pos150 = D then V	If pos3 = L then W
If pos450 = Q then V	If pos11 = Q then W
If pos8 = H then V	

If the characters are order by their position number and the amino acids are mapped back to hexadecimal, the sequence…7...2…d...3...d…will identify a virus signature and the sequence …0…d...1...d…will identify a worm signature, where '...' stands for 'any number of any amino acid'.

It is unlikely that one rule by itself will be sufficient for separating viruses from worms, especially since each position represents only a fragment of code. Nevertheless, such rules can be used as a basis for generating malware 'meta-signatures'.

5.6.3 RELATION BETWEEN NUMBER OF EXPERIMENTS' SIGNAUTURES AND ALIGNMENT LENGTH

In the third set of experiments, we selected new 10, 20, 30, 60 and 70 virus signatures and the same number of worm signatures to check for the effects of alignment on aligned sequence length.

Earlier work (experiments presented in Section 5.6.1 and Section 5.6.2) reported the results of a pruning-based decision tree constructor (J48) for motif extraction, but with only 30 examples of virus signatures and 30 examples of worm signatures. It is important to confirm length increasing of malware signature after their alignment. Length increases will be added to the computational burden of the machine.

The virus and worm signatures were aligned. After alignment, the length of virus and worm signatures were as presented in Table 5.20

Table 5.20: the virus and worm signatures were aligned separately

Number of Instances	Worm length	Virus length
10	118	118
20	142	134
30	161	177
60	166	179
70	166	177

Table 5.21: Comparing double alignment length

Number of instances	Length
20	222
40	294
60	550
120	987
140	680

Fig. 5.4 presents the results of number of samples against aligned signature length.

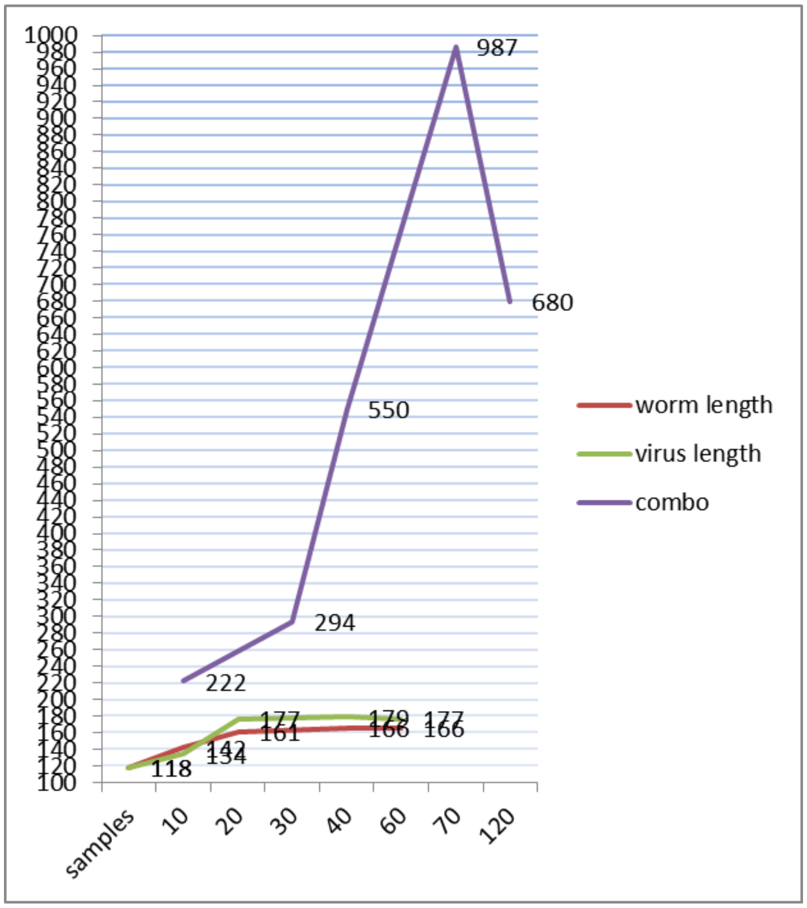

Figure 5.3: Illustrate the examples of worms and viruses alignment situation

Red colour is instances of aligned worms, green colour present instances of aligned computer viruses and purple colour is instances of computer viruses and worms after doubly alignment.

5.6.4 EXPERIMENTAL RESULTS OBTAINED FROM T-COFFEE, BLOSUM AND NEURAL NETWORK IN 3 DIFFERENT REPRESENTATIONS USED ON 30 VIRUS SIGNATURES AND 30 WORM SIGNATURES

The objectives of this experiment are to explore the implications of adopting or not adopting predefined substitution matrices using different residue representations for signatures as well as using a greater variety of ML tools for evaluating the effects of different representation on classification and prediction. The first (Chapter 5.6.1) and second (Chapter 5.6.2) sets of experiments produced results from a pruning-based decision tree constructor (J48) that extracted ‘meta-signatures’ for distinguishing virus from worm signatures but with only 30 doubly aligned virus and 30 doubly aligned worm signatures. It is important to know whether these meta-signatures are an accidental by-product of the representations used or are the evidence of a deeper and unpredicted aspect of applying bio-sequence techniques to virus and worm signatures

The multiple alignments of worms and viruses (steps (a), (b) and (c) above) resulted in fixed-length sequences of 987 residues, 481 residues and 569 residues for R1, R2 and R3(represented in Table 5.2) respectively without BLOSUM matrix. When switching on the BLOSUM matrix, the length of sequences became 981 residues, 1101 residues and 1248 residues for R1, R2 and R3 respectively (Table 5.22). The BLOSUM matrices (BLOcks of amino acid SUbstitution Matrix) are a substitution matrices used for sequence alignment of proteins. BLOSUM matrices are used to score alignments between evolutionarily divergent protein sequences. They are based on local alignments. BLOSUM matrices were first introduced in a paper by Henikoff and Henikoff in 1992 (Henikoff and Henikoff, 1992).

Table 5.22: Comparing alignment length within or without BLOSUM

Sequences	Lengths without matrices	Lengths with BLOSUM
VIRUS–R1	179	223
VIRUS-R2	155	193
VIRUS-R3	171	212
WORM R1	166	194
WORM R2	157	202

WORM R3	170	233
DOUBLY R1	987	981
DOUBLY R2	481	1101
DOUBLY R3	569	1284

Firstly, these three datasets were converted into numerical form using the coding in Table 5.4 and were input to a 987x72x1, 481x72x1 and 569x72x1 perceptron respectively (without alignment, step (f) above). Initially all samples were input to check the architecture, learning rate and estimate of cycles required. Training only accuracy of 100% ((e)(f) above) was achieved using a constant learning rate of 0.1 and standard back-propagation after 200-400 cycles for each of PR1, PR2 and PR3 (using 0.2 as the output classification threshold). The ANN experiments were then repeated for two folds using a 75% training, 25% testing regime, resulting in the figures presented in Table 5.23

Table 5.23: Results obtained from JavaNNS on unaligned sequences, with η=0.1, random order

Training: testing	Accuracy (%)	Sensitivity (%)	Specificity (%)
R1 –all	99.17	98.36	100.00
R1 – 75%Average	46.67	47.35	45.74
0.5	46.67	47.68	45.28
R2 – all	100.00	100.00	100.00
R2 – 75%Average	50.00	49.36	51.02
0.5	49.17	48.39	50.28

R3 – all	100.00	100.00	100.00
R3 – 75%Average	62.50	61.59	63.89
0.5	60.83	60.08	62.14

These three datasets were then converted into numerical form using the coding in Table 5.4 and were input to a 987x72x1, 481x72x1 and 569x72x1 perceptron, respectively (step (d) above). Initially all samples were input to check the architecture, learning rate and estimate of cycles required. Training only accuracy of 100% ((e)(f) above) was achieved using a constant learning rate of 0.1 and standard back-propagation after 200-400 cycles for each of DT60R1, DT60R2 and DT60R3 (using 0.2 as the output classification threshold). The ANNs experiments were then repeated for two folds using a 75% training, 25% testing regime, resulting in the figures displayed in Table 5.24

Table 5.24: Results obtained from JavaNNS on doubly aligned sequences, with η=0.1, RO

Training : Testing	Accuracy (%)	Sensitivity (%)	Specificity (%)
R1 –all training	100.00	100.00	100.00
R1 – 75%Average	75.00	75.95	74.17
0.5	75.00	75.06	75.06
R2 – all training	100.00	100.00	100.00
R2 – 75%Average	85.83	85.53	88.27
0.5	86.67	87.29	88.60
R3 – all training	100.00	100.00	100.00

R3 – 75%Average	78.33	78.84	78.19
0.5	75.00	77.41	74.13

These three datasets were converted into numerical form input using the coding in Table 5.4 and were input to a 1180x72x1, 1038x72x1 and 1198x72x1 perceptron, respectively (step (d) above). Initially all samples were input to check the architecture, learning rate and estimate of cycles required. Training only accuracy of 100% ((e)(f) above) was achieved using a constant learning rate of 0.1 and standard back-propagation after 200-400 cycles for each of DT60R1, DT60R2 and DT60R3 (using 0.5 as the output classification threshold). The ANN experiment were then repeated for four folds using a 75% training, 25% testing regime, resulting in the figures displayed in Table 5.25, which shown that the R1 performance was the best.

Table 5.25: Results obtained from JavaNNS on doubly aligned sequences, with BLOSUM η=0.1, shuffle: yes

Training: Testing	Accuracy (%)	Sensitivity (%)	Specificity (%)
R1 –all training	100.00	100.00	100.00
R1 – 75%Average	84.17	83.80	84.95
0.5	83.33	82.66	84.79
R2 – all training	75.83	100	67.42
R2 – 75%Average	75.00	74.83	75.71
0.5	79.17	80.06	82.24

R3 – all training	100.00	100.00	100.00
R3 – 75% Average	75.83	75.19	77.23
0.5	75.83	75.18	77.73

The three datasets matrixR1, matrixR2 and matrixR3 were then input to PRISM within WEKA. Using no testing or cross-validation, 100% training accuracy was reported (similar to DT60R1-DT60R3 with no testing). With 10-fold cross-validation, the results presented in Table 5.26 were produced, confirming the results of the perceptron that R1 was best in terms of performance, this time using PRISM.

Table 5.26: Results obtained from WEKA PRISM 10-fold cross-validation

	R1	**R2**	**R3**
Accuracy	91.30%	50.00%	52.00%
Sensitivity	91.70%	33.0%3	50.00%
Specificity	91.00%	53.00%	53.30%

The rules produced by PRISM for R1 with no cross-validation (to maximise the extraction of knowledge) and with the removal of W and Y (gap) residues are as follows (with the positions re-ordered from left to right):

Virus if pos41 = M, pos86 = F, pos142 = R, pos339 = S, pos340 = S, pos432 = C, pos447 = S, pos589 = A, pos601 = F, pos666 = C.

Worm if pos32 = H, pos34 = G or H, pos73 = Q or L, pos81 = Q, pos115 = L , pos123 = M, pos137 = M, pos146 = G, pos152 = H, pos352 = D, pos407 = R, pos627 = I.

Converting these R1 residues back to hexadecimal code using Table 5.1 gives us the following possible signatures/motifs: '...a...5...e…f...f...2…f...1...5...2...' for virus and '…7...[67]...[d0]..0...a...a…6...7...3...e...8...' for worm (where '…' stands for 'any number of any amino acid' and square parentheses indicated alternatives.). Earlier work (experiments in Section 5.6.1 and Section 5.6.2) on 30 worms and 30 viruses using R1 reported different signatures: '...a...6...1…6...' for worms and '...9…5...6…4...6...' for virus, but the length of doubly aligned sequences were only 581 (as opposed to 987 here). Clearly, increasing the number of samples has also resulted in increased fixed length after double alignment.

5.6.5 EXPERIMENTS ON 60 VIRUSES AND 60 WORMS USING DIFFERENT ALIGNMENT METHODS AND ANALYSIS ALGORITHMS IN 3 DIFFERENT REPRESENTATIONS

Figure 5.4 illustrates the longest length of signatures, when we use 60 virus signatures and 60 worm signatures after double alignment. In the fifth set of experiments, we select 60 viruses and 60 worms. The multiple alignments of worms and viruses (steps (a)-(d) above) resulted in fixed length sequences varying in length as shown in Table 5.27.

Table 5.27: The effect on length of viral and worm signatures by representation (R), method (Clustal or T-Coffee) and substution matrix (Identity, Gonnet, BLOSUM)

datasets	**ClustalW with ID**	**ClustalW with Gonnet**	**ClustalW with BLOSUM**	**T-coffee with BLOSUM**
VIRUS–R1	121	91	81	223
VIRUS-R2	116	85	79	193
VIRUS-R3	140	85	79	212
WORM-R1	129	90	87	194

WORM-R2	121	87	87	202
WORM-R3	116	89	84	233
Double alignment R1	494	140	103	981
Double alignment R2	440	123	106	1101
Double alignment R3	462	128	86	1284

For each perceptron there were as many input neurons as the length of the sequence, 72 hidden nodes and one output node for the class virus (0) or class worm (1). A fixed learning rate of 0.2 and momentum of 0.1 were used for all 12 perceptrons for 200 epochs. The results obtained from MLP algorithms, under the two test conditions, are presented in Table 5.28. Accuracy results (Table 5.28) show only one difference between the representations and learning algorithms, which is between R1 and R2/R3 when using T-coffee alignment with BLOSUM matrix and between R3 and R1/R2 which using ClustalW alignment with unitary matrix. After ClustalW alignment with unitary matrix alignment, R1 and R2 representations got low rate for correctly classified in Multilayer Perceptron. On the other hand, they all got high rate for correctly classfied by ClustalW with Gonnet matrices.

*Table 5.28: Accuracy figures for R1, R2 and R3 representations across five machine learning algorithms after four different methods of alignment CI, CG, CB and TB. '66/34' and '90/10' refer to the training, testing and cross-validation regimes used***.**

		Naïve Bayes		J48		LAD Tree		OneR		Perceptron		Average		Overall average
		66/34	90/10	66/34	90/10	66/34	90/10	66/34	90/10	66/34	90/10	66/34	90/10	
	Original unaligned	0.26	0.43	0.54	0.54	0.51	0.63	0.49	0.58	0.39	0.48	0.44	0.53	0.49
	column index	a	b	c	d	e	f	g	h	i	j	k	l	m
row index	**R1**													
1	CI	0.49	0.60	0.49	0.77	0.66	0.72	0.56	0.59	0.51	0.54	0.54	0.64	0.59
2	CG	0.95	0.98	0.88	0.83	0.93	0.91	0.78	0.81	0.95	0.97	0.90	0.90	0.90
3	CB	0.90	0.91	0.90	0.82	0.90	0.85	0.81	0.85	0.83	0.79	0.87	0.84	0.86
4	TB	0.71	0.90	0.73	0.82	0.93	0.91	0.78	0.80	0.85	0.95	0.80	0.88	0.84
5	*Average R1*	0.76	0.85	0.75	0.81	0.85	0.85	0.73	0.76	0.79	0.81	0.78	0.82	0.80
	R2													
6	CI	0.63	0.64	0.73	0.65	0.71	0.72	0.51	0.64	0.71	0.58	0.66	0.65	0.65
7	CG	0.95	0.97	0.98	0.88	1.00	0.98	0.88	0.97	0.98	0.96	0.96	0.95	0.95
8	CB	1.00	0.98	1.00	0.95	1.00	0.99	0.98	0.98	0.88	0.89	0.97	0.96	0.96
9	TB	0.44	0.78	0.85	0.78	0.73	0.71	0.61	0.56	0.85	0.86	0.70	0.74	0.72
10	*Average R2*	0.76	0.84	0.89	0.82	0.86	0.85	0.74	0.79	0.85	0.82	0.82	0.82	0.82
	R3													
11	CI	0.98	0.90	1.00	0.98	0.98	0.92	0.90	0.87	0.95	0.93	0.96	0.92	0.94
12	CG	1.00	0.99	0.95	0.96	0.98	0.99	0.95	0.97	0.98	0.98	0.97	0.98	0.97
13	CB	0.93	0.98	0.90	0.97	0.98	0.99	0.98	0.99	0.90	0.96	0.94	0.98	0.96
14	TB	0.59	0.87	0.78	0.80	0.76	0.82	0.76	0.71	1.00	0.94	0.78	0.83	0.80
15	*Average R3*	0.87	0.94	0.91	0.93	0.92	0.93	0.90	0.89	0.96	0.95	0.91	0.93	0.92
16	*Average ML method*	0.80	0.88	0.85	0.85	0.88	0.88	0.79	0.81	0.87	0.86	0.84	0.86	0.85

CI was on the whole the worst performing alignment method. This may seem surprising, given that CI is the one alignment technique that does not use biologically plausible substitution matrices and only inserts gaps to help identical amino acids in different sequences to align in columns. However, identity matrices will work best when there is significant commonality between sequences. Their relatively poor performance here indicates that the sequences do not share much commonality.

Table 1: Averaged alignment accuracy figures of Table 5.28. The underlined figures show the significant differences found through ANOVA, with figures in bold indicating the entries that these underlined figures differed significantly from.

Alignment	CI(%)	CG(%)	CB(%)	TB(%)
NB (66/34)	70	97	94	58
NB (90/10)	71	98	96	85
J48(66/34)	74	94	93	79
J48(90/10)	80	89	91	80
LAD (66/34)	78	97	96	81
LAD (90/10)	79	96	94	81
OneR (66/34)	66	87	92	72
OneR (90/10)	70	92	94	69
P (66/34)	72	97	87	90
P (90/10)	68	97	88	92
Average(66/34)	72	94	93	76
Average(90/10)	74	94	93	82
Combine overall average	73	94	93	79

✧ CI: ClustalW with Identity; CG: ClustalW with Gonnet matrices; CB: ClustalW with Blosum matrices; TB: T-Coffee with Blosum matrices

Overall, R3 performed best (0.92 overall accuracy, row 15, column m of Table 5.28) across all five machine learning techniques under both test conditions, irrespective of alignment method. ClustalW with Gonnet was the best alignment method (0.94) across all ML techniques, irrespective of representation (Table 5.28).

All accuracy results using aligned data were individually better than the benchmark accuracy for the non-aligned datasets. The overall accuracy of 85% (bottom right of Table 5.28) is a major improvement on the 49% accuracy for the benchmarked (non-aligned) data, demonstrating that the use of sequence alignment techniques can serve a useful ML purpose.

To complete the first set of results, the best sequences of the three representations, as given by the overall accuracy figures in Table 5.26 were input to the rule extractor, PRISM. R1 worked best with ClustalW using Gonnet (0.9, row 2), R2 with ClustalW using BLOSUM (0.96, row 8) and R3 with ClustalW using Gonnet (0.97, row 12). All 120 doubly aligned sequences were used for rule extraction (i.e. no training and testing) to maximise knowledge extraction. For R1 CG sequences, the following rules were found (all W and Y gap representations are excluded; 'pos' refers to 'position' in the sequence):

Virus: If pos36 = A, pos21 = D, pos28 = E, pos53 = A, pos20 = N, pos5 = A, pos30 = L, pos32 = A, pos36 = P.

Worm: If pos72 = L, pos73 = P, 51 = H, pos59 = S, pos70 = R, pos73 = D, pos46 = R, pos72 = M, pos71 = S, pos44 = I, pos45 = L, pos70 = G, pos10 = L, pos41 = C, pos45 = D, pos54 = L.

Rewritten as a left to right amino acid sequence using the positional information, these rules produce the strings *..A..ND..E..L..A..[AP]..A..* for virus and *..L..C..I[LD]R..H..L..S..[GR]S[LM][DP]..* for worm, where '..' stands for 'any number of any amino acid' and square parentheses indicates alternatives. Converting these amino acids sequences back to their R1 hexadecimal equivalents using Table 5.1 and removing the gaps produce the meta-signatures '1b3401[1c]1' for the virus and '028[03]e70f[6e]f[0a][3c]' for the worm.

For R2 CB, rules with only gaps except for 'worm if pos21=R' were returned. For R3 CG, the following rules were found by PRISM:

Virus: If pos65 = F, pos12 = A, pos13 = A.

Worm: If pos124 = M, pos122 = A, pos124 = H, pos119 = N, pos55 = M, pos88 = M, pos11 = A, pos33 = I, pos55 = L.

Rewritten as amino acid, left to right sequences, these rules produce *..AA..F..* for virus and *..A..I..[LM]..M..N..A..[HM]..* for worm, which in turn produce the hexadecimal meta-signatures '114' for virus and '17[90]0a1[60]' for worm.

In other words, these meta-signatures represent those parts of the virus and worm signatures that are conserved between variants and families of virus and worm signatures as well as those parts that distinguish viral signatures from worm signatures, using R1 and R3 as the representation methods, ClustalW as the alignment method and Gonnet as the substitution matrix used for both phases of alignment within ClustalW. These meta-signatures could prove useful as byte patterns for matching probabilistically and contiguously against incoming packets to determine whether these packets contain possible malware belonging to the families of virus and worm analyzed here, with subsequent analysis then being required to determine exactly which virus or worm may be involved. Signature writers may find these meta-signatures useful for identifying polymorphic and metamorphic viruses if these hexadecimal codes are mapped back to original malware hexadecimal code.

5.6.6 RANDOM SEQUENCE EXPERIMENTS

The objectives of this experiment are to validate all research methods in Part I. 60 random sequences replace the signatures of viruses and worms in this set of experiments. As can be seen by the overall accuracy figures in Table 5.30. Of all sequences that were input to the rule extractor, PRISM, R1 worked best. These random sequences will still be represented by R1 (table 5.2) which will be aligned by ClustalW with identity matrix as well as used for rule extraction to evaluate the effects on classification and prediction.

Table 5.30: Accuracy of PRIMS in WEKA with different lengths of random sequences

training	**10**	**20**	**40**	**60**	**72**	**98**
50/50	0.48	0.52	0.97	0.58	0.92	0.85
66/34	0.48	0.52	0.97	0.58	0.92	0.85
90/10	0.48	0.52	0.97	0.58	0.92	0.85

- The results of accuracy are for these sequences after double alignment.

For the original random sequence length equal to 10 residues, after alignment by CI, the following rules were found (all W and Y gap representations are excluded; ‘pos’ refers to ‘position’ in the sequence):

Virus: If pos11 = N, If pos5 = I, If pos5 = P, If pos6 = M, If pos4 = H, If pos7 = C, If pos4 = R, If pos5 = M, If pos10 = L, If pos10 = Q, If pos12 = I, If pos16 = E.

Worm: If pos5 = C, If pos5 = N, If pos5 = E, If pos14 = H, If pos3 = L, If pos3 = M, If pos7 = D, If pos7 = E, If pos7 = R, If pos14 = E, If pos3 = F, If pos11 = G, If pos11 = L, If pos6 = R

For original random sequence length equal to 20 residues, after alignment by CI, the following rules were found:

Virus: If pos11 = R, If pos21 = N, If pos27 = R, If pos9 = H, If pos30 = F, If pos31 = H, If pos1 = D, If pos17 = F, If pos27 = H, If pos11 = H.

Worm: If pos12 = G, If pos20 = Q, If pos10 = D, If pos18 = R.

For original random sequence length equal to 40 residues, after alignment by CI, rules with only gaps were returned. For original random sequence length equal to 60 residues, after alignment by CI, the following rules were found:

Virus: If pos26 = P, If pos35 = G, If pos28 = H, If pos87 = N, If pos19 = S, If pos24 = F, If pos15 = D, If pos18 = I, If pos36 = K.

Worm: If pos8 = M, If pos17 = G.

For original random sequence length equal to 72 residues, after alignment by CI, the following rules were found:

Virus: If pos8 = F.

Worm: If pos12 = M, If pos16 = C.

For original random sequence length equal to 98 residues, after alignment by CI, rules with only gaps except for 'virus if pos101 = Q and worm if pos62 = K' were returned.

Table 5.31: Rewritten above rules as amino acid, with random sequences, as following:

Original random length	Doubly aligned sequence's length	Rules produce for virus	Rules produce for worm
10	19	…[HR][IMP]MC…[LQ]NI…E	…[FLM]…[CNE]R…[DER]…[GL]…[EH]
20	38	D…H…[HR]…F…N…[HR]…FH…	…D…G…R…Q
40	62	NILL	NILL
60	95	…D…IS…F…P…H…GK…N…	…M…G
72	95	F	…M…C…
98	126	Q	K

Table 5.30 shows that the classification accuracy of the random sequences still varies widely after alignment. Table 5.31 illustrates shows that with the length of random sequences increasing, the amount of obtained information decreases. Random sequences cannot get any benefit from machine learning after alignment. In other words, these meta-signatures represent those parts of the virus and worms signatures that are valuable.

5.7 CONCLUSION

The results of the experiments show the effectiveness of a bio-inspired data mining method for dealing with problematic ML data. It has been clearly demonstrated that the method of converting malware hexadecimal signatures to amino acid representations affects learning and therefore the signatures extracted.

In the next chapter, we will show that once samples are represented as bio-sequences, protein modeling may offer new ways of interpreting the data and rules extracted by ML.

Chapter 6 Methods II and Experiment Results

6.0 INTRODUCTION

The results discussed in the previous chapter above indicate that ClustalW with Gonnet (CG) produces the most accurate classification results (0.94 in Table 28) as well as most interpretable and useful meta-signatures for data mining purposes, especially when R1 is used as the representation method. The meta-signatures produced by PRISM using CG, namely, ‘1b3401[1c]1’ for R1 virus and ‘028[03]e70f[6e]f[0a][3c]’ for R1 worm, and ‘114’ for R3 virus and ‘17[90]0a1[60]’ for R3 worm, should ideally be interpreted by tracing back to the op codes in the original source code from which the hexadecimal signatures were initially derived. But the malware source code is not available in this case (due to security concerns regarding the public dissemination of malware code) and may not be available for other datasets where there has been a non-reversible transformation or conversion from the original data to that used for data mining. The aim of the second set of experiments is to determine the feasibility of interpreting these meta-signatures by reference to the biological domain.

The linear representation of sequences represents the primary structure of the protein. The secondary structure of an amino acid sequence provides an indication of how segments can form a 3D structure through local interactions. The basic secondary structures are ‘helix’, ‘coil’ , ‘sheet’ and ‘beta turn’, and can be calculated from the hydrogen bonds between amino acids and carboxyl groups. Secondary structure gives rise to tertiary structure, which is the 3D representation of the entire sequence determined by atomic coordinates. Many sources now exist that describe the relationships between these three structures and the problems associated with computational prediction of secondary and tertiary structure from the primary sequence. Finally, there is quaternary structure, which is 3D representation of a complex protein structure consisting of two or more functionally distinct subsequences.

6.1 SEQUENCE STRUCTURE OF BIO-INFORMATICS 3D VIEW

One way to check for interpretability is to see whether a particular representation, after alignment, has biological plausibility in terms of matches with real protein sequences. That is, the 'most plausible representation' can be hypothesized to be the one that maps signatures of a class, after initial alignment to identify commonalities across families within that class, to the most biological analogues, whereas the least plausible representation' is one that maps to few biological analogues. Using a representation that has more biological plausibility may, in turn, be reflected in improved alignment due to the use of biologically based substitution matrices that reflect known frequencies of mutations and conserved subsequences of residues in existing protein sequences. A set of experiments discussed in this chapter takes each of the 120 sequences in the singly aligned datasets using all three representations and entering them, one by one, into the PRINTS database as queries to see which of the existing known proteins are returned as the closest match between the artificial, aligned polypeptide sequence and naturally occurring polypeptide sequences. PRINTS (Attwood ,Beck, Bleasby & Parry-Smith. 1994; Attwood, Coletta, Muirhead, Pavlopoulou, Philippou, Popov, Roma-Mateo, Theodosiou & Mitchell. 2012) consist of conserved motifs (called 'fingerprints') found in already aligned proteins with known function and structure. Newly aligned sequences can be matched against these fingerprints to tentatively assign these sequences to known families of protein and hence predict the function and structure of these new sequences. The number of hits in the first experiment will help us determine the most plausible representation as given by the highest number of hits. To provide a benchmark, the original, unaligned sequences in all three representations were also input to PRINTS to see if alignment produced better results. These experiments investigate the three primary structure representations (R1-R3) produced by the four alignment methods in more detail.

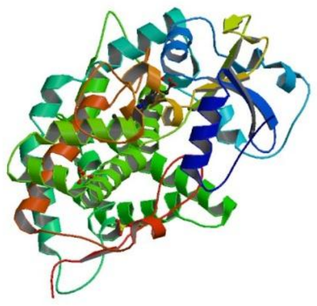

Figure 6.1: Biological assembly image for 1 GWI which found d in a non-aligned virus Virus.Acad.Bursted.a, which was searched by tool Motif3D

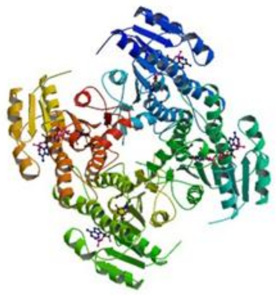

Figure 6.2: Biological assembly image for 1 GWI and 1CYC which found d in doubly alignment for Virus.Acad.Bursted.a, which was searched by tool Motif3D

In summary, the experimental method for the second set of experiments is as follows:

Step (k): Check the sequences from step (a) against naturally occurring proteins with 3D view

Take the six non-aligned datasets VR1-VR3 and WR1-WR3 and find the number of hits against known proteins using each sequence as a fingerprint. Repeat for the singly aligned versions of these datasets, i.e. SAVCIR1-R3, SAWCIR1-R3, SAVCGR1-R3, SAWCGR1-R3, SAVCBR1-R3, SAWCBR1-R3, SAVTBR1-R3, SAWTBR1-R3 (see Table 5.2, Appendix I for a description of these datasets). Compare the results.

6.2 EXPERIMENT RESULTS

Table 6.1 describes the number of hits by sequence and in total for each of the alignment methods, including non-alignment as a benchmark. As can be seen, TB provides the least biological plausibility (only one hit) and CG the most in terms of total hits against proteins (368) and the most number of different proteins found (349). CI was less plausible than the benchmarked non-aligned sequences in terms of hits on all three measures.

Table 6.1: The number of hits against existing proteins using PRINTS before and after the first alignment.

		R1-virus	**R1-worm**	**R2-virus**	**R2-worm**	**R3-virus**	**R3-worm**	**Total**
Non-aligned	SC	15	15	18	12	9	17	86
	TC	39	37	42	29	18	32	197
	PC	39	37	42	29	18	32	197
CI	SC	14	8	11	11	4	9	57
	TC	51	22	22	32	9	19	155
	PC	51	22	22	32	9	19	155
CG	SC	24	12	29	16	22	16	119
	TC	65	32	93	41	83	54	368
	PC	63	32	85	41	82	46	349
CB	SC	23	17	22	18	17	12	109
	TC	68	47	64	46	48	46	319
	PC	65	47	64	46	48	44	314
TB	SC	0	0	0	0	1	0	1
	TC	0	0	0	0	1	0	1
	PC	0	0	0	0	1	0	1

The rows represent the 60 signatures represented using R1-R3. The columns represent the alignment method. 'SC' provides the total number of sequences out of 60 in that particular dataset that were matched against existing proteins; 'TC' provides the total number of proteins matched by all sequences in that dataset; 'PC' provides the total number of different proteins matched by all sequences in that dataset.

CG showed a small but interesting reduction in the number of different proteins found in comparison to total proteins, indicating some common hits against the same protein by different sequences belonging to R1 virus (from 65 to 63), R2 virus (from 93 to 85), R3 virus (from 83 to 82) and R3 worm (from 54 to 46). Similarly, CB showed some small reductions for R1 virus (from 68 to 65) and R3 worm (from 46 to 44). These common hits were examined in more detail.

For CB R1 virus, the common proteins were (using PDB fingerprint ID codes) 1EWV (twice), 1KYO (twice) and 1KB9 (twice). 1EWV,1KYO and 1KYO all refer to ubiquinol-cytochrome reductase complex core proteins found in *S. cerevisiae* (bc1 complex) and play a key part in energy conversion of the respiratory and photosynthetic electron transfer chains (Hunte, Koepke, Lange, Rossmanith & Michel., 2000). For CB R3 worm, the common proteins were 1COV (twice) and 1JEW (twice), both of which are parts of the coxsackiervirus. 1COV is part of the coat (Muckelbauer, Kremer, Minor, Tong, Wlotnick, Johnson and Rossmann, 1995) and 1JEW is a receptor (He, Chipman, Howitt, Bator, Whitt, Baker, Kuhn, Anderson, Freimuth & Rossmann, 2001). Coxsackievirus is an Entovirus, one of the most common human pathogens (Guevara, 2007).

For both the CG R1 virus and CG R1 worm, the common proteins were 1PG4 (twice for R1 virus; five times for R1 worm) and 1PG3 (twice for R1 virus; five times for R1 worm). 1PG4 and 1PG3 are both acetyl CoA synthetases from *Salmonella enterica* (Gulick, Starai, Horswill, Homick, & Escalante-Semerena 2003). Synthetases (or ligases) catalyse the joining together of two molecules (Nash and Lindahl, 1996). *S.enterica* is one of the most common causes of illness caused by food infection (Giannella, 1996).

6.3 CONCLUSION

In this chapter, we take each of the 120 sequences in the singly aligned datasets using all three representations and entering them, one by one, into the PRINTS database as queries to see which of the existing known proteins are returned as the closest match between the artificial, aligned polypeptide sequence and naturally occurring polypeptide sequences. Further work will be hanging on whether non-biological data being represented as amino acid sequences confers any benefit from naturally biological sequences.

Conclusion and further work will be discussed in next Chapter 7.

Chapter 7 Conclusion and Further Work

7.0 CONCLUSION

We first benchmarked the ability of neural networks and rule extraction methods via learning to separate worms from viruses without using any multiple sequence alignment (ANN results in Table 5.7, WEKA results in Table 5.16), and without testing. When comparing the initial ANN results in Table 6 (each sample initially 72 characters long) against doubled aligned results in Table 5.8 (each sample now 581 characters long), we found that SSE was actually better for the non-aligned results (Table 5.7). However, when using ANN accuracy, precision and sensitivity measures for testing purposes (Table 5.9 for non-aligned, Table 5.12 for aligned), we found a significantly improved ANN training performance using aligned sequences (Table 5.12) and comparable ANN results using a leave one out testing strategy (Tables 5.7 and Table 5.8). We fixed η to 0.0005 and repeated the ANN experiments using 90/10 and 80/20 training/test ratios. For non-aligned sequences (Tables 5.9, 5.10, 5.11), the overall average ANN accuracy, precision and sensitivity across five runs for 90/10 is 91.12% (Table 5.10) and for 80/20 it is 92.12% (Table 10). However, when the ANN experiments were repeated using double aligned sequences, the figures were 100% (Tables 5.12, 5.13 and 5.14), using the median ANN output value as the class value threshold.

We ran traditional symbolic rule extraction software (Tables 5.17 and Table 5.18). J48 produced 85% classification accuracy using all unaligned samples for testing alone (Table 5.17) and an average 63% accuracy when adopting 95/5, 90/10, 85/15 and 80/20 training/testing ratios. When the doubled aligned sequences were used, training only produced 96.67% accuracy and training/testing produced an average accuracy of 69.5% across the same training/testing ratios (Table 5.18). We converted one of the J48 rules back into a hexadecimal representation for interpretation purposes. Interestingly, hex 6 occurs frequently in this rule to help identify both viruses and worms.

We converted all three PRISM rules back into a hexadecimal representation for interpretation purposes.

Figure 5.8 presents the results of number of samples against aligned signature length. Fewer than 30 samples for each of viruses and worms, the length of alignment increased substantially. When the numbers of samples were over 60, the growth of the aligned signature length slowed down.

We used three different representations R1-R3 in total. The main difference, apart from different amino acids for different hexadecimal code, lies in the use of Z to represent gaps in R1-R3. In the amino acid alphabet ‘Z’ represent either Glutamine (Q) or glutamic acid (E). Substitution matrices will therefore map Z against Q and E

To summarize, the results of the first set of experiments show the feasibility of a bio-inspired data mining method for dealing with problematic ML data. The method of converting malware hexadecimal signatures to amino acid representation has been clearly demonstrated to affect learning and therefore the signatures extracted. Given that there are ${}^{20}P_{16}$ ways to undertake the hex-residue conversion, further experiments are required. In particular, two more representations of the hexadecimal code in the amino acid alphabet were tested to check for the effects of having one amino acid representing gaps as well as possible amino acids (Table 7.1).

Table 7.1: Amino Acid representation II with ‘Z’ replaced by ‘W’ for the first gap

Hexadecimal Code	R4	R5
1	I	K
2	K	I
3	L	H
4	M	G
5	N	F
6	P	E
7	Q	D
8	R	C
9	S	A
0	A	S
a	C	R
b	D	Q
c	E	P
d	F	N
e	G	M
f	J	L
-	Y	Y
-	W	W

This time the alignment method CG was selected, as CG appeared to have most (positive) effect on average classification accuracies in comparison within four methods of alignment, CI, CG, CB and TB (table 5.28 above). The overall results for symbolic algorithms and the perceptron, under 50:50 condition with same 10-fold cross validation were also presented.

Table 7.2: Accuracy figures for R1, R2, R3, R4 and R5 representations across 3 machine learning algorithms after 1 method of alignment CG,. '50/50' refer to the training, testing and cross-validation regimes used.

ML			**Unaligned**					**Aligned**		
Perceptrons	R1	R2	R3	R4	R5	R1	R2	R3	R4	R5
Accuracy	0.517	0.533	0.533	0.608	0.617	0.967	0.967	0.975	**1.000**	0.983
Sensitivity	0.516	0.540	0.542	0.607	0.613	0.983	0.967	0.967	**1.000**	0.968
Specificity	0.517	0.529	0.528	0.617	0.633	0.950	0.967	0.983	**1.000**	1.000
J48	R1	R2	R3	R4	R5	R1	R2	R3	R4	R5
Accuracy	0.483	0.508	0.558	0.542	0.542	0.825	0.883	0.958	0.883	**0.975**
Sensitivity	0.483	0.508	0.554	0.541	0.541	0.783	0.871	**0.982**	0.848	0.967
Specificity	0.483	0.509	0.564	0.550	0.550	0.900	0.900	0.933	0.933	**0.983**
NaiveBayes	R1	R2	R3	R4	R5	R1	R2	R3	R4	R5
Accuracy	0.425	0.475	0.533	0.542	0.542	0.975	0.967	0.992	**1.000**	0.983
Sensitivity	0.426	0.476	0.537	0.545	0.545	0.983	0.967	0.984	**1.000**	0.968
Specificity	0.424	0.474	0.530	0.500	0.500	0.967	0.967	1.000	**1.000**	1.000
Summary										
Accuracy	0.475	0.506	0.542	0.564	0.567	0.922	0.939	0.975	0.961	**0.980**
Sensitivity	0.475	0.508	0.544	0.564	0.566	0.916	0.935	**0.978**	0.949	0.968
Specificity	0.475	0.504	0.541	0.556	0.561	0.939	0.945	0.972	0.978	**0.994**

This time, R4 clearly got the highest rate of accuracy within all five representations. Much more work would be required to identify 'optimal' coding schemes taking into account the purposes of data mining.

All accuracy figures for aligned sequences were better than the original benchmark accuracy figures for the non-aligned sequences, indicating that, for this dataset, alignment leads to major improvements in classification accuracy (Table 5.28 and Table 7.2). The disadvantage with DCG120R3 is that it appears to produce the best result of all three representations (Table 5.28) while ML the machine learning results indicate that longer sequences (DCG120R1) lead to greater accuracy and that the rules derived for generating malware meta-signatures provide relatively rich information when in compared to rules for shorter alignment sequences. Even in the latest research, R4 got the best result but the least amount of relative information in PRISM. Moreover, amino acid representation can only code a maximum of 20 different attribute values including gaps. Continuous data or categorical data consisting of more than 18 values (if two separate amino acids are used to represent gaps) may need to be converted to a maximum of 18 different values to allow sequence alignment using amino acids to be used.

Converting the hexadecimal code of viruses and worms to amino acids and then rational numbers between 0 to 1 for input to neural networks has also shown to be effective, provided that the viral and worm sequences are double aligned appropriately (ClustalW with Gonnet, 0.97 accuracy, Table 5.28). The first research question formulated for this story is it possible to use sequencing techniques from bioinformatics to help distinguish between computer virus and worm signatures. It has been answered positively in this particular case.

The results from the second set of experiments show that, once samples are represented as bio-sequences, protein modeling may offer new ways of interpreting the data and rules extracted by ML. If a ‘discriminatory’ representation is required, where the samples are the most dissimilar to naturally occurring proteins, R3 is the best before alignment and any representation using T-Coffee with BLOSUM after alignment (Table 5.28). If a ‘generalizable’ representation is needed, alignment using ClustalW with Gonnet gives the most hits (Table 5.28) for all three representation methods in comparison to the other alignment methods. Both ClustalW with Gonnet and ClustalW with BLOSUM returned hits of aligned sequences against a small number of naturally occurring ‘dangerous’ proteins, i.e. proteins associated with infection. This could be coincidence or it could show some implicit relationship between the substitution matrices used for alignment and their construction from bacterial and viral databases in the past. Further work is required to determine whether more can be made of such implicit relationships or whether some fundamental sharing of structural information between natural infection agents and artificial infection agents is at play here.

7.1 FURTHER WORK

Finally, we are aware that this is the first time that the potential benefit of using bioinformatics sequence alignment techniques for improving the behaviour of malware classifiers has been demonstrated, which contribute observations to the nascent area of malware theory. If such malware structures do in fact share structural and functional relationships with naturally occurring counterparts, biological knowledge on how, for example, to stop adhesion or uncontrolled growth could be adapted to inspire novel theoretical advances in our understanding of how to deal with computer malware detection and prevention.

Reference

Accuracy. In *Wikipedia*. Retrieved 2012 from http://en.wikipedia.org/wiki/Accuracy_and_precision

Amino Acid. In *Wikipedia*. Retrieved 2012 from http://en.wikipedia.org/wiki/Amino_acid

Attwood, T.K., Beck, M.E., Bleasby, A.J. and Parry-Smith, D.J. (1994) PRINTS - A database of protein motif fingerprints. *Nucleic Acids Research*, 22(17), 3590-3596.

Attwood, T.K., Coletta, A., Muirhead, G., Pavlopoulou, A., Philippou, P.B., Popov, I., Roma-Mateo, C., Theodosiou, A. & Mitchell, A. (2012) The PRINTS database: a fine- grained protein sequence annotation and analysis resource - its status in 2012. *Database*, 10.1093/database/base019.

Baldi, P. and Brunak, S. (2001) *Bioinformatics: The Machine Learning Approach*. MIT Press.

Bayoglu, B. and Sogukpinar, I. (2008) Polymorphic worm detection using token-pair signatures. *Proceedings of the 4th International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing* (SecPerU '08), 7-12.

Breiman, L., Friedman, J.H., Olshen, R. and Stone, C. (1983) *Classification and Regression Trees*. Wadsworth.

Cendrowska, J. (1987) PRISM: An algorithm for inducing modular rules, *Journal of Man-Machine Studies*, 27(4), 349-370.

Chen, T. Intrusion detection for viruses and worms (2004) *IEC Annual Review of Communications*, vol. 57.

Chen, Y., Narayanan, A., Pang, S. and Tao, B. Malicious (2012a) Malicious software detection using multiple sequence alignment and data mining. 26th *IEEE International Conference on Advanced Information Networking and Applications* (AINA 2012), Fukuoka, Japan, March 2012, 8-14.

Chen, Y., Narayanan, A., Pang, S. and Tao, B. Malicious (2012b) Multiple sequence alignment and artificial neural networks for malicious software detection. *Proceedings of 8th IEEE Conference on Natural Computation* (ICNC'12), Chonqing, China, May, 2012, 261-265.

Cohen, W. (1995) Fast effective rule induction. *Proceedings of Twelfth International Conference on Machine Learning*, Tahoe City, CA, July 1995, 115-123.

Computer worm. In *Wikipedia*. Retrieved 2012 from http://en.wikipedia.org/wiki/Computer_worm#cite_note-2

Creighton and Thomas H. (1993) Chapter 1. Proteins: Structures and Molecular Properties. *San Francisco*: W.H. Freeman

Dmitry, O. G and Solomon, A. (1995) *Dr. Solomon's Virus Encyclopedia,* Dr. Solomon"s S&S International ISBN 1-897661-00-2

DNA In *Wikipedia*. Retrieved 2012 from http://en.wikipedia.org/wiki/DNA

Dorsam, R.T. and Gutkind, .JS. (2007). G-protein-coupled receptors and cancer. *Nat Rev Cancer* 7 (2): 79–94. doi:10.1038/nrc2069.

Eddy, S.R. (1998). Profile hidden Markov models. *Bioinformatics*, 14, 755-763.

Flower, D.R (1996) The Lipocalin protein family: structure and function. Biochem.J.., 318, 1-14.

Goldhaber, A.S. and Nieto, M.M (2010). Photon and graviton mass limits. *Rev. Mod. Phys*. 82: 939

Gonnet, M., Cohen, A., and Benner, S. (1992). Exhaustive matching of the entire protein sequence database. *Science.* 256 (5062), 1443–1445.

Gulick, A.M., Starai, V.J., Horswill, A.R., Homick, K.M., Escalante-Semerena, J.C. (2003) The 1.75 A crystal structure of acetyl-CoA synthetase bound to adenosine-5'-propylphosphate and coenWyme A. *Biochemistry* 42: 2866-2873.

Hara, T., Sato K. and Ohya, M. (2010) MTRAP: Pairwise sequence alignment algorithm by a new measure based on transition probability between two consecutive pairs of residues. *BMC Bioinformatics*, 11:235. http://www.biomedcentral.com/1471-2105/11/235

He, Y., Chipman, P.R., Howitt, J., Bator, C.M., Whitt, M.A., Baker, T.S., Kuhn, R.J., Anderson, C.W., Freimuth, P., Rossmann, M.G. (2001). Interaction of coxsackievirus B3 with the full length coxsackievirus-adenovirus receptor. *Nat.Struct.Biol*. 8: 874-878.

Henikoff, S. and Henikoff, J. G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl Acad.Sci.* USA, 89, 10915-10919.

Holte, R.C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning.* 11:63-91.

Hunte, C., Koepke, J., Lange, C., Rossmanith, T., Michel, H. (2000) Structure at 2.3 A resolution of the cytochrome bc(1) complex from the yeast Saccharomyces cerevisiae co-crystalliWed with an antibody Fv fragment. *Structure Fold.Des*. 8: 669-684.

Hynes, R. O. (1987) Integrins: a family of cell surface receptors. *Cell* 48 549-554.

Jakobsson, M. and Ramzan, Z., (2008). *Crimeware: Understanding New Attacks and Defenses.* Addison-Wesley Professional; 1 edition

Jiangmin anti-virus warning center, (2007) *$100,000 Offered to Crack Down on Computer Virus.* Orienglish. Retrieved 2012 http://english.cri.cn/2906/2007/01/24/63@188373.htm

Keedwell E.C. and Narayanan A. (2005) *Intelligent Bioinformatics*. Wiley.

Kondrak, G. (2002) *Algorithms for Language Reconstruction*. University of Toronto, Ontario,2002. http://www.cs.ualberta.ca/~kondrak/papers/thesis.pdf.

Kolter, J.Z. and Maloof M.A. (2006). Learning to detect and classify malicious executable in the wild. Journal of Machine Learning Research 7, 2721-2744. Retrieved from http://www.google.co.nz/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=1&cad=rja&ved=0CCwQFjAA&url=http%3A%2F%2Fwww.cs.cmu.edu%2F~zkolter%2Fpubs%2Fkolter-jmlr06.pdf&ei=W3uCUdzfIY_XkgXUwYCQBg&usg=AFQjCNFHxR4c8LsUgXpN8IAg_iY-I92dDg

Landesman, M. (2010) *Boot sector virus repair*. Antivirus About.com, 2012 http://antivirus.about.com/od/securitytips/a/bootsectorvirus.htm

Leydon, J., (2009) Scareware *Mr bigs enjoy 'low risk' crime bonanza*. Retrieved 2012 http://www.theregister.co.uk/2009/10/20/scareware_psychology/

Lipman, D.J. and Pearson, W.R. (1985) Rapid and sensitive protein similarity searches. *Science* , 1985, 227 (4693): 1435–41.

McGhee, S. (2007) *Pairwise Alignment of Metamorphic Computer Viruses*. Masters Project Paper 37, 2007. Faculty of the Department of Computer Science San Jose State University. http://scholarworks.sjsu.edu/etd_projects/37

Microsoft (2006) Frequenctly Asked Questions: Word Macro Viruses", 2012. http://support.microsoft.com/kb/187243/en

Mount, D. M. (2001) *Bioinformatics: Sequence and Genome Analysis*, 3rd , Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press Cold Spring Harbor, NY.

Muckelbauer, J.K., Kremer, M., Minor, I., Tong, L., Wlotnick, A., Johnson, J.E., Rossmann, M.G. (1995) Structure determination of coxsackievirus B3 to 3.5 A resolution. *Acta Crystallogr.*,Sect.D 51: 871-887

Needleman S.B. and Wunsch, C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology* 48 (3), 443–53.

Neural Network. In *Wikipedia*. Retrieved 2012 from http://en.wikipedia.org/wiki/File:Artificial_neural_network.svg

Notredame, C., Higgins, D.G. and Heringa, J. (2000) T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302: 205-217.

Parikka, J. (2007) *Digital Contagions. A Media Archaeology of Computer Viruses*. Peter Lang: New York.

Philip J. D. (2006). What is meant by the word random. *Mathematics and Common Sense*, 180-182

PrinWie, A, and Van den Poel, D. (2006) Incorporating sequential information into traditional classification models by using an element/position-sensitive SAM,. *Decision Support Systems* 42 (2), 508–526.

Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Ramneek, P., (2003) Bots & botnet: an overview. *SANS Institute*. Retrieved 2012 http://www.sans.org/reading_room/whitepapers/malicious/bots-botnet-overview_1299

Residues conserved among various G protein coupled receptors are highlighted in green In *Wikipedia*. Retrieved 2012 from http://en.wikipedia.org/wiki/File:Conserved_residues.svg

Rieck, K., Hols, T., Willems, C., Düssel, P. and Laskov, P. (2008) Learning and classification of malware behavior. In D. Wamboni, editor, *DIMVA*, Volume 5137 of Lecture Notes in Computer Science, Springer, 108-125.

Rose J.P., Wu, C.K., Hsiao, C.D., Breslow, E., Wang, B.C. (1996) oxytocin-neurophysin complex based on: "Crystal structure of the neurophysin-oxytocin complex". *Nat.Struct.Biol.* 3: 163-169

Schuster, S. (2007). Blocking marketscore: why cornell did it. *The Original*, http://web.archive.org/web/20070214111921/http://www.cit.cornell.edu/computer/security/marketscore/MarketScore_rev2.html

Singhal, P. and Raul, N. (2012) Malware detection module using machine learning algorithms to assist with centraliWed security in enterprise networks. *International Journal of Network Security & Its Applications* (IJNSA), Vol.4, No.1, 61-67.

Smith, T.F. and Waterman, M.S. (1981) Identification of Common Molecular Subsequences, *Journal of Molecular Biology* 147, 195–197.

Sparks, S. and Butler, J. (2005). Widnows rottkits of 2005, part one. *Symantec Connect* Received 2012 http://www.symantec.com/connect/articles/windows-rootkits-2005-part-one

Strickland, J. (2011) "Ten worst computer viruses of all time." 2011. http://computer.howstuffworks.com/worst-computer-viruses1.ht

Symantec Internet Security Threat Report (2011). *Trends for 2010*. Vol. 16, 2011. http://www.symantec.com/business/threatreport/index.jsp

Szor, P. (2005) *The art of computer virus research and defense*. Addison Wesley.

Szor, P. and Ferrie, P (2011) "Hunting for metamorphic" 2012. http://www.peterszor.com/metamorp.pdf

Tang, Y. and Chen, S. (2007) An automated signature-based approach against polymorphic internet worms. *IEEE Transactions on Parallel and Distributed Systems* 18(7): 879-892.

Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*., 22, 4673-4680.

Tulloch, M., In Koch, J. and Haynes, S. (2003). *Microsoft Encyclopedia of Security*. Redmond, Washington: Microsoft Press. P.16

Watson, S. and Arkinstall, S. (1994). In *G Protein-Linked Receptor Factsbook,* Academic Press.

Wolfgan Merkel, (2002). Kolmogorov Loveland Stochasticity in Automata, languages and programming: 29th international colloquium. In Widmayer, P et al. (2002nd Ed.) *Springer,* pp. 391

Xinguang, T., Miyi, D., Chunlai, S. and Xin, L. (2009) Detecting network intrusions by data mining and variable length sequence pattern matching. *J Sys Eng Electr*, 20 (2), 405-411.

Xu D. and Whang, Y. (2012) Ab initio protein structure assembly using continuous structure fragments and optimiWed knowledge-based force field. *Proteins*, 80, 1715-1735.

Appendix I

Overview of datasets produced by systems and methods

Datasets	**Description**	**Number**
70 viruses signatures	Original hexadecimal signautes	1
70 vorm signatures	Original hexadecimal signatures	1
V10	Original 10 viruses hexadecimal signatures converted into amino acid represetations (table 1)	1
W10	Original 10 worms hexadecimal signatures converted into amino acid represetations (table 1)	1
V20	Original 20 viruses hexadecimal signatures converted into amino acid represetations (table 1)	1
W20	Original 20 worms hexadecimal signatures converted into amino acid represetations (table 1)	1
V30	Original 30 viruses hexadecimal signatures converted into amino acid represetations (table 1)	1
W30	Original 30 worms hexadecimal signatures converted into amino acid represetations (table 1)	1
V60	Original 60 viruses hexadecimal signatures converted into amino acid represetations (table 1)	1
W60	Original 60 worms hexadecimal signatures converted into amino acid represetations (table 1)	1
V70	Original 70 viruses hexadecimal signatures converted into amino acid represetations (table 1)	1
W70	Original 70 worms hexadecimal signatures converted into amino acid represetations (table 1)	1
V30R1-R3	Original 30 viruses hexadecimal signatures converted into amino acid represetations (table 2)	3
R1RW30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘W’	1
R1RV30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘V’	1
R1R10W30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘W’. Length of sequences equal 10	1
R1R10V30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘V’. Length of sequences equal 10	1
R1R20W30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘W’. Length of sequences equal 20	1
R1R20V30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘V’. Length of sequences equal 20	1
R1R40W30	Original 30 numeric random sequences coverted	1

	into maino acid represetation R1 (table 2), class name ‘W’. Length of sequences equal 40	
R1R40V30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘V’. Length of sequences equal 40	1
R1R60W30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘W’. Length of sequences equal 60	1
R1R60V30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘V’. Length of sequences equal 60	1
R1R72W30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘W’. Length of sequences equal 72	1
R1R72V30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘V’. Length of sequences equal 72	1
R1R98W30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘W’. Length of sequences equal 98	1
R1R98V30	Original 30 numeric random sequences coverted into maino acid represetation R1 (table 2), class name ‘V’. Length of sequences equal 98	1
W30R1-3	Original 30 worms hexadecimal signatures converted into amino acid represetations (table 2)	3
V60R1-5	Original 60 viruses hexadecimal signatures converted into amino acid represetations (table 2 and table 29)	5
W60R1-5	Original 60 worms hexadecimal signatures converted into amino acid represetations (table 2 and table 29)	5
SAVT30	Single alignmed virus signatures using T-Coffee without matrix and representation (table 1)	1
SAWT30	Single alignmed worm signatures using T-Coffee without matrix and representation (table 1)	1
SAVT30R1-3	Single alignmed virus signatures using T-Coffee without matrix and representation 1-3 (table 2)	3
SAWT30R1-3	Single alignmed worms signatures using T-Coffee without matrix and representation 1-3 (table 1)	3
SAWCI60R1-3	Single aligned worm signatures using ClustalW, identity matrix and representation 1-3	3
SAVCI60R1-3	Single aligned viruses signatures using ClustalW, identity matrix and representation 1-3	3
SAWCG60R1-5	Single aligned worms signatures using ClustalW, Gonnet and representation 1-5	5
SAVCG60R1-5	Single aligned viruses signatures using ClustalW, Gonnet and representation 1-5	5
SAWCG60R1-3	Single aligned worms signatures using ClustalW,	3

	BLOSUM and representation 1-3	
SAVCG60R1-3	Single aligned viruses signatures using ClustalW, BLOSUM and representation 1-3	3
SAWTB60R1-3	Single aligned worms signatures using T-Coffee, BLOSUM and representation 1-3	3
SAVTB60R1-3	Single aligned viruses signatures using T-Coffee, BLOSUM and representation 1-3	3
R1R10W30CI	Single aligned random sequences of class ‘W’ using ClustalW, identity matrix and repesetation R1 (table 2). Length of sequences equal 10	1
R1R10V30CI	Single aligned random sequences of class ‘V’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 10	1
R1R20W30CI	Single aligned random sequences of class ‘W’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 20	1
R1R20V30CI	Single aligned random sequences of class ‘V’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 20	1
R1R40W30CI	Single aligned random sequences of class ‘W’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 40	1
R1R40V30CI	Single aligned random sequences of class ‘V’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 40	1
R1R60W30CI	Single aligned random sequences of class ‘W’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 60	1
R1R60V30CI	Single aligned random sequences of class ‘V’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 60	1
R1R72W30CI	Single aligned random sequences of class ‘W’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 72	1
R1R72V30CI	Single aligned random sequences of class ‘V’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 72	1
R1R98W30CI	Single aligned random sequences of class ‘W’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 98	1
R1R98V30CI	Single aligned random sequences of class ‘V’ using ClustalW, identity matrix and represetation R1 (table 2). Length of sequences equal 98	1
WCCI60R1-3	Worm consensus using ClustalW, identity matrix and representations 1-3	3
VCCI60R1-3	Virus consensus using ClustalW, identity matrix and representations 1-3	3
WCCG60R1-5	Worm consensus using ClustalW, Gonnet matrix and representations 1-3	5

VCCG60R1-5	Virus consensus using ClustalW, Gonnet matrix and representations 1-3	5
WCCB60R1-3	Worm consensus using ClustalW, BLOSUM matrix and representations 1-3	3
VCCB60R1-3	Virus consensus using ClustalW, BLOSUM matrix and representations 1-3	3
WCTB60R1-3	Worm consensus using T-Coffee, BLOSUM matrix and representations 1-3	3
VCTB60R1-3	Virus consensus using T-Coffee, BLOSUM matrix and representations 1-3	3
DT60	Doubly aligned worm and virus signatures using T-Coffee without matrix and representations (table 1)	1
DT60R1-3	Doubly aligned worm and virus signatures using T-Coffee without matrix and representations 1-3 (i.e. aligning SAWT30R1 with SAVT30R1, SAWT30R2 with SAVT30R2 and SAWT30R3 with SAVT30R3)	3
DACI120R1-3	Doubly aligned worm and virus signatures using ClustalW, identity matrix and representations 1-3 (i.e. aligning SAWCI60R1 with SAVCI60R1, SAWCI60R2 with SAVCI60R2 and SAWCI60R3 with SAVCI60R3)	3
DACG120R1-5	Doubly aligned worm and virus signatures using ClustalW, Gonnet matrix and representations 1-3 (i.e. aligning SAWCG60R1 with SAVCG60R1, SAWCG60R2 with SAVCG60R2, SAWCG60R3 with SAVCG60R3, SAWCG60R4 with SAWCGR4 and SAWCGR5 with SAWCGR5)	5
DACB120R1-3	Doubly aligned worm and virus signatures using ClustalW,BLOSUM matrix and representations 1-3 (i.e. aligning SAWCB60R1 with SAVCB60R1, SAWCB60R2 with SAVCB60R2 and SAWCB60R3 with SAVCB60R3)	3
DATB120R1-3	Doubly aligned worm and virus signatures using T-Coffee, BLOSUM matrix and representations 1-3 (i.e. aligning SAWTB60R1 with SAVTB60R1, SAWTB60R2 with SAVTB60R2 and SAWTB60R3 with SAVTB60R3)	3
R1R10CI60	Doubly aligned two classes using ClustalW, identity matrix and repesetation R1 (table 2), Length of sequences equal 10	1
R1R20CI60	Doubly aligned two classes using ClustalW, identity matrix and repesetation R1 (table 2), Length of sequences equal 20	1
R1R40CI60	Doubly aligned two classes using ClustalW, identity matrix and repesetation R1 (table 2), Length of sequences equal 40	1
R1R60CI60	Doubly aligned two classes using ClustalW,	1

	identity matrix and repesetation R1 (table 2), Length of sequences equal 60	
R1R72CI60	Doubly aligned two classes using ClustalW, identity matrix and repesetation R1 (table 2), Length of sequences equal 72	1
R1R98CI60	Doubly aligned two classes using ClustalW, identity matrix and repesetation R1 (table 2), Length of sequences equal 98	1
Total number of datasets		140

Appendix II

Processing Alignment with six non-aligned virus and worm hexadecimal signatures

A simple dataset of 6 non-aligned viruses and worms

Virus.Acad.Bursted.a:

5f5cc33674618daf291b8ead159da2a79c7cf0609702bce121a34a8ab4f5474ba2

bae3a5

Virus.Acad.Bursted.c

fc26102aa4e1675a00d60156d9a8ef4ef85f193b870a1221eb67d248f2d6f6db80

c0848b

Virus.Acad.Bursted.d

e03487ef5450d0176a1ba5185f9d1423dec513fa3451e5a90576b33b7ec4038121

5ba380

Worm.BAT.Agent.j

6fb56373bde388174126fecf9143eeff2aae6b7486224c8fd213918abc38393357

fa4fc7

Worm.BAT.Agent.k

860c3bc3e5fc6a56ff6031ba46c245f6a4e2b8c09ef4b7acd814e46ce91527405e

eda980

Worm.BAT.Agent.n

51bef6dbf6a89278a3fb09e448192c79abafe47dd56f19d0a114ed07677e646040

8c5c14

Step (0): Converted into amino acid representation

>CV000001|descr=Virus.Acad.Bursted.a.

CSCMMNNQEDQAGFLSRHAKGPLFACHFLRLEHMEMSIQIHEIRKMPARALNDLGLKDSCDEDKLR

KLPNLCRCQN

>CV000002|descr=Virus.Acad.Bursted.c.

SMRQAIRLLDPAQECLIIFQIACQFHLGPSDPSGCSAHNKGEILARRAPKQEFRDGSRFQSQFKGI

MIGDGKAHGR

>CV000003|descr=Virus.Acad.Bursted.d.

PINDGEPSCDCIFIAEQLAKLCAGCSHFADRNFPMCANSLNDCAPCLHICEQKNNKEPMDINGARA

CKLNGINGGI

>CW000001|acc=CW000001|descr= Worm.BAT.Agent.j

QSKCQNENKFPNGGAEDARQSPMSHADNPPSSRLLPQKEDGQRRDMGSFRANHAGLKMNGNHNNCE

SLDSME

>CW000002|acc=CW000002|descr= Worm.BAT.Agent.k

GQIMNKMNPCSMQLCQSSQINAKLDQMRDCSQLDPRKGMIHPSDKELMFGADPDQMPHACREDICP

PFLHGI

>CW000003|acc=CW000003|descr= Worm.BAT.Agent.n

CAKPSQFKSQLGHREGLNSKIHPDDGAHRMEHLKLSPDEFFCQSAHFILAADPFIEQEEPQDQIDI

GMCMAD

Step (a) Align the sequences of computer viruses and worms separately

CLUSTAL W (1.83) multiple sequence alignment

CV000001|descr=Virus.Acad.Bursted.a.

--------------C------SCMMNNQEDQ------AGFLSRHA------KGPL-------------FACHFLRLEHMEMSIQIHEIRKMP-ARALNDLGLKDSCDEDKLRKLPNLCRCQN

CV000002|descr=Virus.Acad.Bursted.c.

SMRQAIRLLDPAQECLIIFQIACQFHLGPSD------PSGCSAHN------KGEI-------------LARRAPKQE----------FRDGS-RFQSQFK GIMI------G----DGKAHGR

CV000003|descr=Virus.Acad.Bursted.d.

P----INDGEPSCDCIFIAEQLAKLCAGCSHFADRNFPMCANSLN----- -DCAP-------------CLHICEQKN-------NKEPMDIN-GARA----CK-------L----NGINGGI

CW000001|acc=CW000001|descr=

QS-----------KCQNE----NKFPNGGAEDA----RQSPMSHA------DNPP------------ -SSRLLPQKE-------DGQRRDMG-SFRANHA GLKMNGNHNNC----ESLDSME

CW000002|acc=CW000002|descr=

GQ--IMNKMNP---CSMQLCQSSQINAKLDQMR------DCS----------------------- ----QLDPRKG-------MIHPSDKELMFGADPD QMPHACREDIC----PPFLHGI

CW000003|acc=CW000003|descr=

--------------CAKP----SQFKSQLGHREGLNSKIHPDDGAHRMEH LKLSPDEFFCQSAHFILAADPFIEQEE-------PQDQID---------- -------------- IGMCMAD

Step (b) Represent the gaps with an amino acid. Code gaps as 'W'

>CV000001|descr=Virus.Acad.Bursted.a.

WWWWWWWWWWWWWCSCMMNNQEWWWWWWWWDQAGFLSRHAKGPLFA CHFLRLEHMEMSIQIHEIRKMPARALNDLGLKDSCDEDKLRKLPWWWWWW WWWWWNLCRCQN

>CV000002|descr=Virus.Acad.Bursted.c.

SMRQAIRLLDPAQECLIIFQIACQFHLGPSDPSGCSAHNKGEILARRAPKQEFR DGSRFQSQFWWWWWWWWWWWWWWWWWWWKGIMIWWWWWWWWW WWGDGKAHGR

>CV000003|descr=Virus.Acad.Bursted.d.

WWWWPINDGEPSCDCIFIAEQLAWWWWWWKLCAGCSHFADRWWWWWWW WWWWWWWWWWWWWWNFPMCANSLNDCAPCLHICEQKNNKEPMDINGA RACKL NGINGGI

>CW000001|acc=CW000001|descr= Worm.BAT.Agent.j

QSKWWWWWWWWWWWWCWWWWWWQNENKFPNGWGAEDARQSPMSHAD NPPSSRLLPQKEDGQWWWWWWWRWWWWWRDMWGSFRANHAGLKMNGN HNNCESLDSME

>CW000002|acc=CW000002|descr= Worm.BAT.Agent.k

GQIMNKMNPCSMQLCQSSQINAKLDQMRWWWWWWWWDCSQLDPWWWW RKGMIHPSDWWWWWWWWWWWWWWWWWWWKELMFGADPDQMPHACR EDICPPFLHGI

>CW000003|acc=CW000003|descr= Worm.BAT.Agent.n

WWWWWWWWWWWWWWCWWWWWWAKPSQFKSQLGHREGWWWWWWW WWWWLNSKIHPDDGAHRMEHLKLSPDEFFCQSAHFILAADPFIEQEEPQDQI DIGMCMAD

Step (c) Align all samples together

CLUSTAL W (1.83) multiple sequence alignment

CV000001|descr=Virus.Acad.Bursted.a.

WWWWWWWWWWWWWCSCMMNN-------QEWW-------WWWW-WWDQAGFL SR--HAKGPL---FACHFLRLEHMEMSIQIHEIRKMPARAL- NDLGLKDSCDEDKLRKLPWWWWWWWW-WWWNLCRCQN

CV000002|descr=Virus.Acad.Bursted.c.

SMRQAIRLLDPAQECLIIF-------QIAC-------QFHLGPSDPSG-C SA-- HNKGEI---LARRAPKQEFRDGSRFQSQFWWWWWWWW- WWWWWWWWWWWWWKGIMIWWWWWWWWW-WWGDGKAHGR

CV000003|descr=Virus.Acad.Bursted.d.

WWWWPINDGEPSCDCIFIA-------EQLA-------WWWWWWKLCAG-C SH-- FADRWW---WWWWWWWWWWWWWWWWWWWWNFPMCANSL- NDCAPCLHICEQKNNKEPMDINGARA-CKLNGINGGI

CW000001|acc=CW000001|descr=

QSKWWWWWWWWWWWWCWWWW-------WWQN-------ENKF-PNGWGA------ EDARQS--- PMSHADNPPSSRLLPQKEDGQWWWWWWWRWWWWWRDMWGSFRANH AGLKMNGNH--NNCESLDSME

CW000002|acc=CW000002|descr=

GQIMNKMNPCSMQLCQSSQINAKLDQMRWW-------WWWW-WWDCSQLD PWWWWRKGMIHPSDWWWWWWWWWWWWWWWWWWWKELMFGAD--- ---------------PDQMPHACREDICPPFLHGI

CW000003|acc=CW000003|descr=

WWWWWWWWWWWWWWCWWWW-------WWAKPSQFKSQLGHR- EGWWWW-W WW--WWWWLN--- SKIHPDDGAHRMEHLKLSPDEFFCQSAHF-ILAADPF I-EQEEPQD---------- QIDIGMCMAD

Code all gaps introduced as this stage as ‘Y’.

>CV000001|descr=Virus.Acad.Bursted.a.

WWWWWWWWWWWWCSCMMNNYYYYYYYQEWWYYYYYYYYWWWWYWWDQ

AGFLSRYYHAKGPLYYYFACHFLRLEHMEMSIQIHEIRKMPARALYNDLGLKD

SCDEDKLRKLPWWWWWWWWYWWWNLCRCQN

>CV000002|descr=Virus.Acad.Bursted.c.

SMRQAIRLLDPAQECLIIFYYYYYYYQIACYYYYYYYQFHLGPSDPSGYCSAYYHN

KGEIYYYLARRAPKQEFRDGSRFQSQFWWWWWWWWYWWWWWWWWWWW

WKGIMIWWWWWWWWWYWWGDGKAHGR

>CV000003|descr=Virus.Acad.Bursted.d.

WWWWPINDGEPSCDCIFIAYYYYYYYEQLAYYYYYYYWWWWWWKLCAGYCSH

YYFADRWWYYYWWWWWWWWWWWWWWWWWWWNFPMCANSLYNDCAPC

LHICEQKNNKEPMDINGARAYCKLNGINGGI

>CW000001|acc=CW000001|descr= Worm.BAT.Agent.j

QSKWWWWWWWWWWWCWWWWYYYYYYYYWWQNYYYYYYYENKFYPNGWG

AYYYYYYEDARQSYYYPMSHADNPPSSRLLPQKEDGQWWWWWWWRWWWW

WRDMWGSFRANHAGLKMNGNHYYNNCESLDSME

>CW000002|acc=CW000002|descr= Worm.BAT.Agent.k

GQIMNKMNPCSMQLCQSSQINAKLDQMRWWYYYYYYYYWWWWYWWDCSQL

DPWWWWRKGMIHPSDWWWWWWWWWWWWWWWWWWWKELMFGADYY

YYYYYYYYYYYYYYYYPDQMPHACREDICPPFLHGI

>CW000003|acc=CW000003|descr= Worm.BAT.Agent.n
WWWWWWWWWWWWWWCWWWWYYYYYYYWWAKPSQFKSQLGHRYEGWW
WWYWWWYYWWWWLNYYYSKIHPDDGAHRMEHLKLSPDEFFCQSAHFYILA
ADPFIYEQEEPQDYYYYYYYYYYQIDIGMCMAD

Appendix III

List of sequence alignment software and machine learning methods

Multiple Sequence Alignment

ClustalW	It will also make use of multiple processors, where present. In addition, the quality of alignments is superior to previous versions, as measured by a range of popular benchmarks.
T-Coffee	(Tree-based Consistency Objective Function For alignment Evaluation) is a multiple sequence alignment software using a progressive approach

Motif finding

Motif3D	is a simple wireframe protein structure viewer that has been designed specifically for use with the PRINTS database

Machine Learning

J48	an open source Java implementation of the C4.5 decision tree algorithm
LAD Tree	Logical Analysis of Data is the method for classification proposed in optimization literature.
Naive Bayes	is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions.
OneR	short for "One Rule", is a simple, yet accurate, classification algorithm that generates one rule for each predictor in the data, then selects the rule with the smallest total error as its "one rule".
Multiple Layer Perceptron	is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs
PRISM	Aim to reduce modular classification rules directly from

the training set.