國立交通大學

資訊科學與工程研究所

碩 士 論 文

針對未知攻擊辨識之混合式入侵偵測系統

Hybrid Intrusion Detection System Toward Unknown Attack Classification

研 究 生：蔡秉任

指導教授：曾文貴　教授

中 華 民 國　103　年　9　月

針對未知攻擊辨識之混合式入侵偵測系統

Hybrid Intrusion Detection System Toward Unknown Attack Classification

研 究 生：蔡秉任　　Student：Ping-Jen Tsai

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2014

Hsinchu, Taiwan, Republic of China

中華民國 103 年 9 月

針對未知攻擊辨識之混合式入侵偵測系統

學生：蔡秉任　　指導教授：曾文貴

國立交通大學資訊科學與工程研究所碩士班

摘　要

入侵偵測系統(IDS)用途是作為偵測網路惡意行為的防線，透過流量、封包等資料所擷取出的特徵，交由系統來做惡意行為的偵測與過濾，以防止惡意入侵行為，並減少其所造成的各種損失。

入侵偵測系統又可分為特徵偵測(Signature-based)與異常偵測(Anomaly-based)兩大類型。特徵偵測系統主要藉由過去的攻擊行為所取出的特徵，建立異常特徵資料庫，往後便依據特徵的比對結果來偵測惡意及入侵行為；而異常偵測系統，則是利用如機器學習(Machine Learning)的方法，針對資料集的樣本，根據其特徵與標籤的關係，建構出攻擊行為的模型，來做為辨識的依據。異常偵測類型的系統，相較於特徵偵測之系統，其優勢是有找出零時攻擊(Zero-day Attack)之能力。

本研究之目的便在於針對使用異常偵測機制的入侵偵測系統，加強其辨識零時攻擊或未知攻擊(Unknown Attack)之能力，來達到對零時攻擊更全面的捕捉。為了加強辨識零時攻擊的能力，將透過多種方法的混合應用來達成目的，如自我學習機制(Self-Learning)、多層次系統(Multilevel System)、分群法(Clustering)、以及隱藏馬可夫模型(Hidden Markov Model)等等，而從實驗數據可看到，透過自我學習機制(Self-Learning)，對於未知攻擊的偵測率結果有明顯的上升，同時代表誤判的偽陽性(False Posi-tive)僅微幅的上升，也因此，整體的偵測準確率(Accuracy)也較原始結果來的好；然而，其他如混合分群法的方式，或是隱藏馬可夫模型等，卻較無實質上之效能改善及偵測能力之提升，此結果除了與資料集的特性有關，也意味著藉由簡單的馬可夫狀態之模型，可能無法建立出完整的惡意行為模型以做為辨識之用，應需要有更詳盡的演算法及定義其他相關的特徵資訊，才有辦法對惡意入侵行為達到更完善的偵測結果。

關鍵字：入侵偵測系統、零時攻擊、未知攻擊。

Hybrid Intrusion Detection System Toward Unknown Attack Classification

Student：Ping-Jen Tsai Advisors：Wen-Guey Tzeng

Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University

Abstract

Intrusion detection system (IDS) is used to detect the malicious network behavior(e.g., Denial-of-Service、Trojan Horse). It detects attacks by the features extract from network traffic, packet, etc. and alert the users when a potentially threat was be detected to reduce the damage of data, system, and money.

IDS could fall into 2 categories, signature-based and anomaly-based. Signature-base IDS is scanned for known signs of attacks. A database of signature is built by human expert, they extract the signature by the knowledge and analysis of past attacks. Anomaly-based IDS is built the malicious behavior model by training data and machine learning algorithm. The machine learning model will classify the instance is anomaly or not. The most import thing is that anomaly-based IDS have ability to detect the zero-day attack what signature-based IDS couldn't.

In this paper, we will focus on the ability to detect zero-day attack or unknown attack on anomaly-based IDS. To improve the detection rate of unknown attack, we apply self-learning, multilevel, and voting algorithms and combine these mechanisms to make the hybrid IDS more powerful. In addition, we have proposed a HMM classifier take the continues feature of netflow as observations. According to the result of experiment, we could find that self-learning will obviously make the classification result better, both of detection rate and accuracy increase significantly. But the hybrid system of self-learning and multilevel couldn't improve the result anymore and HMM classifier take a bad classification result. It was not only related the dataset property but also about the feature what we defined. And it illustrate that simply markov model possibly couldn't model the complicated attack behavior. These problems will be discuss in the paper.

Keywords：Intrusion Detection System(IDS)、Zero-day Attack、Unknown Attack

誌　謝

這兩年的碩士生涯，要感謝的人很多。當中最感謝的，就是我的指導教授，曾文貴教授的教導，無論是在事物的學習、研究的方法、報告該注意的技巧，許許多多的指導都令我獲益良多，而在我碩二的這年，老師即使人在美國進修，仍然每周撥出寶貴的時間與我們meeting，指導我的研究，討論的過程中，更給予我許多寶貴的建議與方向，讓我最後可以完成我的碩士論文，也感謝口試委員蔡錫鈞教授與孫宏民教授的指導與意見，讓我的論文更加的完整。

接著我要感謝博士班的學長，沈宣佐及陳毅睿，從剛進碩班時的讀書會，到後來修課、計畫、以及個人的研究及學習，都很仰賴他們的協助與指導；也謝謝我的同學們，李柏青、林浚騰、蔡立倫，一起度過這兩年的碩士生涯，從一開始一起參加比賽到最後的口試準備，都要感謝你們的付出，一切才能順利。

另外還有許多朋友的鼓勵跟支持，都是我碩士生涯一路走來的動力。最後，我更要謝謝我的家人，有你們，我才能完成我的碩士論文，謝謝你們。

目　錄

圖目錄

表目錄

一、 緒論(Introduction)

1.1. 入侵偵測系統

隨著網路及科技的快速發展，有著越來越多樣的網路協定、網路應用及服務充斥在我們的生活周遭，無論是過去的個人電腦、筆電，還是近年來熱門的智慧型手機、平板電腦，皆可藉由這些服務提供我們更便捷的生活。然而，在大量科技服務的生活背後，衍生出的各種惡意入侵行為卻往往被大眾所忽視，舉凡未經授權的資料存取行為、透過程式上的弱點或漏洞來嘗試取得伺服器主機的主控權、或是藉由大量的數據連線來癱瘓服務，皆可視為惡意入侵行為，而此類惡意行為所造成的影響，小至個人的私密資料外洩，大至網路企業因服務遭癱瘓而造成的鉅額財務損失，也突顯了防護入侵的重要性

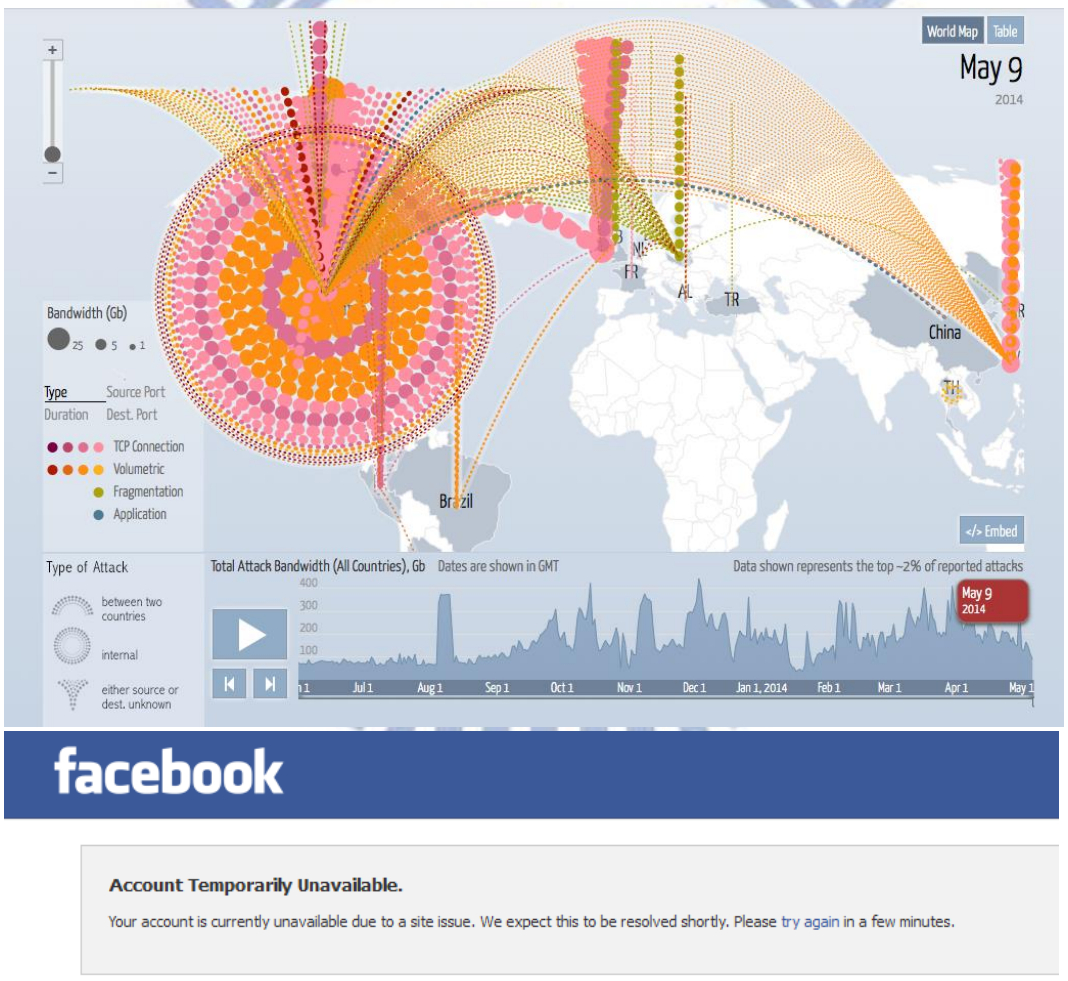


圖 1 使 facebook 服務中斷之大規模 DDOS 攻擊即時影像(來源:[1])

入侵偵測系統，便是透過數據連線的資料，包含流量、封包、連線類型等特徵值，來對入侵行為預先做到偵測，並由系統做初步的過濾並對管理員做出警示

的動作，再由管理員來檢視高風險的連線或是重新分析是否有誤判的情形。目前的入侵偵測系統，主要可以分為兩種類型，分別為特徵偵測(Signature-based)與異常偵測(Anomaly-based)兩種，特徵偵測是利用過去的入侵攻擊樣本，透過專家及系統的分析擷取出攻擊行為的特徵，建立攻擊行為的異常特徵資料庫，往後便利用異常特徵資料庫的比對來做入侵攻擊的偵測辨識，若連線資料吻合資料庫中的特徵，便會判定為入侵攻擊，此方法的優點為不容易誤判，但特徵的擷取及資料庫的建立往往需耗費大量的人力及時間，在當今攻擊行為快速變動的環境下，特徵的建立往往比不上攻擊的出現速度，也因此，特徵偵測系統最大的缺點即是無法偵測到未知的零時攻擊，對於緊急威脅沒有立刻應變的能力，知名的軟體 Snort 即為獲得廣泛運用的的特徵偵測系統；異常偵測則是利用已知的攻擊樣本與一般的正常樣本混合成的資料集，對個別樣本的特徵，藉由機器學習的演算法建構出攻擊行為的模型，作為辨識之用，其優點便在於有能力偵測出未知的零時攻擊，可以在新攻擊出現時便即時的察覺，但其同時卻也有著較高誤判率的缺點，即把一般正常的行為錯誤的辨識成惡意入侵攻擊，造成錯誤的連線阻斷，而過高的誤判率，也將導致系統的效益下降，以及造成管理人員的負擔，也因此，目前尚無較為廣泛運用的異常偵測系統，多為廠商在整體防護解決方案中所運用的部分辨識演算法而已。

1.2. 研究目標

對於零時攻擊的偵測是當今資訊安全上的重要課題，然而，全新的攻擊較難透過自動化的方式來達到偵測的結果，往往需要各方面的資料與專家來做手動的偵測辨識，才能發現全新的攻擊行為；慶幸的是，許多惡意入侵攻擊皆是由某些攻擊樣本變種而來，因此雖然是新的攻擊，卻會與過去的攻擊樣本有一定的相似性存在，而異常偵測機制的系統，便可透過建立攻擊行為的模型來進一步辨識出這些原先所沒看過的攻擊樣本，本研究的目標便在於加強此異常偵測優於特徵偵測機制的優點，希望對未知攻擊可以有更精確的辨識偵測結果，以期對當今惡意行為快速出現變化的環境，可以有更完善的因應及防護，同時，雖然異常偵測相較於特徵偵測有能夠辨識未知攻擊的優點，卻也因而可能導致誤判率過高，使得系統整體的偵測準確率下降，此也是需要注意的部分。

1.3. 貢獻

本研究將自我學習機制套用在入侵偵測系統上，並由實驗結果可看到，系統對於未知攻擊的偵測能力有顯著性的改善，而在加強偵測能力的同時，其誤判率僅為些許的上升，整體的偵測精確性也是呈現上升的狀況。為了嘗試讓自我學習之入侵偵測系統有更佳的表現，也加入了多層與投票的機制在其中，以避免最壞情況的大量發生。此外，也利用網路流當中連續性的特徵，來建立隱藏馬可夫模

型以嘗試作為降低誤判率的第二層過濾器，但隱藏馬可夫模型在此狀況下較無法精確的辨識出惡意行為與否，原因可能為過於簡單的特徵無法精準的建構出複雜行為的模型，此為未來仍需改進的部分。

1.4. 全文架構

本論文其餘的章節部分整理如下：在第二章主要會介紹目前 IDS 與 Unknown Attack 偵測相關的研究內容，以及分群法與隱藏馬可夫模型在 IDS 上的應用；第三章則會說明自我學習機制的概念及作法，並解釋在自我學習機制上混合多層化系統及建立投票機制的原因與想要達成的目的；第四章說明隱藏馬可夫模型及應用的概念，包含如何將離散的特徵資料整理成連續性的特徵供建立馬可夫模型使用，還有不同的辨識方式運用；第五章介紹實驗所使用的資料集，以及如何將單純的流量紀錄資料轉換成實驗所需的資料特徵集，並說明當中運用到的相關工具；第六章為實驗的數據結果呈現與討論；第七章則是全篇的總結。

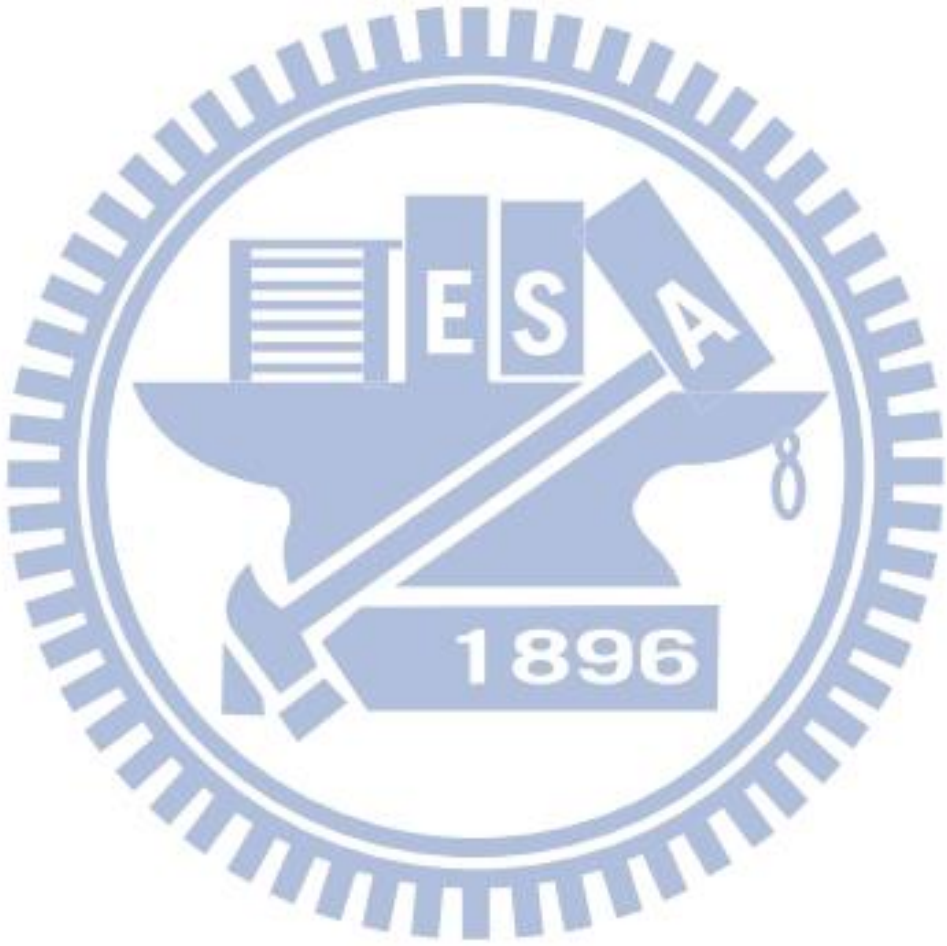
二、 相關研究(Related work)

入侵偵測系統的研究，從過去以特徵偵測為主的方式，到後來因電腦運算能力的增強，進而蓬勃發展的機器學習演算法，一直以來都是個熱門的研究領域。當中包含針對特定攻擊類型的研究，如 S.Seufert[2] 等人即提出了專門用於處理分散式阻斷服務攻擊的入侵偵測系統;或是對特定演算法應用在入侵偵測系統上的改進，G.Münz[3] 等人便對於 K-means 分群法做了更進一步的改善，藉由增加判斷樣本與群中心的門檻值，以及正規化方法的改善，來達到更好的辨識結果；另外，如 M.Barreno[4] 等人，對於此類使用機器學習方法來作為入侵偵測及郵件過濾的系統，也探討了其安全性相關的問題，說明了機器學習的演算法，本身就會成為一種新攻擊模式的目標；而對於入侵偵測系統的諸多研究，M.H.Bhuyan[5] 等人對其做了詳盡的概觀整理，從攻擊的類型定義、入侵系統的演算法類型、一直到使用的資料集以及系統效益的評測方法，都有相當清楚的說明及相關參考資訊。

至於對零時攻擊與未知攻擊的研究，P.Natesan[6] 等人於所提出的系統中，特別提出其對於未知攻擊的偵測能力並加以討論；P.Jongsuebsuk[7] 等人則利用模糊基因演算法，來加強入侵偵測系統對未知攻擊的辨識能力，並依據攻擊的類型分組，提供更全面的實驗結果，惟其所採用的攻擊僅阻斷式服務(DOS)及探測(probe)兩種，其辨識難度通常較為容易，也因此實驗所呈現的數據可能過度完美而失去某種程度上的評估能力；A.Aleroud[8] 等人則提出了透過數學的線性轉換方法，來對零時攻擊做進一步的辨識與偵測；P. M.Comar[9] 等人則是透過網路流量的資訊及入侵偵測系統，來偵測未知的惡意軟體(Malware)，而所提出的 Tree-Based 特徵，在支持向量機器(SVM)對未知惡意軟體流量的偵測上，也有相當程度的改善。

另外在實驗的資料集部分，過去研究上常用的為 KDD 99[10] 資料集，此為根據 1999 年的 DARPA 入侵偵測評測專案所建立的評測資料集，將 DARPA 專案的原始連線流量資料，透過異常與正常的標籤加註，以及對樣本所定義的高階特徵(High Level Feature)所加工而成，由於該類惡意流量資料集的缺乏，以及該資料集的連線資料涵蓋多種攻擊類型，KDD 99 在過去往往被用來做為入侵偵測系統的評測及比較標準，也因為 KDD 99 已經有整理好的高階特徵，對使用機器學習的異常偵測系統研究而言，可以更方便的使用，將心力放在機器學習演算法的改進上，故在入侵偵測系統上，多利用此資料集作為評測的實驗之用。而 DARPA 1999[11] ，是由麻省理工學院 Lincoln Labs 蒐集以作為入侵偵測研究及評估系統效益所使用的原始連線流量資料，其包含了三周的正常流量資料，以及兩周包含有惡意入侵攻擊的連線資料，以作為實驗及研究之用。另外針對 KDD 99 資料集，

M.Tavallaee[12] 等人有對此資料集的問題及缺陷提出研究與討論，並更進一步的修改此資料集之樣本，提出了 NSL-KDD[12] 資料集，並表示此修改後之資料集，能夠更有鑑別度的作為評測入侵偵測系統之用。



三、 混合式自我學習機制(Hybrid Self-Learning)

自我學習機制的方法，其目的在於利用未標籤樣本，配合演算法，嘗試建立出對未知攻擊有良好偵測能力之系統，而所謂的"混合"，在本研究中，指的是將多層以及投票的方法加入到系統中，希望改進系統的表現。首先，利用多層的機制，希望可以由多個針對個別攻擊的分類器，對各類攻擊達到更完美的辨識結果；而投票，則是在自我學習的機制下，在多階段的分類器訓練中，避免其可能產生最壞的狀況，使得最後訓練出的分類器之表現過於低落，本章將先介紹自我學習機制的內容，說明其運作的方法及概念，之後再討論如何在自我學習機制的系統上，加入多層分類器的方法，以及分層的原因及依據，試圖以混合式的方式來增進分類器對攻擊偵測的表現，最後，則是說明如何以投票的方法，來避免訓練結果的最壞情況發生。

3.1 自我學習(Self-Learning)

自我學習是種半監督式學習的方式，從自我訓練的方法延伸而來，希望藉由大量的未標籤樣本，配合原先利用訓練樣本所建立出的模型及演算法，增進模型的完整性及能力，以求訓練出更好的分類器作為未知攻擊偵測之用。

3.1.1 半監督式學習(Semi-Supervised Learning)

在機器學習的演算法中，依據訓練資料集當中資料的標籤有無，可將演算法分為監督式學習(Supervised Learning)與非監督式學習(Unsupervised Learning)，監督式學習即是依據已標籤的資料，將此作為訓練學習之用，建立辨識分類的模型，常見的如單純貝氏(Naïve Bayes)、決策樹(Decision Tree)演算法；相反的，非監督式學習則是以不具有標籤的資料，單單以資料樣本的特徵值，進行辨識模型的建立之用，如 K-Means 等分群法就是非監督式學習的方式，由尚未標籤的資料自行找出各個資料集的群中心並對資料加以分群處理。

半監督式學習則是介於監督式學習與非監督式學習之間的方式，以有標籤的資料為訓練基礎，再加上其他未標籤樣本所提供的特徵及行為資訊，混合監督式學習與非監督式學習的方法，以增進模型的完整性，並試圖改善演算法的辨識結果。半監督式學習的方法之所以受到廣泛的運用，原因在於資料的標籤工作，往往需要專業的人工方式來操作，相當的耗費人力及時間，因此，若要取得大量的已標籤資料是相當困難且高成本的，而藉由一部分的

已標籤資料，以及通常較為大量的未標籤資料，配合不同的半監督式學習演算法，來建立模型並進行辨識工作，就是種相當實用的方式。

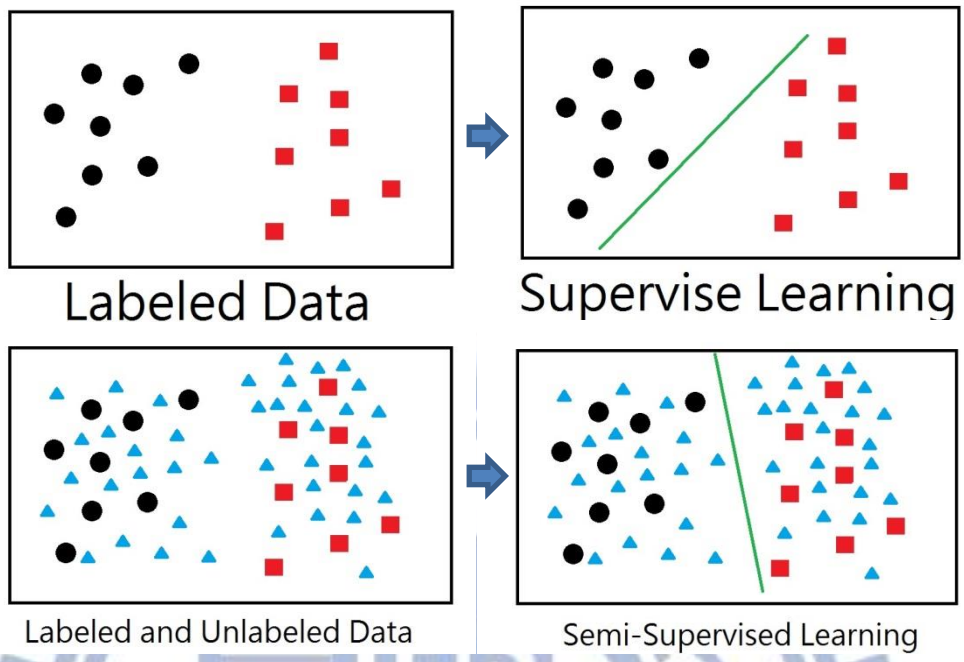


圖 2 Supervised and Semi-Supervised Learning

3.1.2 自我訓練與協同訓練(Self-Training and Co-Training)

在半監督式的機器學習演算法中，有所謂的 Co-Training 以及 Self-Training 演算法。A.Blum[13] 等人提出了 Co-Training 的方法，利用同樣一群擁有不同特徵集的已標籤樣本，即每個樣本都有兩種不同類型的特徵集，分別訓練出兩個分類器，再由此兩個分類器對未知標籤的樣本進行辨識，依照設定的係數從這些被辨識的未知樣本中挑出特定數量的自行標籤樣本，將其加入到訓練樣本中，並重新訓練出新的分類器。[13] 也證明了此方法在網頁內容的辨識上有非常好的成果。

V.Ng[14] 等人則透過類似 Co-Training 的概念，並加以利用 Bagging 及 Majority Voting 的方法，實作了 Self-Training，最大的差異在於所使用的是同樣的特徵值集合，而非如 Co-Training 是以不同的特徵集訓練出不同的分類器。Self-Training 最普遍的方式，是利用已標籤的訓練樣本建構出一個或多個分類器，再依這些分類器對尚未標籤的樣本進行標籤的動作，並依照所訂定的規則，例如投票機制、門檻值的設定，來決定要將哪些擁有較高信心權值標籤的未知樣本加入到訓練樣本當中，增加訓練集中的樣本數後，再對分類器重新進行訓練，以建立更完善的機器學習模型。

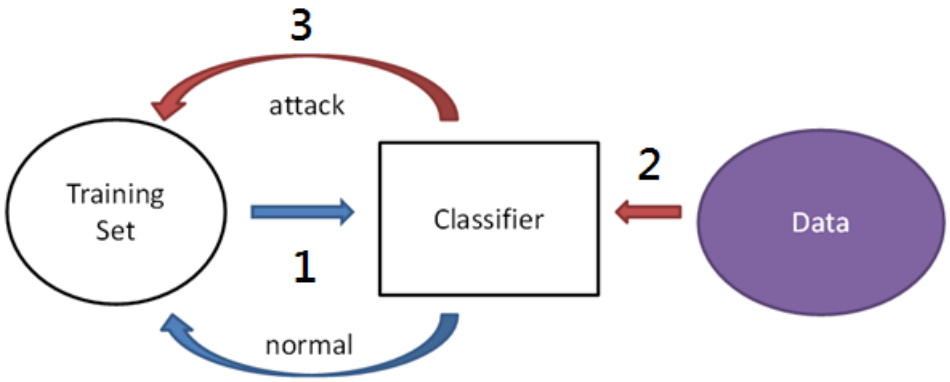


圖 3 Self-Training

3.1.3 自我學習(Self-Learning)

而本研究中所指的自我學習機制，則是由 J.Zhang[16] 等人，在對未知服務的流量辨識上所採用的方法，可視為由 Self-Training 的精神所延伸發展而成，其概念在於藉由重複對未標籤樣本集做辨識，來增加訓練集當中的惡意樣本數目。首先利用已標籤的訓練樣本訓練出分類器，以此分類器對未標籤樣本進行辨識且標籤，並將被辨識為惡意樣本的資料加到訓練樣本中，重訓練新的分類器後，再以新的分類器對上一輪被辨識為普通樣本的資料加以辨識及加上標籤，接著同樣將這些樣本中被標籤為惡意樣本的資料加到訓練樣本，重複這樣的程序，直到沒有樣本被辨識為惡意，或是到達設定的執行上限門檻為止。此作法的意義便在於增加原本已標籤的訓練資料中較少的惡意樣本數，並藉由不斷重複的過程來辨識可能為惡意的樣本，以獲得更多與惡意樣本有關的特徵與行為資訊，詳細的演算法如下。

Self-Learning Algorithm

Input: Positive(anomaly) training set $\mathbf{T_A}$、Negative(normal) training set $\mathbf{T_N}$、Unlabeled data set **U**、Iteration limit **L**

Output: New training set **T**, New classifier **C**

$T_0 \leftarrow T_A \cup T_N$
Create classifier C_0 from T_0
Classify U by C_0
Put positive samples classified by C_0 into A_1

Put negative samples classified by C_0 into N_1

$i = 1$

While ((A_i != ∅) && ($i < L$))

$T_A \leftarrow T_A \cup A_i$

$T_i \leftarrow T_A \cup T_N$

Create classifier C_i from T_i

Classify N_i by C_i

Put positive samples classified by C_i into A_{i+1}

Put negative samples classified by C_i into N_{i+1}

$i = i + 1$

return T_i, C_i

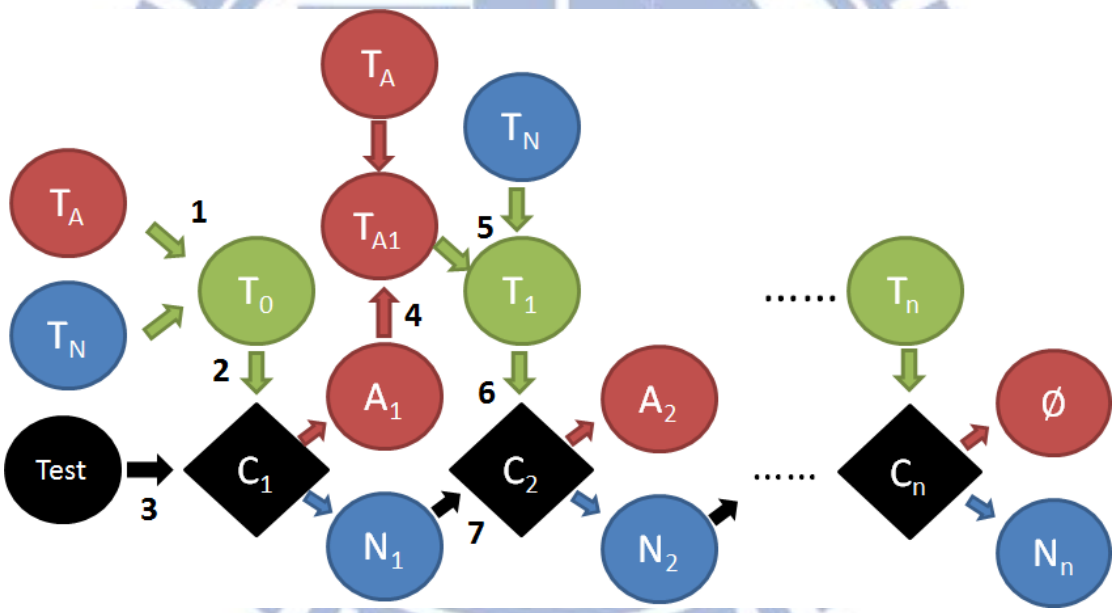


圖 4 Self-Learning 示意圖

而將此方法套用在對惡意、未知攻擊的偵測上，由第六章的實驗結果可以看到，對於辨識結果同樣有著相當程度的改進。

3.2 混合式多層機制(Hybrid Multilevel Mechanism)

混合式機制的概念在於透過多種演算法的配合、互相補足，往往可以得到比單一演算法更好的結果，如對不同類型的資料採用不同的機器學習演算法進行辨識，或是在機器學習演算法中加入如模糊演算法(Fuzzy Algorithm)來調整機器學

習訓練出的模型，諸如此類的混合式機制，都是在機器學習上相當常見的做法。而在這邊，我們將嘗試利用多層的方式，以多個針對不同類型攻擊的分類器合作，並以過濾式的流程，配合前一節所提的自我學習方法，希望藉此來得到更佳的辨識結果。

3.2.1 多層(Multilevel)

多層的分類器做法是為了能夠針對各類型的攻擊，採用最適合偵測該攻擊類型的機器學習分類器，以提高攻擊的偵測率，如表二便顯示了 Decision Tree 與 Naïve Bayes 演算法，對 KDD 99 資料集進行辨識的的不同結果與差異[17] ；同時在將各類型的攻擊樣本分開後，對於各個分類器來說，減少了某些行為模式可能相差過大，但卻同樣被標籤為惡意的樣本。如阻斷式服務攻擊與緩衝區溢位攻擊，兩者樣本的行為特徵可能就有很大的不同，而藉由分層，便可以過濾掉訓練樣本上的雜訊，以求得更精確的辨識結果。

	Decision Tree	Naïve Bayes
Normal	**99.50%**	97.68%
DoS	**97.24%**	96.65%
R2L	0.52%	**8.66%**
U2R	**13.60%**	11.84%
Probe	77.92%	**88.33%**

表 1　不同演算法辨識能力的差異(來源:[17])

3.2.2 攻擊種類多層(Multilevel of Attack Category)

DARPA 及 KDD 資料集所定義的四大攻擊類型，如下所列：

- **DOS**: denial-of-service, e.g. syn flood;
- **R2L**: unauthorized access from a remote machine, e.g. guessing password;
- **U2R**:　unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow'' attacks;
- **Probe**: surveillance and other probing, e.g., port scanning.

而攻擊種類多層(Multilevel of Attack Categor)便是依據此分類來做分層的原則。其中由於 R2L 與 U2R 類型之樣本數較少，故將其合併為同一層，藉由同一個分類器來做辨識；而因根據統計分析，對 DOS 的偵測誤判率為最低，Probe 次之，誤判率最高的則是 R2L 及 U2R 這類，於是便利用過濾的方式，從 DOS 的攻擊開始偵測，以達到最好的效益，建構好的多層入侵偵測系統之架構如下圖。

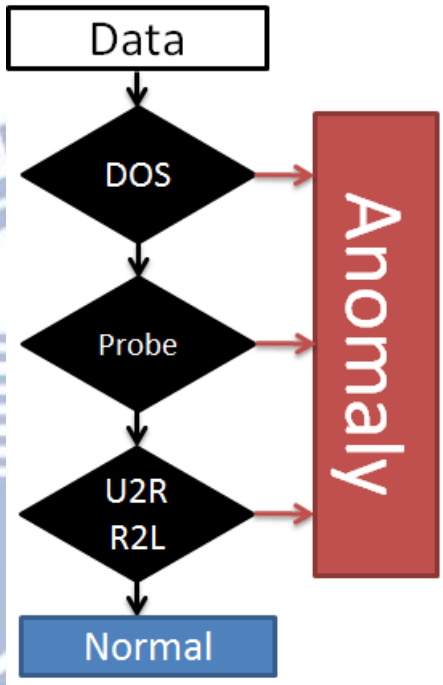


圖 5　攻擊種類多層系統架構

3.2.3 分群多層(Multilevel of Clustering)

但僅藉由攻擊的種類來對系統做分層，並不見得客觀，首先面臨到的問題，便在於要分幾層？要針對官方所給的攻擊種類分層，還是根據個別的攻擊分層，又或是根據攻擊的樣本數量分層？無論哪一項，皆缺乏了一個客觀的分層依據。所以在此就嘗試利用分群法的方式，來對攻擊樣本們作分群的動作，希望藉由分群後的結果，來做為分層的客觀依據。

示意圖如下，首先將訓練樣本中的惡意攻擊資料進行分群，再將不同群的惡意攻擊資料與正常的資料混合，之後依照各個群的資料樣本，訓練出各個不同的分類器，以作為偵測攻擊之用。

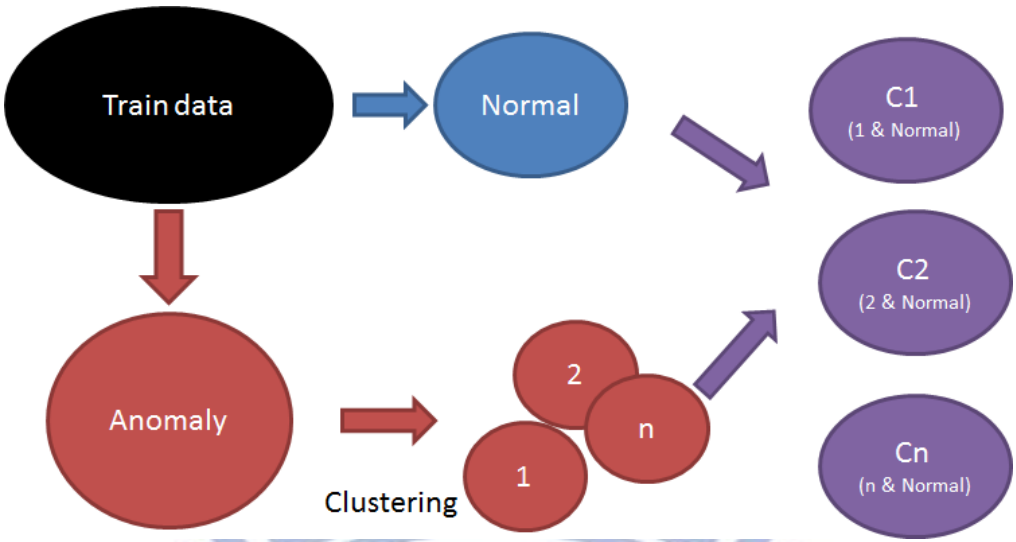


圖 6　將訓練樣本分群並訓練出多個分類器

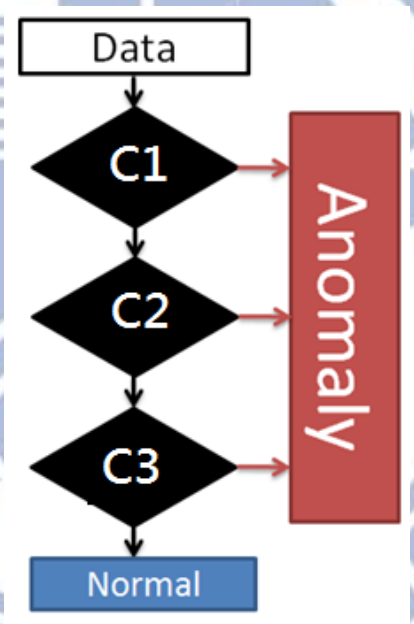


圖 7　分群多層系統架構

而在分群的演算法中，本研究所採用的是使用最普遍的 K-means 演算法，其方法為計算樣本間的歐式距離，並以此作為分群的依據，找出各群的群中心並對資料加以標籤；而由於特徵集當中，每一特徵的數值大小範圍不同，特徵又有連續性或離散性的差別，於是便需要先做正規化的動作，計算後之距離才不會有過大的差異。

正規化部分，主要依據”numeric”及”string”兩種特徵類型進行處理，對於”numeric”的資料，簡單的正規化取樣公式如下，即對於每個特徵值得數值範圍重新計算該值的大小。

(Value X – Attribute Min) / (Attribute Max – Attribute Min)

而對於”string”的特徵，則直接判斷 string 值是否相同，若不同則其距離為 1，相同則為 0。

```
if (string A == string B)
        return 0
else return 1;
```

而無論是攻擊種類多層或是分群多層，在混合自我學習機制後，系統的概念示意圖如下

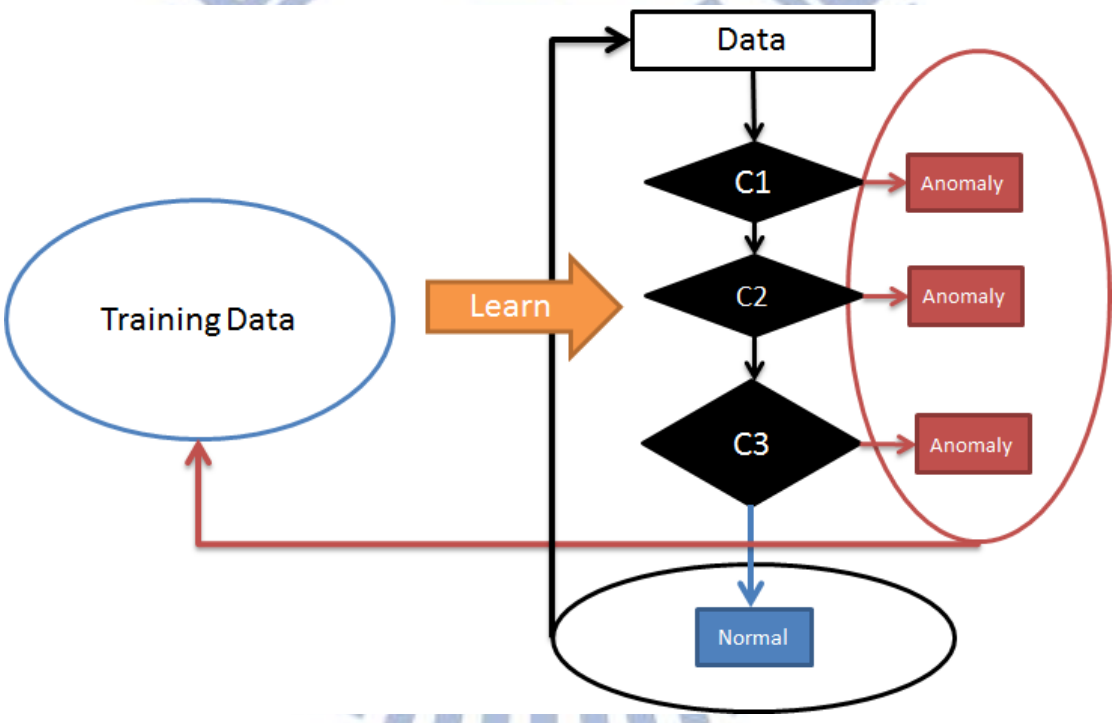


圖 8　混合多層及自我學習機制

3.3 投票(Voting)

在 Self-Learning 的過程中，等同於在每個階段皆訓練出不同的分類器(Classifier)，而每次訓練出的新分類器，其準確率可能升高，也可能降低，而為了避免最快情況的發生，便嘗試利用 Voting 的辦法來解決。

如圖 9 所示，隨著橫軸的訓練回合數不斷往後，可以看到每回合所訓練出的

分類器，其準確率會有不同，而此例子便是在利用了 Self-Learning 的方法後，所訓練出的新分類器，其能力及效益反而大幅下降的例子，反而原始的辨識準確率是最佳的，而會加入投票(Voting)的方法，即是為了避免此類狀況，導致分類器的辨識結果過於差勁。

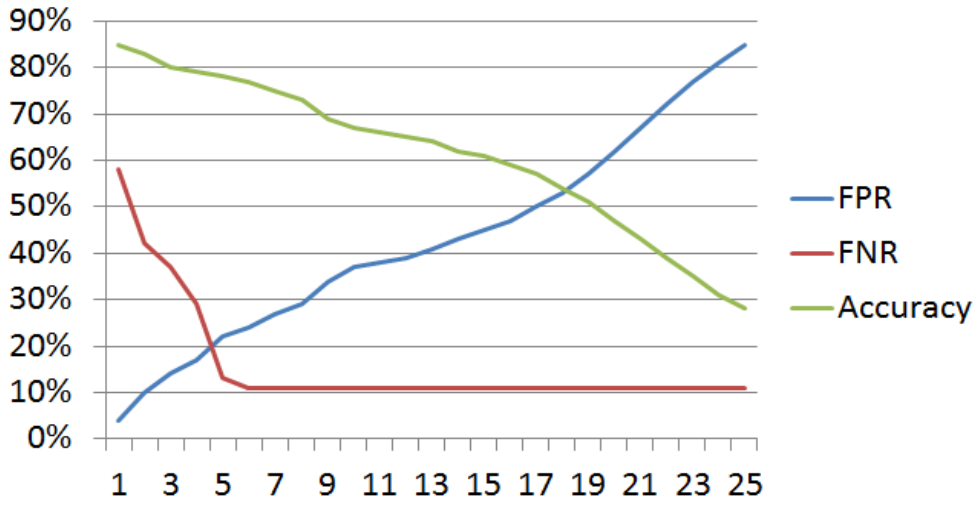


圖 9　Self-Learning 下，不同回合之分類器與分類結果

投票，就是利用每個回合所訓練出的分類器，來進行分類結果的統合整理，而非單純利用最終的分類器來進行分類辨識，如下所示，將要被辨識的樣本交給每個分類器進行辨識，再依照不同的分類器權值以及門檻值進行計算，最後才得到此樣本的辨識結果。

Voting Mechanism

Classifier: **C1,C2....Cn**
Classifying Result: **R1,R2....Rn**
Weight: **W1,W2....Wn**
Threshol: **T**

```
for each instance
    classify it by each classifier to obtain R1 ~ Rn
    Val = (R1 * W1) + (R2 * W2) + ....+(Rn * Wn)
    if (Val > T)
        label instance as positive
    else
        label instance as negative
```

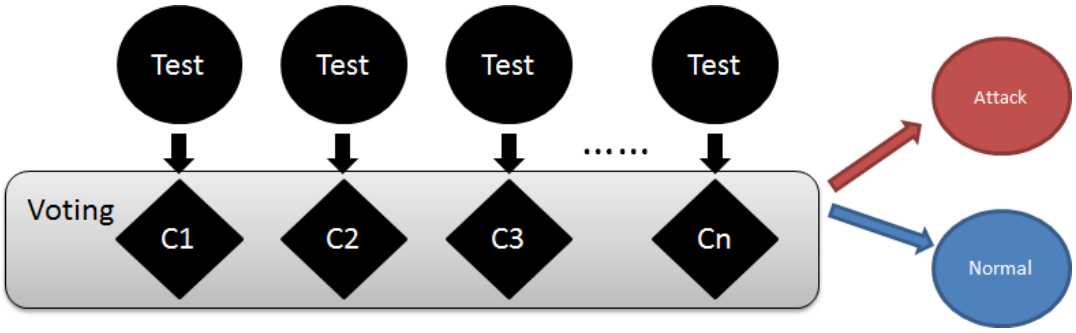


圖 10　Voting 概念示意圖

對於投票的演算法，重點在於每個分類器的權重(Weight)及門檻值之設置，這邊定義了三種 Weight 的分配方法以及兩種門檻值的定義方式，分別如下：

Weight

1. All equal to 1
   (1, 1…, 1)
2. Decreasing of training order
   (n, n-1… 2, 1)
3. Tune the factor x by (n+x) / (1+x)
   (n+x, n-1+x… 2+x, 1+x)

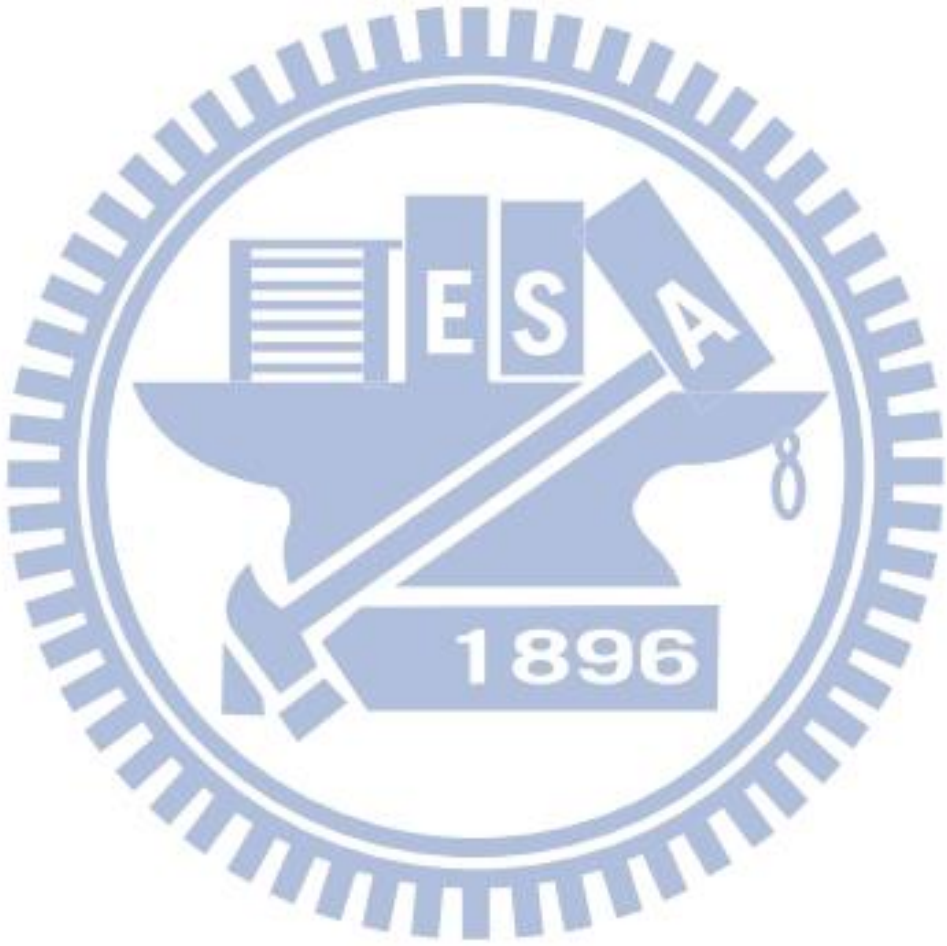
Threshold

1. Average of the weights
   (1+2+…+n)/n
2. Sum of the k smallest weights

Weight 的三種設置方法中，第一種是將每個分類器給予一樣的權重，依照分類器的平均情況來作為辨識的結果；第二種則是隨著 Self-Learning 的進行，越後訓練出的分類器，其權重越低，而原始的分類器之權重最高，此作法即較為重視初始的分類器，而對後續 Self-Learning 過後的分類器給予較低的權值，原因在於經 Self-Learning 後的新分類器之表現其不穩定性較大，雖可能有較高的攻擊偵測率，其誤判率也可能過高，此法特別適合用在辨識率較差的狀況下，但這樣的權值的設定也有缺點存在，即初始分類器與最終分類器之權重值過於懸殊，將使得後半段產生之分類器，在投票的過程中，幾乎處於被忽略的情形；而第三種方法，則是改進第二種方法，藉由 x 係數的調整，來決定初始與最終訓練出的兩個分類器之間的權值比重，避免過度忽略後期訓練出之分類器。

Threshold 的設定，第一為基本的平均方式，將所有的權值取平均作為門檻

值，若搭配皆為 1 的 weight 權重，就是基本的 Majority Voting 方式；第二種方式，則是可藉由調整 k 的參數，來設定門檻值的高低，改變的做法則可由調整 k 值，來決定是 k 個擁有最小權值的分類器總和來作為依據，更彈性化的設定門檻值。



四． 隱藏式馬可夫模型(Hidden Markov Model)

4.1 簡介(Brief Introduction)

隱藏馬可夫模型，是在語音辨識、手勢辨識上常見的機器學習演算法，其所描述的，便是藉由一可觀察的馬可夫鏈，來推測出另一我們所不知道的馬可夫鏈，即所謂的隱藏狀態。

首先，馬可夫鏈是用來表示一連串事件接續發生的機率，就像天氣的狀態，今天的天氣是冷或熱，也間接地影響到明天是天氣熱或是天氣冷的機率，而隱藏馬可夫模型，指的就是在這個過程中有些狀態被隱藏起來，是我們所不知道的，舉例來說，就好像我們無法知道天氣究竟是冷還是熱，我們只能看到窗外植物的葉子大小，而藉由紀錄每天窗外葉子大小的狀況，加上過去所知葉子大小跟天氣冷熱之間的關係，我們就有機會從葉子大小的紀錄中，去推測出這段時間的天氣變化。隱藏馬可夫模型，指的就是這兩個馬可夫鏈之間的關係，而我們所希望做到的，便是從可以觀察的、已知的現象，去推論出未知的狀態。

而 HMM 的限制，就在於觀測的現象必須是個序列，即馬可夫鏈，但在連線資料的樣本上，每個 Netflow 之間的關係都是獨立分開的，該如何去定義出馬可夫鏈的狀態，也就是連續性的 Feauture，則是個問題；而根據[9]當中所提到，惡意連線的封包之大小可能有些固定的數值，因此，這邊就嘗試利用每個 Netflow 樣本中，其每個封包的大小以及 Internal Arraival Time(IAT)來做為連續性的特徵資料，希望利用這些特徵，推論出隱性的馬可夫狀態，來分辨該 Netflow 是否為惡意攻擊或僅是一般連線。

本章接著將說明如何處理 Netflow 之連續性特徵來建立模型，以及如何利用 HMM 來對網路連線進行辨識的工作，至於 Hidden Markov Model 的相關細節，包括模型如何建立、 HMM 的三個 Problems，以及 Baum-Welch 演算法，M.Stamp 在[18] 中皆有詳盡的說明，這邊就不再加以贅述；而本研究中，程式的實作則參考自[18] 之 Pseudo Code 及[19] 之 Source Code。

```
public void train(int[] o, int steps) {
    int T = o.length;
    double[][] fwd;
    double[][] bwd;

    double pi1[] = new double[numStates];
    double a1[][] = new double[numStates][numStates];
    double b1[][] = new double[numStates][sigmaSize];

    for (int s = 0; s < steps; s++) {
        fwd = forwardProc(o);
        bwd = backwardProc(o);

        for (int i = 0; i < numStates; i++)
            pi1[i] = gamma(i, 0, o, fwd, bwd);

        for (int i = 0; i < numStates; i++) {
            for (int j = 0; j < numStates; j++) {
                double num = 0;
                double denom = 0;
                for (int t = 0; t <= T - 1; t++) {
                    num += p(t, i, j, o, fwd, bwd);
                    denom += gamma(i, t, o, fwd, bwd);
                }
                a1[i][j] = divide(num, denom);
```

圖 11　隱藏馬可夫模型之片段程式碼

4.2 特徵(Feature)

本節將說明如何將網路流量(Network Traffic)的特徵(Feature)轉換為隱藏式馬可夫模型之連續性觀測狀態(Continuous Observation)。首先，對於一個 Netflow 的連續性特徵，如 Packet Size 及 IAT，必須要先將其處理，從 Numeric 的 Feature Value 轉為 Discrete 之 Observation，而在這邊，預計將這些 Feature 分成五種 Observation，而分類的依據，則依照常態分佈的情況，加上平均值(Avg)及標準差(Std)來處理， 由於常態分佈在平均值正負一個標準差內，應占有全部的 68% 左右，於是這邊我們使用以下四個值作為界線，將連續性的特徵數值加以轉換，分別是:

- avg - std
- avg - 0.25*std
- avg + 0.25*std
- avg + std

依此四個門檻值將 Feature 轉成 0~5 的 Observation，如圖 12。

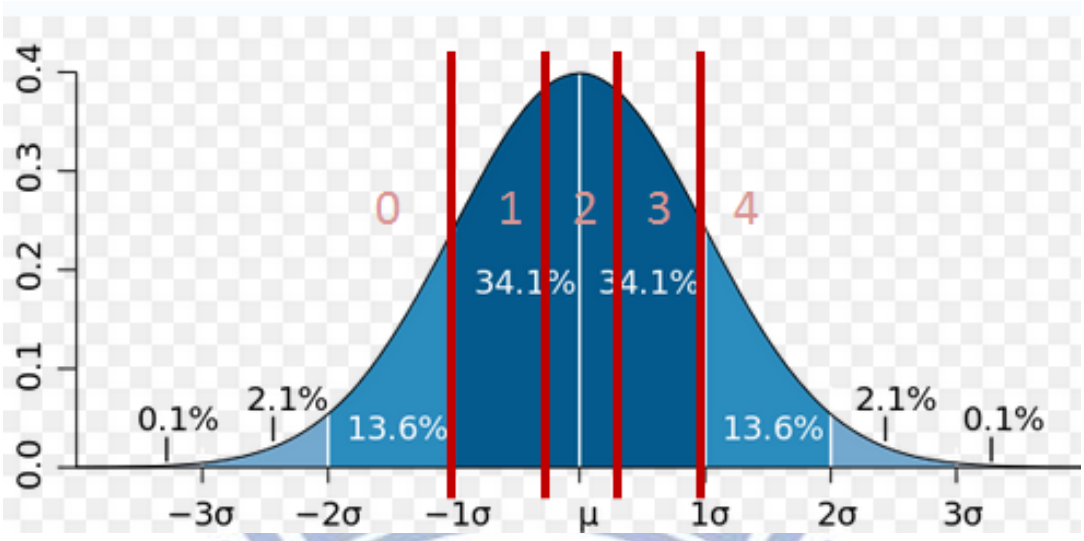


圖 12　常態分佈狀況與門檻值設定之特徵轉換狀況(底圖來源:[20])

但在此轉換方式下，由於平均值在減去標準差之後可能為負數，但如 Packet Size 及 IAT 等 Feature，皆不可能為負數，因此，需要針對此狀況做 Shift 之動作，簡單的處理便是將 avg-std 這個最小值作為 Shift 之大小，對四個門檻值作 Shift 之動作，則最小的門檻值則為 0，如此一來，門檻值的狀況調整將如下

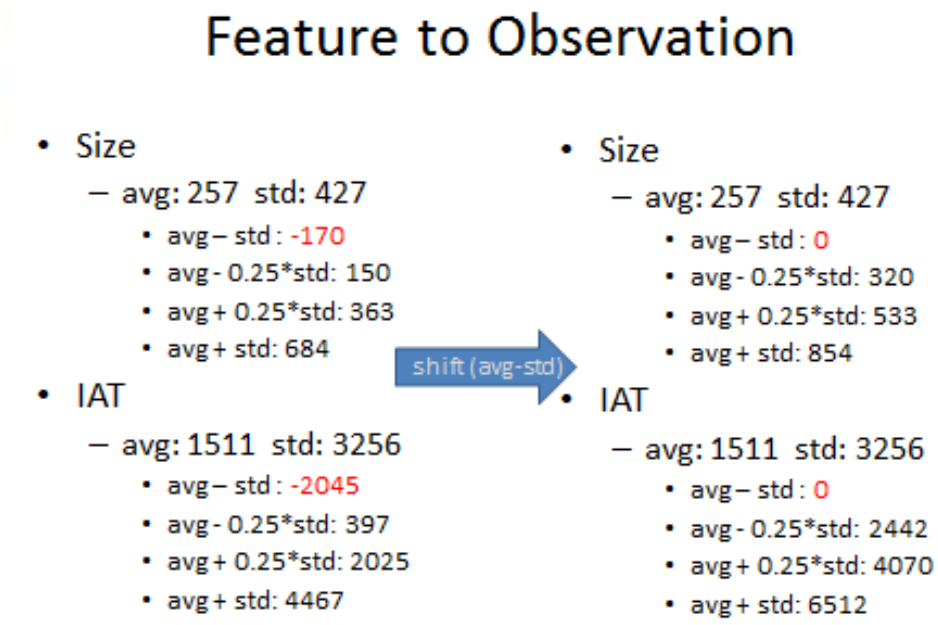


圖 13　Feature to Observation

利用程式進行批次的轉換情形則如下

```
46,46,46,131,46,62,66,62,84,77,84,79,78,46,90,1064,46,807,59,46,64,4
6,46,46,46
76,76
63,160
63,160
63,160
63,160
63,160
63,160
76,142
63,144
46,46,46,72,54,49,68,73,78,68,46,80,1500,1500,1500,1500,1500,46,46,9
32,1500,46,1500,1500,1500,46,1500,932,1500,723,46,46,46,73,46,95,46,
46,46,46,
```

圖 14　連續性的 Feature: Netflow 之 Packet Size

```
1111111111111114131111111
11
11
11
11
11
11
11
11
11
1111111111144444114414441443111111111
```

圖 15　由連續性 feature 轉換後之 Observation

4.3 運用方法(Usage)

利用訓練樣本建立隱藏馬可夫模型後，為了要利用該模型來進行辨識，在過程中，我們使用了兩種方法，一種是以樣本與模型相似性為基礎的方法，另一種則是以樣本的狀態變化來進行預測且判斷的方法。

4.3.1 相似性(Similarity)

根據樣本在此隱藏馬可夫模型的發生機率作為分辨(Classify)的依據，做法為先以一般正常(Normal)的訓練樣本來建立馬可夫模型，接著對於

測試樣本，將該樣本的 Observation Sequence 利用已建立的馬可夫模型，

來計算該樣本在這個馬可夫模型的出現機率

另一相反的做法，則是利用惡意(Anomaly)的樣本來建立隱藏馬可夫模型，再以該模型來判斷樣本的機率，若機率高於門檻值則判定為攻擊，若未高於門檻值，則判定為正常的流量。

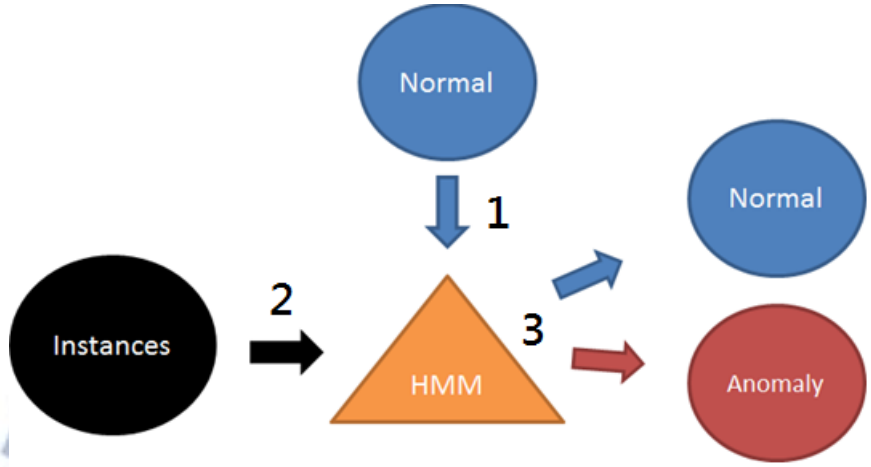


圖 16　利用馬可夫模型的相似性方法進行辨識

4.3.2 狀態模式(State Pattern)

狀態模式，則是根據樣本在隱藏馬可夫模型的 State 變動情況作為分辨的依據。首先利用訓練樣本建立馬可夫模型，接著對所有的惡意訓練樣本計算其最有可能的狀態變化，再統計這些惡意樣本經過各狀態的次數，然後就可以設定某 n 個最常在惡意樣本中出現的狀態為惡意狀態，而對測試樣本做辨識的方法，就是計算該測試樣本在這隱藏馬可夫模型中機率最大的狀態變化路徑，再計算其總過經過幾次被標記為惡意攻擊的狀態，若超過門檻值則標記為惡意攻擊。

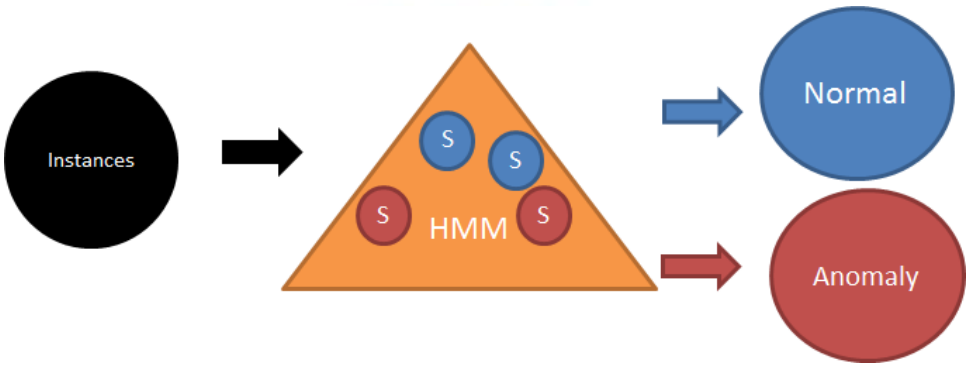


圖 17　隱藏馬可夫模型之狀態模式分辨方法

五、 資料集與工具(Datasets and Tool)

5.1. 總覽(Overview)

本論文用來做實驗評測的資料集，分別為 NSL-KDD，還有對 NSL-KDD 重新取樣調整後的 Modified NSL-KDD，以及由 DARPA 1999 的原始連線流量資料所轉換的資料集，詳細的說明如下。

NSL-KDD[12] 是由 KDD 99 資料集所修改而來，其主要修正了兩個問題，一是 KDD 99 中含有過多重複的樣本，這些過多的重複樣本導致了實驗結果評估上的失真，另一則是依據樣本的辨識難度將其分類，將辨識難度過低的樣本進行過濾。經過此兩樣修改後，使得此資料集有著相當大的改變，在減少重複的樣本後，訓練樣本的數量由 4,898,431 減少至 1,074,992，測試樣本的數量則由 311,027 減少至 77,289，再將辨識難度較低的樣本過濾後，訓練樣本剩下 125,973，測試樣本則剩下 22,544。而從作者的實驗數據顯示，如 Decision Tree 的實驗結果，辨識的 Accuracy 由 93%降低至 81%，代表修改過後的資料集，修正了過去舊的資料集之辨識過於容易，評測結果不夠客觀的問題，也使得實驗之數據更加有鑑別度與說服力。

Modified NSL-KDD 則是將 NSL-KDD 的資料重新取樣，提高其 Unknown Attack 比例後之資料集，以期更精確的評估系統對未知攻擊的偵測能力，在 5.2 中將有更詳細的說明；而 DARPA 1999 的原始流量資料，在經過轉換為 netflow 並提取出其特徵後，將包含有 flow 中連續性的特徵狀態，用途為建立第四章所提之 HMM 之用，至於如何使用工具轉換並提取特徵，將在 5.3 當中說明。5.4 則會說明各資料集的特徵定義。

所使用到的工具部分，則有 Softflowd 及 Weka，Softflowd 主要用於將原始連線流量資料轉換為 Netflow 的格式之用，修改程式碼後，除了轉換為 Netflow 格式外，一併直接將轉換為 Netflow 之 Insatnce 的特徵給匯出成檔案；Weka 則是實驗時所使用的機器學習工具，包含各種機器學習演算法，如決策樹、分群法等等，以及實驗結果的統計及圖像繪製之用，詳細部分將在後續說明。

5.2. 重新取樣之 NSL-KDD(Modified NSL-KDD)

在 P.Jongsuebsuk[7] 有提到，作者利用不同的實驗對照組，來驗證系統針對未知攻擊的偵測能力做評估，作者的做法是將樣本中 17 種不同的攻擊類型，每次任意取三種作為 Unknown Attack，將該類型的樣本從訓練集中移除，一共分為七組實驗組，來測試系統對於未知攻擊的偵測能力。

而本研究中用類似的方式，對於 NSL-KDD 的資料集做了更動，希望藉由此更動後的資料集，更進一步的探究系統對於未知攻擊的偵測能力。調整的辦法是將四種攻擊類型的樣本，在已標籤的訓練樣本中，都只留下該類型當中最多數量的攻擊，而將其餘的惡意攻擊樣本拿掉，因此，在做系統的測試時，大多數的攻擊樣本都是屬於未知的攻擊類型。下面分別列出 NSL-KDD 所含有的攻擊類型，有 * 號者表示在該類型攻擊僅在測試樣本中出現，訓練樣本中沒有該類型的攻擊，即為 Unknown Attack，有 & 號者則表示該攻擊類型僅在訓練樣本中出現。

DoS(6+4)	
back	teardrop
land	*apache2
neptune	*mailbomb
pod	*processtable
smurf	*udpstorm
Probe(4+2)	
ipsweep	satan
nmap	*mscan
portsweep	*saint
U2R(4+4)	
buffer_overflow	*httptunnel
loadmodule	*ps
perl	*sqlattack
rootkit	*xterm
R2L(8+7)	
ftp_write	*sendmail
guess_passwd	*named
imap	*snmpgetattack
multihop	*snmpguess
phf	*xlock
&spy	*xsnoop
&warezclient	*worm
warezmaster	

表 2　NSL-KDD 中的攻擊一覽

而在更動後，訓練樣本中僅留下表 3 中的四種攻擊作為訓練集中之攻擊樣本，剩下的攻擊類型都將移除，增加未知攻擊所占的比例擊數量，以求測試系統偵測未知攻擊的能力，另外在圖 18 中，也列出更動前後，Unknown Attack 在資料集中所占比例狀況。

DoS	Neptune
U2R	buffer_overflow
R2L	warezclient
Probe	satan

表 3　各種類的攻擊中其樣本數最多之攻擊

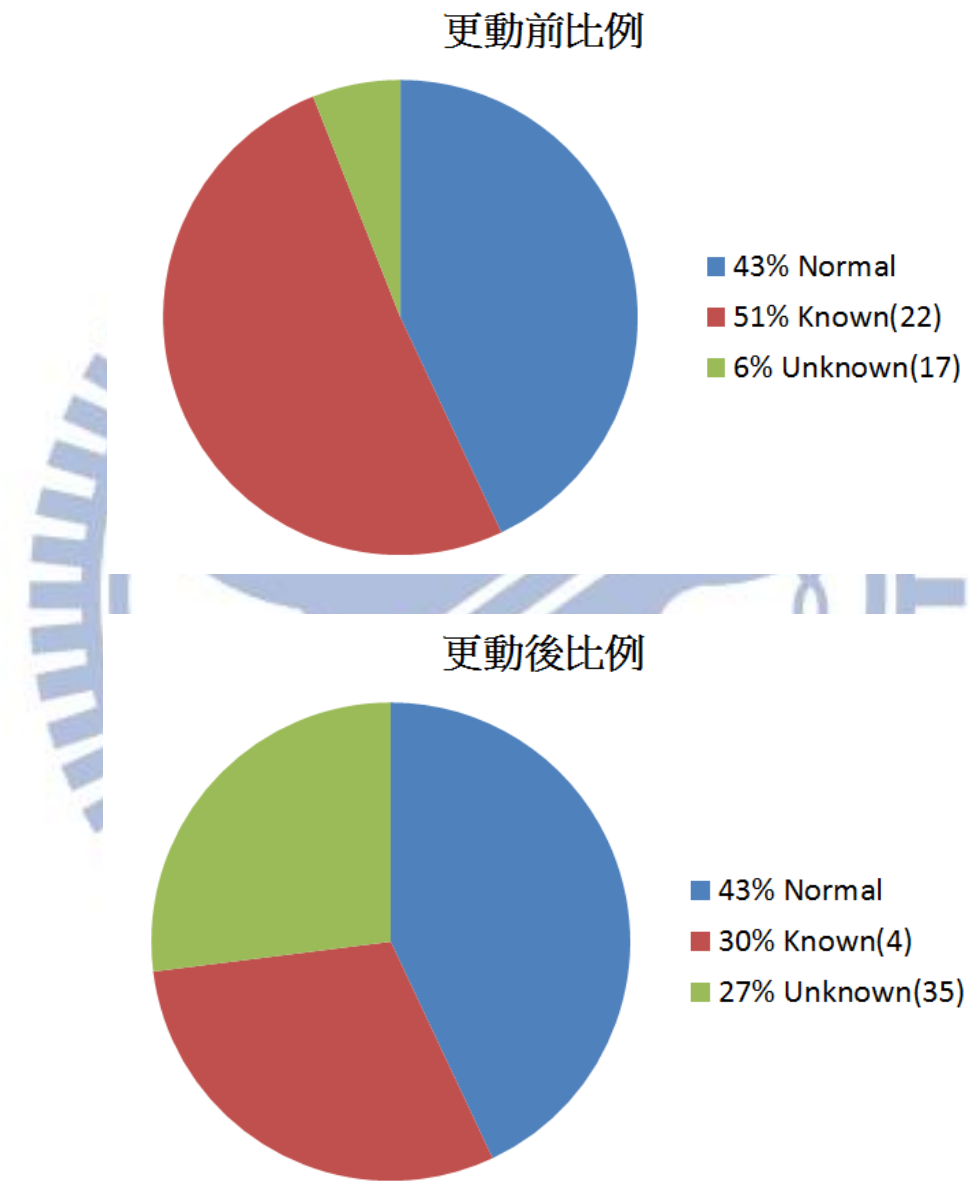


圖 18　資料集更動前後之所占比例狀況

5.3. 原始流量資料轉換(Raw Traffic Transformation)

這邊的原始流量資料轉換，即是將連線流量的 Raw Data，轉換成 Netflow 的格式，再對個別的 Netflow，提取出他們的特徵資料，接著以這些特徵資料作為

機器學習的模型建立之用，而如何轉換並提取出 Netflow 之特徵值，就是本節的主要內容。

5.3.1 網路流(Netflow)

Netflow 是由 Cisco 所定義的格式，是將在某段時間內，擁有相同連線狀態如 IP、Port、Protocol 的所有連線資料統整，將其視為一個 Flow，比起鬆散零碎的封包資料，更方便對一些服務或特定用途之連線做到控管以及處理之用，本研究所採用的是 Netflow V5 的格式，其 Flow 由以下七個連線資料來做處理。

1. Ingress interface (SNMP ifIndex)
2. Source IP address
3. Destination IP address
4. IP protocol
5. Source port for UDP or TCP, 0 for other protocols
6. Destination port for UDP or TCP, type and code for ICMP, or 0 for other protocols
7. IP Type of Service

而在將連線資料處理後，Version 5 的 Header 及 Format 如圖 19 所示

byte 3	byte 2	byte 1	byte 0

Version 5 Flow Header

version		count
sysUptime		
unix seconds		
unix nanoseconds		
flow sequence number		
engine type	engine ID	reserved

Version 5 Flow Entry

source IP address	
destination IP address	
next hop IP address	
input interface index	output interface index
packets	
bytes	
start time of flow	
end time of flow	
source port	destination port
pad / TCP flags	IP protocol / TOS
source AS	destination AS
src netmask length / dst netmask length	padding

圖 19　Cisco 所定義的 Netflow Version5 format(來源:[21])

5.3.2 Softflowd 工具

Softflowd[22] 此一開放原始碼工具其作用為將由測錄工具所紀錄之原始網路流量資料，如 nfdump、pcap 檔案格式之資料，轉換為 Netflow 之資料型態，而其轉換的依據，則是根據 Cisco 的 netflow 定義，並依照其格式將資料匯出，轉換前及轉換後的資料如以下圖示。

No.	Time	Source	Destination	Protocol	Length	Info
213	1.062299	192.168.1.104	140.113.207.128	UDP	865	Source port: 57083 Destination port: 40811
214	1.067060	140.113.207.128	192.168.1.104	UDP	95	Source port: 40811 Destination port: 57083
215	1.075119	140.113.207.128	192.168.1.104	UDP	81	Source port: 40811 Destination port: 57083
216	1.105150	192.168.1.104	140.113.207.128	UDP	162	Source port: 57083 Destination port: 40811
217	1.105800	192.168.1.104	140.113.207.128	UDP	1066	Source port: 57083 Destination port: 40811
218	1.106058	192.168.1.104	140.113.207.128	UDP	452	Source port: 57083 Destination port: 40811
219	1.107266	140.113.207.128	192.168.1.104	UDP	84	Source port: 40811 Destination port: 57083
220	1.138962	192.168.1.104	140.113.207.128	UDP	125	Source port: 57083 Destination port: 40811
221	1.154445	140.113.207.128	192.168.1.104	UDP	81	Source port: 40811 Destination port: 57083
222	1.192380	140.113.207.128	192.168.1.104	UDP	81	Source port: 40811 Destination port: 57083
223	1.278044	140.113.207.128	192.168.1.104	UDP	82	Source port: 40811 Destination port: 57083
224	1.304050	140.113.207.128	192.168.1.104	UDP	81	Source port: 40811 Destination port: 57083
225	1.304360	192.168.1.104	140.113.207.128	UDP	66	Source port: 57083 Destination port: 40811
226	1.305820	192.168.1.104	140.113.207.128	UDP	84	Source port: 57083 Destination port: 40811
227	1.309216	140.113.207.128	192.168.1.104	UDP	82	Source port: 40811 Destination port: 57083

```
⊞ Frame 224: 81 bytes on wire (648 bits), 81 bytes captured (648 bits)
⊞ Ethernet II, Src: Tp-LinkT_ba:f4:d6 (5c:63:bf:ba:f4:d6), Dst: AskeyCom_3e:3b:8b (b4:82:fe:3e:3b:8b)
⊞ Internet Protocol Version 4, Src: 140.113.207.128 (140.113.207.128), Dst: 192.168.1.104 (192.168.1.104)
⊞ User Datagram Protocol, Src Port: 40811 (40811), Dst Port: 57083 (57083)
⊞ Data (39 bytes)
```

```
0000  b4 82 fe 3e 3b 8b 5c 63  bf ba f4 d6 08 00 45 00   ...>;.\c ......E.
0010  00 43 7e 42 40 00 3a 11  a4 65 8c 71 cf 80 c0 a8   .C~B@.:. .e.q....
0020  01 68 9f 6b de fb 00 2f  37 3d e1 b8 01 00 4e 64   .h.k.../ 7=....Nd
0030  04 00 82 88 0b 17 24 6b  17 00 0c 00 00 00 f5 a4   ......$k ........
0040  01 00 07 00 00 00 82 05  e9 52 5a 0d 38 66 42 4f   ........ .RZ.8fBO
```

圖 20　tcpdump file 之 raw data

```
Date flow start         Duration Proto     Src IP Addr:Port          Dst IP Addr:Port   Flags Tos  Packets
2013-12-05 18:49:07.270    1.522 UDP    140.113.207.128:40811 ->   192.168.1.104:57083 ......   0       70
2013-12-05 18:49:07.270    1.522 UDP     192.168.1.104:57083 ->  140.113.207.128:40811 ......   0      153
2013-12-05 18:49:07.376    0.846 TCP     119.235.235.91:443   ->   192.168.1.104:19588 .A....   0        2
2013-12-05 18:49:07.376    0.846 TCP     192.168.1.104:19588 ->   119.235.235.91:443   .AP...   0        3
Summary: total flows: 4, total bytes: 115419, total packets: 228, avg bps: 606670, avg pps: 149, avg bpp: 506
Time window: 2013-12-05 18:49:07 - 2013-12-05 18:49:08
Total flows processed: 4, Blocks skipped: 0, Bytes read: 236
Sys: 0.000s flows/second: 0.0       Wall: 0.000s flows/second: 15267.2
```

圖 21　經由 softflowd 轉換後之 netflow 資料

5.3.3 將 DARPA 之原始流量資料進行特徵擷取

本研究中，利用了 Softflowd 對原始連線資料進行 Netflow 處理，而配合修改其原始碼，將可以一併在處理為 Netflow 時，將個別 Netflow 之特徵匯出成特徵資料集，將整理後的 Netflow 資料傾印出其特徵資料，如 Packet 數、Bytes 大小等等特徵資料，皆可在 softflowd 對個別封包連線進行過濾處理時，一併處理匯出。另外為了配合建立第四章所提的 HMM，在匯出特徵值時，也特地將 Netflow 中，每一個 Packet 之大小及 Internal Arrival Time(IAT)也一併匯出，作為連續性之特徵作為建立 HMM 所用。

```
printf("Flow:%d\t", i);
printf("%15s:%5hu <> %15s:%5hu\t", FLOW_INFO[i].addr[0], ntohs(FLOW_INFO[i].port[0]),
        FLOW_INFO[i].addr[1], ntohs(FLOW_INFO[i].port[1]));
printf("Packets\tin:%d out:%d sum:%d\n",FLOW_INFO[i].packets[0],
        FLOW_INFO[i].packets[1], FLOW_INFO[i].packets[2]);
printf("%d\t",(int)((double)FLOW_INFO[i].start.tv_sec+(double)FLOW_INFO[i].start.tv_usec / 1000000.0));
flowtime.tv_sec = (int)((double)FLOW_INFO[i].start.tv_sec+(double)FLOW_INFO[i].start.tv_usec / 1000000.0);
flowtime.tv_sec = (int)((double)flowtime.tv_sec- 46800.0);//week4 46800  week5 43200
printf("%s", ctime(&flowtime.tv_sec));
// common feature
tmp = (double)FLOW_INFO[i].last.tv_sec +
        ((double)FLOW_INFO[i].last.tv_usec / 1000000.0);
tmp -= (double)FLOW_INFO[i].start.tv_sec +
        ((double)FLOW_INFO[i].start.tv_usec / 1000000.0);
if (tmp < 0.0)
        tmp = 0.0;

printf("%d,",tmp);//duration
printf("%d,", FLOW_INFO[i].proto);//protocol
printf("%d,%d,",FLOW_INFO[i].packets[0], FLOW_INFO[i].packets[1]);//src,dst packets
printf("%d,", FLOW_INFO[i].packets[2]); // packets
 //printf("%d"); // packets-payload

printf("%d,%d,",FLOW_INFO[i].foctets[0], FLOW_INFO[i].foctets[1]);//src,dst bytes
printf("%d,",FLOW_INFO[i].foctets[2]); //bytes
//pay-bytes
```

圖 22　Softflowd 中之部分新增程式碼

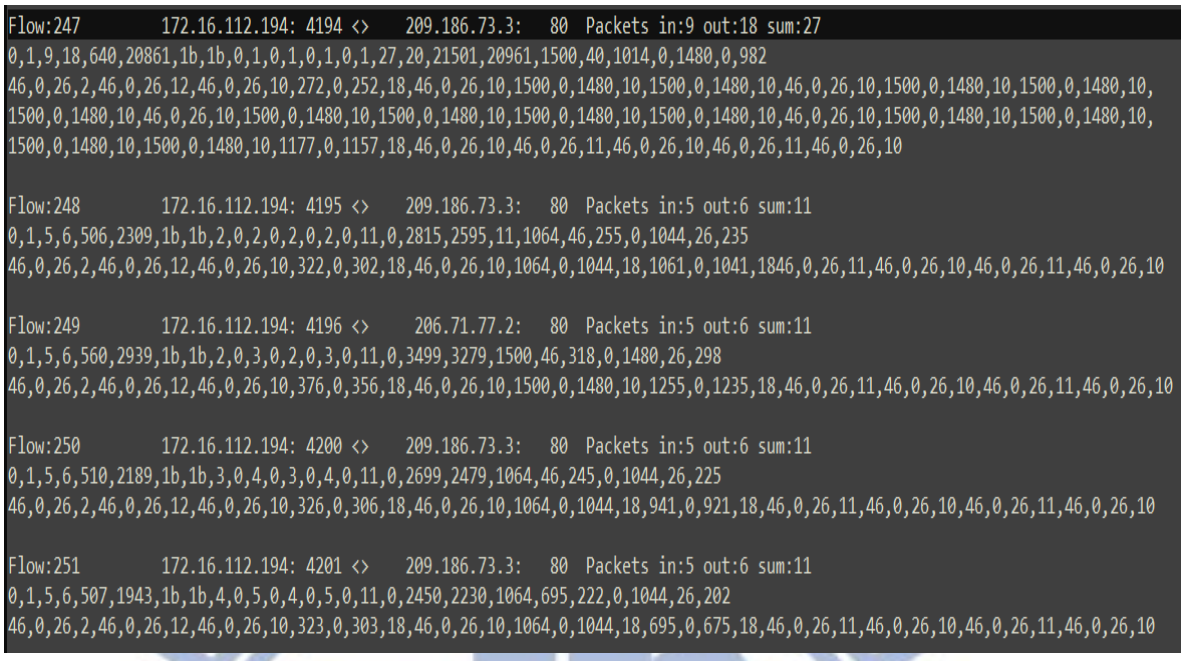


圖 23 使用修改後之 Softflowd 所印出之 flow 及其特徵資料集

5.4. 特徵(Feature)

特徵值是用來提供機器學習演算法建立模型之用的數據，而在本研究中主要有兩組的特徵值，一組為 NSL-KDD 原先即定義好之特徵，另一組則為自行定義的 DARPA 特徵值，由於 DARPA 僅為原始的流量連線資料，在轉換為 netflow 後，我們參考了其他資料集的特徵並自行定義了 DARPA 的特徵，除了基本的 Packets、Bytes、Protocol 等等，另外也加入了 Time-based 及 Connection-based 的 Feature，用來判斷過去某段時間或一定數量的連線數時，重複的 IP 及 Port 有哪些，最後，也為了 HMM 建立模型所用，定義了連續性之 Packet Feature，匯出一個 Flow 中每個 Packet 大小及 Internal Arrival Time(IAT)。

5.4.1 NSL-KDD

根據官方所給的特徵，可分為三種主要類別，分別為 Basic feature、Content features 以及 Traffic features，其特徵名稱及描述如下。

feature name	*description*	*type*
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http,	discrete

	telnet, etc.	
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous

表 4　Basic features

feature name	*description*	*type*
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete

表 5　Content features

feature name	*description*	*type*
count	number of connections to the same host as the current connection in the past two seconds	continuous
serror_rate	% of connections that have ``SYN'' errors	continuous
rerror_rate	% of connections that have ``REJ'' errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
srv_serror_rate	% of connections that have ``SYN'' errors	continuous
srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

表 6 Traffic features

5.4.2 DARPA

DARPA 資料集只是單純的原始連線流量資料，本研究參考 NDS-KDD 以及 P.Gogoi[23] 的研究中所定義的特徵，在這些原始的流量資料轉換為網路流(Network Flow)後，對每個樣本處理出下表所列的特徵值。

其中比較特別的是從編號 30 之後的特徵值，為該 Netflow 中，第一個封包的大小、內部到達時間，接著是第二個封包的大小、內部到達時間，依序直到第 n 個封包為止，此連續性的封包特徵是作為第四章的隱藏馬可夫模型建立之用。

No.	Feature name	Description	Type
1	duration	length of the flow	continuous
2	protocol-type	type of the protocol, e.g. tcp, udp, etc.	discrete
3	src-packets	number of packets from source to destination	continuous
4	dst-packets	number of packets from destination to source	continuous

5	#pkts	number of all packets	continuous
6	#pkt-p	number of all packets with payload	continuous
7	src-byte	number of data bytes from source to destination	continuous
8	dst-byte	number of data bytes from destination to source	continuous
9	bytes	number of all data bytes	continuous
10	pay _bytes	number of all payload bytes	continuous
11	maxsz	maximum size of all packets	continuous
12	minsz	minimum size of all packets	continuous
13	avgsz	average size of all packets	continuous
14	stdsz	standard Deviation of all packets	continuous
15	maxpy	maximum size of all payload	continuous
16	minpy	minimum size of all payload	continuous
17	avgpy	average size of all payload	continuous
18	stdpy	standard Deviation of all payload	continuous
19	avgIAT	average IAT of all packets	continuous
20	src_flag	connection flags from source to destination	continuous
21	dst _flag	connection flags from destination to source	continuous
22	srcip-time	number of flows from the same src IP in the last T sec	continuous
23	dstip-time	number of flows to the same dst IP in the last T sec	continuous
24	srcport-time	number of flows from the same src port in the last T sec	continuous
25	dstport-time	number of flows to the same dst port in the last T sec	continuous
26	srcip-conn	number of flows from the same src IP in the last N flows	continuous
27	dstip-conn	number of flows to the same dst IP in the last N flows	continuous
28	srcport-conn	number of flows from the same src port in the last N flows	continuous

29	dstport-conn	number of flows to the same dst port in the last N flows	continuous
30	szX_1	size of X_1 packet	continuous
31	$IATX_1$	IAT of X_1 packet	continuous
32	szX_2	size of X_2 packet	continuous
33	$IATX_2$	IAT of X_2 packet	continuous
34	szX_3	size of X_3 packet	continuous
…	…	…	continuous

表 7　DARPA 轉換之 Feature 一覽

5.5. 機器學習工具(Weka)

目前有許多的開源機器學習工具可供使用，而本研究中所使用的是 weka[24] 這套工具，主因在於其完整性、方便性，以及本身對於 Java 的熟悉程度較高，而與 Weka 同樣較為完整的開源工具，如 Orange[25] ，則是以 python 語言開發，另外也有許多相關研究是以 R 語言來對資料進行處理及模型的建立，這邊則不再加以討論。

5.5.1 Weka 介紹

Weka 為 University of Waikato 的 Machine Learning Group 所開發的開放原始碼軟體，以 Java 實作了完整的機器學習方法，包含有 Data 的 Pre-Processing、 Classification, Regression, Clustering, Association rules, Visualization 等各類的功能及演算法實作，除了提供給一般使用者的圖形介面及命令列模式，更提供該函式庫的 Java API 作為自行撰寫程式碼之用，另由於 Weka 為開放原始碼軟體，若使用者有自身的任何需求，也可透過自行修改其原始碼，來作為編寫開發機器學習演算法之用。

圖 24　weka 之標誌

5.5.2 ARFF 檔案格式

Attribute-Relation File Format(ARFF)[26] 為 Weka 所用的資料集檔案格式，以 ASCII 編碼的文字來記錄資料集當中的樣本及其特徵。其主要分為兩個部分，一為 Header 區塊，另一則為 Data 區塊。Header 區塊主要由"@RELATION"及"@ATTRIBUTE"組成。"@RELATION"用來表示該檔案的資料集記錄與何相關的資訊，與檔名的意義雷同，對於資料的處理上不會有影響，純粹是方便使用者辨識資料集意義之用；而"@ATTRIBUTE"則用來宣告在此資料集中，每個樣本的各項特徵名稱及其表示格式，如圖 25 中，第二列指的便是每個樣本會有"duration"這項特徵，而該特徵的格式為 real。特徵可宣告的格式共可分為以下四種，分別為 numeric、<nominal-specification>、string、date，關於格式的資訊在[26] 中有更詳細的說明。

而 Data 區塊，每一列即代表一個樣本的特徵值，每個特徵間以逗號間隔，其特徵的次序，即是@attribute 區塊的排列次序，依次列出該樣本的每一個特徵值，直到最後一個樣本為止。

```
@relation 'normalfeature_normal'
@attribute 'duration' real
@attribute 'protocol_type' real
@attribute 'src-packets' real
@attribute 'dst-packets' real
@attribute '#pkts' real
@attribute 'src-byte' real
@attribute 'dst-byte' real
@attribute 'bytes' real
@attribute 'srcport-conn' real
@attribute 'dstport-conn' real
@attribute 'class' {'0','1'}
@data
0,2,1,1,2,76,76,152,76,76,76,0,0,0,0,0,0,0,0,0,0,0,0,0
79152,2,2640,0,2640,137280,0,137280,52,52,52,0,55,0,0,0
79152,2,2643,0,2643,138736,0,138736,72,52,52,0,55,0,0,0
79153,2,1237,1237,2474,94012,94012,188024,76,76,76,0,62
79154,2,2844,0,2844,204768,0,204768,72,72,72,0,47,0,0,0
79073,2,315,0,315,79514,0,79514,310,203,252,20,3345,0,0
32353,2,74,155,229,8218,16566,24784,163,60,108,23,657,0
```

圖 25　ARFF 檔案格式內容

5.5.3 圖形介面與 API

Weka 提供多元的使用方式，本論文中除了圖形介面的基本使用，為了實現第三章所提到的方法，最後是搭配 Weka 所提供的 API，自行撰寫 Java 的程式碼，而其 API 在[27] 中，包括函數的型態，傳回值，繼承方式等，都有著詳細的文件說明，可以藉由 API，輕易的自行撰寫機器學習程式碼。

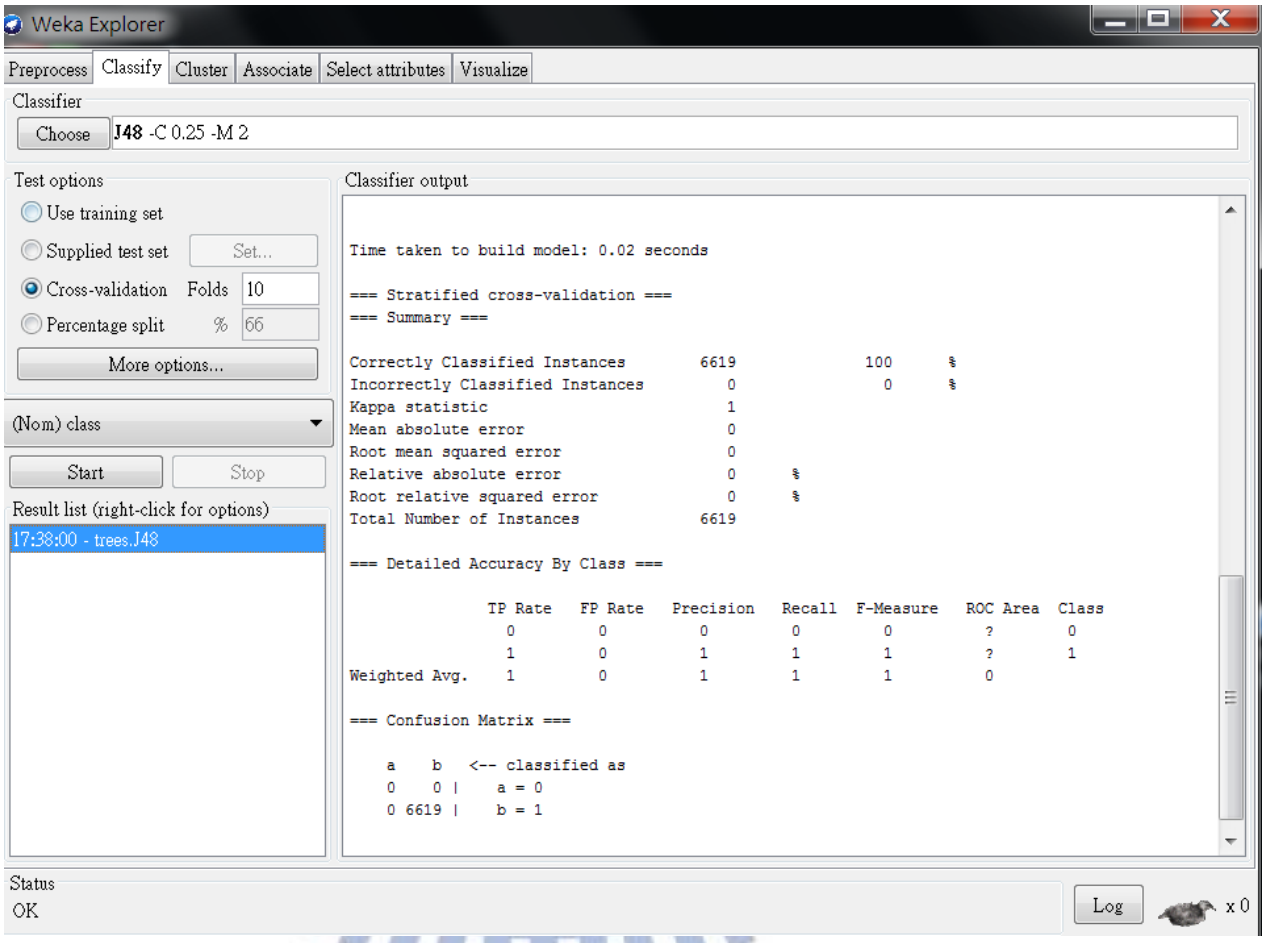


圖 26 Weka 之圖形使用介面

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.Serializable;

import weka.classifiers.rules.JRip;
import weka.filters.Filter;
import weka.classifiers.Classifier;
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.bayes.BayesNet;
import weka.classifiers.trees.J48;
import weka.core.Instance;
import weka.core.InstanceComparator;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class fiveLevel {
    public static void main(String [] argv) throws Exception{

        //Set Options
        Boolean onlineLearning = true;
        Boolean addToAllTrain = false;
        int trainingLimit = 5;

        Date timeStart = new Date(); //Get the beginning time of program

        System.out.println("----- Reading ARFF -----");

        // Dos train
        DataSource source1 = new DataSource("C:\\Users\\rex852753\\Desktop\\cluster\\5\\seed10\\5Cluste
        Instances train1 = source1.getDataSet();
        train1.setClassIndex(train1.numAttributes() - 1);
```

圖 27　使用 Weka API 撰寫之程式碼

六、 實驗結果(Experiment)

本章首先將介紹分類辨識上的各種評估指標及其意義，以方便後續的結果圖表閱讀，而後將統整前面各所介紹的方法所做出來的辨識結果，並就實驗的結果與其他研究之結果進行比較與分析討論。

6.1. 評估指標介紹

- True Positive(TP):
  代表將實際上為 Positive 的樣本標示為 Positive。

- True Negative(TN):
  代表將實際上為 Negative 的樣本標示為 Negative。

- False Positive(FP):
  代表將實際上為 Negative 的樣本標示為 Positive。

- False Negative(FN):
  代表將實際上為 Positive 的樣本標示為 Negative。

- Confusion Matrix:
  將上述的 TP、TN、FP、FN 分別以四個象限表示，可以清楚的看到辨識的結果，有哪些 Positive 的樣本分類正確、那些錯誤，Negative 的樣本亦同，如表 8。

Predicted Class	Actual Class	
	Positive	Negative
Classified as Positive	True Positive	False Positive
Classified as Negative	False Negative	True Negative

表 8　Confusion Matrix Example

- Precision:
  TP / (TP+FP)，用來表示*被辨識*為 Positive 的樣本之辨識正確性率。

- True Positive Rate(TPR) or Recall:
  TP / (TP+FN)，用來表示實際為 Positive 樣本之辨識正確率。

- False Positive Rate(FPR):
  FP / (FP+TN)，用來表示實際為 Negative 的樣本之誤判率。

- Accuracy(ACC):
  (TP+TN) / (TP+FP+TN+FN)，用來表示此資料集之正確辨識率。

- F1 Score:
  2TP/(2TP+FP+FN)，可用來同時表示 Precision 及 Recall 其相關關係的的參數。

- ROC Curve and Area Under the Curve(AUC):
  ROC Curve 為一用來判斷辨識結果的方法，以二維線性圖表示，其 x 軸為 FPR，y 軸為 TPR，而 AUC 則是表示在繪製出的 ROC Curve 上，在曲線下方的面積區域所占越大，代表系統的 TPR 越高，FPR 越低，也就代表其辨識的結果越好，如圖 28。

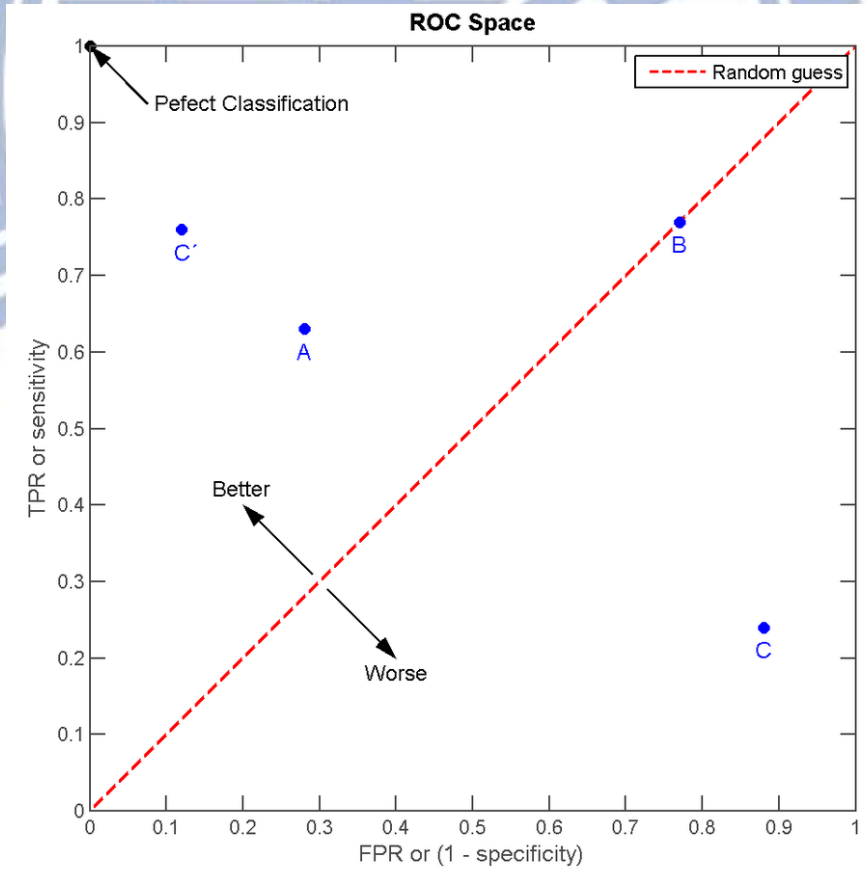


圖 28　Example of ROC Curve and AUC(來源:[28])

6.2. 自我學習方法(Self-Learning)

這邊將列出有無使用 Self-Learning 在辨識結果上的差異,使用的資料集包含 NSL-KDD、Modified NSL-KDD、以及由 DARPA 所轉換的 Dataset，而使用的演算法上，則以 Decision Tree(J48)為主，並分析其結果。

首先列出使用 Decision Tree 的演算法下，是否使用 Self-Learning 所做實驗之 Confusion Matrix

NSL-KDD	Decision Tree (J48)	
	Actual Class	
Predicted Class	Positive	Negative
Classified as Positive	8933	263
Classified as Negative	3900	9448

表 9　Confusion Matrix @NSL-KDD

NSL-KDD	Decision Tree (J48) with Self-Learning	
	Actual Class	
Predicted Class	Positive	Negative
Classified as Positive	11536	395
Classified as Negative	1297	9316

表 10　Confusion Matrix @NSL-KDD

Modified NSL-KDD	Decision Tree (J48)	
	Actual Class	
Predicted Class	Positive	Negative
Classified as Positive	6466	158
Classified as Negative	6367	9553

表 11　Confusion Matrix @Modified NSL-KDD

Modified NSL-KDD	Decision Tree (J48) with Self-Learning	
	Actual Class	
Predicted Class	Positive	Negative
Classified as Positive	8440	304
Classified as Negative	4393	9407

表 12　Confusion Matrix @Modified NSL-KDD

DARPA	Decision Tree (J48)	
	Actual Class	
Predicted Class	Positive	Negative
Classified as Positive	10390	30809
Classified as Negative	149894	641194

表 13　Confusion Matrix @DARPA

DARPA	Decision Tree (J48) with Self-Learning	
	Actual Class	
Predicted Class	Positive	Negative
Classified as Positive	138956	130530
Classified as Negative	21328	541473

表 14　Confusion Matrix @DARPA

接著針對以上的結果，計算其各項數值，統計如表 15 所示。

Dataset	NSL-KDD		Modified NSL-KDD		DARPA	
Algorithm	J48	J48 with Self -learning	J48	J48 with Self -learning	J48	J48 with Self -learning
Precision	**0.97**	0.96	**0.97**	0.96	0.25	**0.51**
TPR	0.69	**0.89**	0.50	**0.65**	0.06	**0.86**
FPR	**0.02**	0.04	**0.01**	0.03	**0.04**	0.19
Accuracy	0.81	**0.92**	0.71	**0.79**	0.78	**0.81**
F1 score	0.81	**0.93**	0.66	**0.78**	0.1	**0.64**

表 15　實驗結果之數據統計

可以看到在加入了 Self-Learning 的機制後，整體的實驗結果，雖然在 FPR 上會有些許的上升，但因 TPR 的急遽上升，也就是對攻擊的辨識捕捉率提高，連帶地使得整體的辨識結果來的更好，使 Accuracy 也都是呈現上升的狀況，而其改進的原因，便在於藉由 Self-Learning 的過程中，藉由已知的模型作為辨識基礎，將更多未知的樣本加入到訓練樣本中，使得訓練集中的攻擊樣本之數量及多樣性都獲得增加，連帶使得後續訓練出的分類器對攻擊將擁有更好的辨識能力，而由 Modified NSL-KDD 的結果中也可看出，即使在原始的攻擊訓練樣本數量缺乏下，利用此機制，仍然可以使攻擊偵測率提高許多，也可證明 Self-Learning 在針對 Unknown Attack 的偵測上，有著一定程度上的改進。

圖 29 及圖 30 則是有無使用 Self-Learning 機制在 NSL-KDD 資料集上的 ROC Curve 比較，可明顯的看出加入 Self-Learning 後表現上升許多，Area under Curve

也來的大上許多，藉由加入 Self-Learning 機制可顯著的提升 J48 之效益。

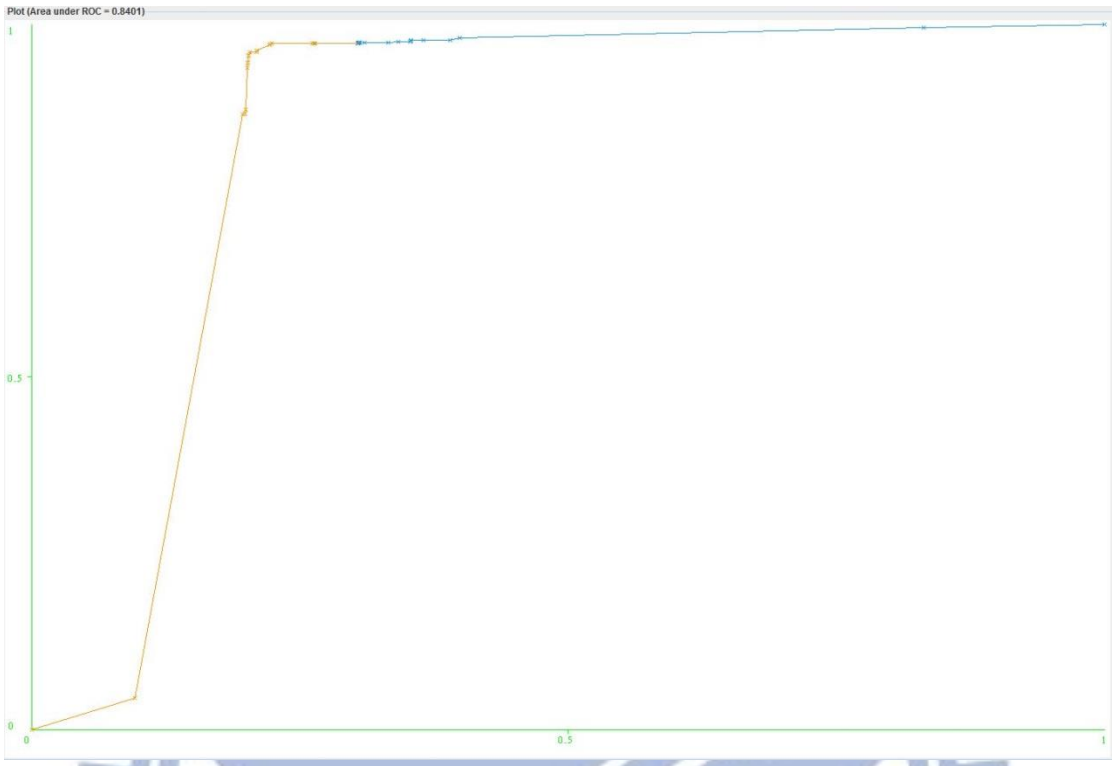


圖 29　ROC Curve and AUC of J48 @NSL-KDD

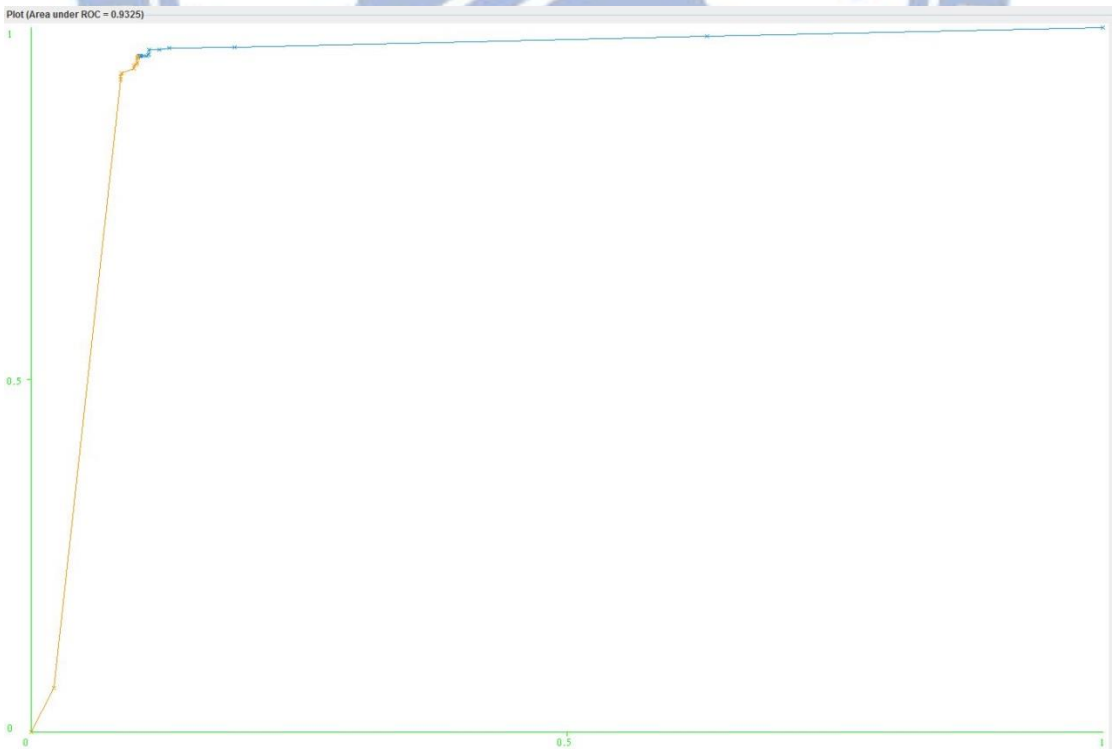


圖 30　ROC Curve and AUC of J48 with Self-Learning @NSL-KDD

另外，在表 16 中，也列出一些同樣以 NSL-KDD 作為實驗資料集的研究，對其提供的結果數據加以比較，也可看到無論是 True Positive Rate 或是 False Positive Rate，使用 Self-Learning 之數據皆比其他研究之結果要來的好。

Approach	TPR	FPR
One-class SVM(Inner product) [30]	0.60	0.48
Anomalous Payload-based IDS [31]	0.60	0.09
SOM& NN [32]	0.79	-
Local Outlier Factor [33]	0.66	-
Bayes Estimators [34]	0.72	0.068
Gaussian Field [35]	0.53	0.40
Linear discriminant functions [8]	0.83	0.09
1-Class NN [8]	0.8	0.17
J48 with Self-learning	**0.89**	**0.04**

表 16　NSL-KDD 之實驗結果比較

對於 Self-Learning 應用在不同演算法上的比較，則以 NSL-KDD 作為實驗資料集，將結果列於下表 17。而由當中的數據可以看到，Self-Learning 的方式，在對於 Decision Tree 類型的演算法，如 J48、Random Forest 等等，皆有著良好的效率提升以及辨識能力之改善，相反的，於 Bayes 類型之演算法，則沒有什麼改善甚至沒有變動，主要是由於其演算法的作法，是利用假設各個為獨立的 Feature 與類別之間的條件機率來計算辨識結果，而單單是引進更多的訓練樣本，對於其訓練器並不會有明顯之改善，且由於 Feature 數較多且各 Feature 之間的關係並不見得為獨立，也因此使得其辨識效率較為低落。

Algorithm	J48		BayesNet		Naïve Bayes	
	Original	Self-learning	Original	Self-learning	Original	Self-learning
Precision	**0.97**	0.96	0.96	0.96	0.92	0.92
TPR	0.69	**0.89**	0.57	**0.61**	0.63	0.63
FPR	**0.02**	0.04	0.02	0.02	0.06	0.06
Accuracy	0.81	**0.92**	0.74	**0.76**	0.76	0.76
F1 score	0.81	**0.93**	0.71	**0.75**	0.75	0.75
Algorithm	Random Forest		Random Tree		RBF Network	
	Original	Self-learning	Original	Self-learning	Original	Self-learning
Precision	**0.96**	0.92	**0.92**	0.85	0.90	**0.91**
TPR	0.64	**0.96**	0.73	**0.99**	0.57	**0.67**

FPR	**0.02**	0.10	**0.07**	0.22	**0.07**	0.08
Accuracy	0.78	**0.93**	0.81	**0.90**	0.72	**0.78**
F1 score	0.77	**0.94**	0.81	**0.91**	0.70	**0.77**

表 17　不同演算法與 Self-Learning 之比較

6.3. 混合式系統(Hybrid System)

本節將針對 Self-Learning 機制的分類器，混合多層級投票的方式後之結果加以討論，並討論影響其結果之因素。

6.3.1 分層

對於分層，這邊主要分為攻擊種類分層與分群分層兩種；對於攻擊種類分層，分別嘗試了分為兩層(DoS+Probe & U2R+R2L)及分為三層(DoS& Probe& U2R+R2L) 之分類器，而演算法方面，則使用前一節所提到之演算法交叉測試，留下實驗結果最好之數據做為比較；而對於分群分層，對於 K-means 分群法之群數設定，則有 2~ 6 群之不同實驗，並且在每種群數之實驗中，皆會對 Random Seed 取不同之五個值，統計後留下最好的辨識結果作為統計比較。

而從圖 29 及圖 39 之圖表結果，可看出再混合多層機制後，無論是利用攻擊種類多層，或是分群多層、對於分類器的表現並沒有明顯的改進，且雖然在模型建立的訓練時間上，有長有短，並沒有過大的差異，但這並沒將做分群法的額外時間加入，因此若以分群多層，可能還會花去更多時間。

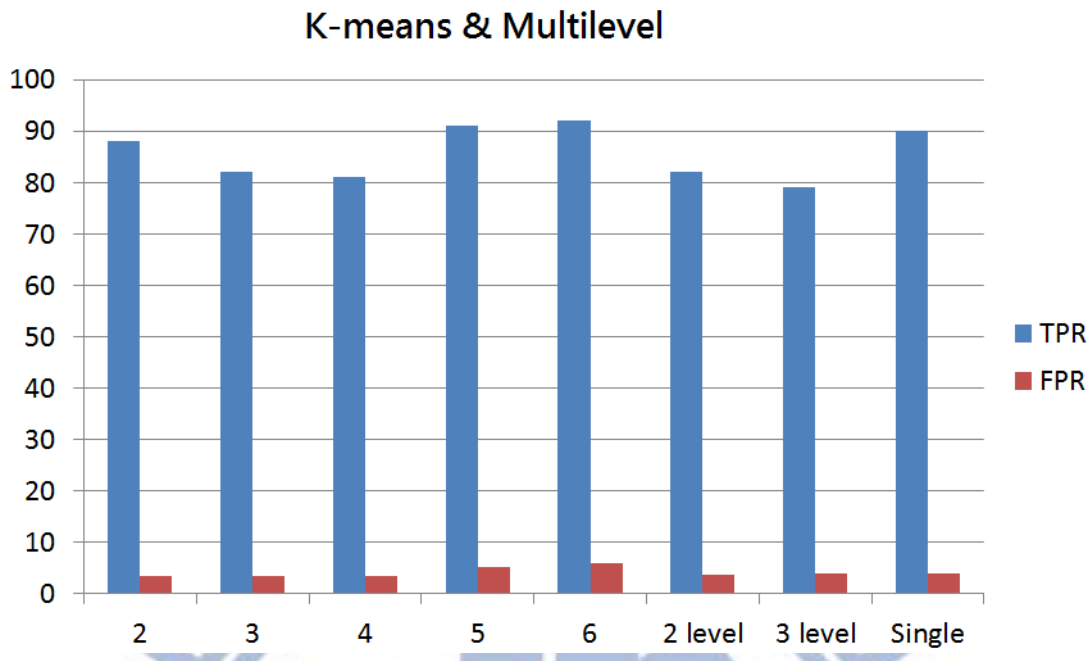


圖 31 混合多層機制下之實驗結果

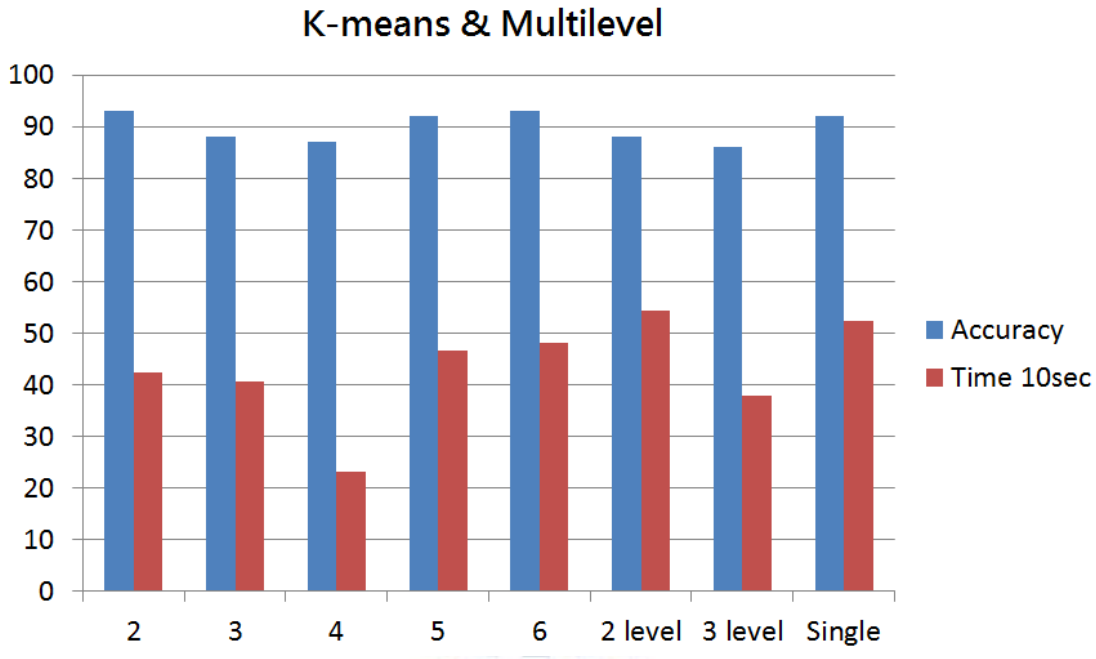


圖 32 混合多層機制下之實驗結果

藉由這些實驗結果也說明了，分層後，反而會使得 Self-learning 所獲得的大量惡意樣本之優點遭到稀釋，使得整體結果的提昇不如預期，而 由圖 31，則可更加驗證此想法，由圖可以看到，從 1~3 層，若未使用 Self-Learning 的方法，其 Accuracy 在伯仲之間，但在採用 Self-Learning 機制後，反而是單一分類器的準確率提升的最多。

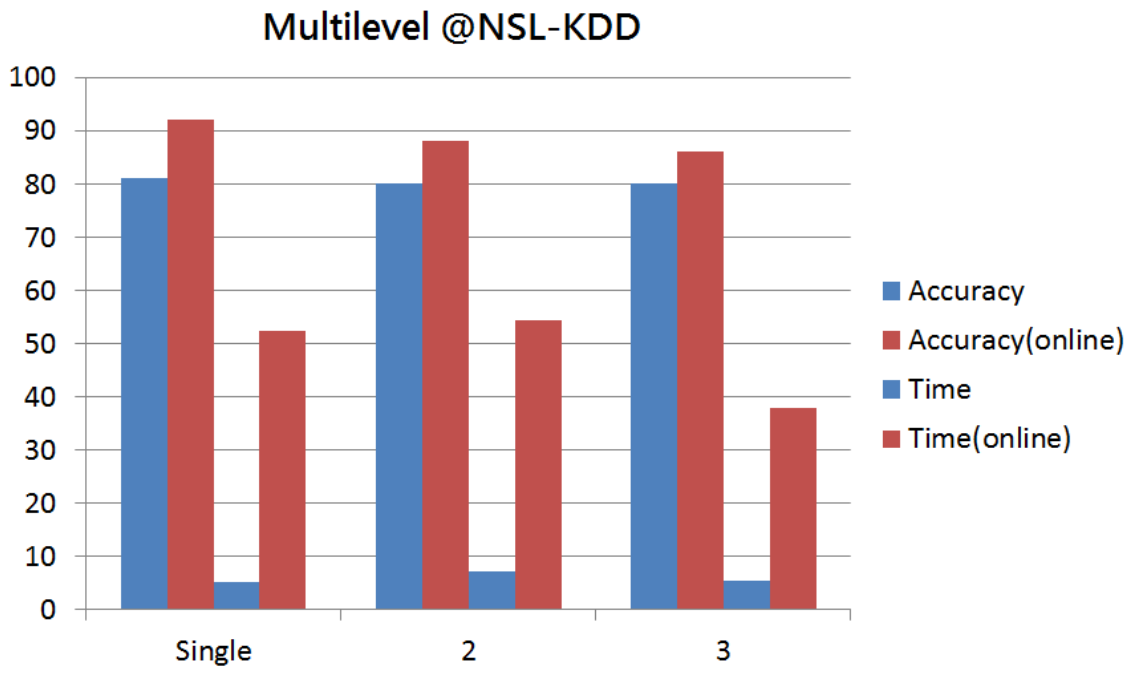


圖 33　混合多層機制下之結果比較

6.3.2 投票

投票機制的用途在於避免 Self-Learning 的過程因分類錯誤導致有過多錯誤樣本，使得最後訓練出之分類器其辨識能力低落的狀況，而在不同的參數設定下，也會導致不同的結果，至於要如何訂定分類器的權重關係以及門檻值，則必須要視情況及目的進行調整，若希望對未知攻擊有高一點的偵測能力，則可以調高 Self-Learning 後期所訓練出之分類器權重，但其同時也必須誤判率較高之風險，相反的，若只是想將 Self-Learning 之後訓練出之分類器作為原始分類器的附加參考依據，則可將後續分類器的權重調低，並升高門檻值，如此辨識結果便會接近原始分類器的狀況，風險較低，但相對的，其偵測未知攻擊的能力便會較差。

而在表 18，則列出在不同資料集、不同演算法下，原始分類器，Self-Learning 分類器，以及經過 Voting 之辨識結果之 Accuracy 作為統計跟比較，這邊的 Voting 數值是配合不同的參數設定所調整出的結果，想表達的是，若有合適的參數設置，藉由 Voting 方法不僅可以避免最壞狀況，甚至可以調整出比原始分類器以及 Self-Learning 分類器都來的好的辨識結果。但要如何在不知測試樣本標籤的情況下，配合所期望的結果，調整出最佳的參數，則是尚待討論的部分。

Accuracy Statistic (Original – Self-learning – Voting)						
	J48	Naïve	Bayes	Random	Random	RBF

		Bayes	Net	Tree	Forest	Network
NSL-KDD	81-92 92	76-76 76	74-76 76	81-90 **93**	78-93 93	72-78 80
Modified NSL-KDD	71-79 79	72-72 72	69-71 71	71-90 **93**	70-93 93	72-75 75
DARPA	78-81 **84**	73-69 **75**	90-89 90	85-28 85	81-89 **90**	80-80 80

表 18　Accuracy 比較

6.4. 隱藏馬可夫模型(Hidden Markov Model)

使用隱藏馬可夫模型所做出的辨識結果，可依據 Feature、Observation 之長度、以及辨識方式有不同的分類，依序將其結果圖表列於下。

首先，在使用正常連線作為模型建立之樣本，也就是相似性模式來進行辨識時，結果如表 19，可發現其辨識能力相當的糟糕，使用 Packet Size 作為 Feature 之 Accuracy 甚至低於 50%，而 IAT 則是幾乎將所有的樣本辨識為正常流量。

Observation Length:10000	Feature			
threshold: -1	Packet Size		IAT	
# of State	TPR-FPR	Accuracy	TPR-FPR	Accuracy
2	0.12- 0.50	0.48	0.39- 0.13	0.85
3	0.02-0.68	0.30	0.39- 0.13	0.85
4	0.86- 0.76	0.24	0.39- 0.13	0.85
5	0.26- 0.73	0.26	0.39- 0.13	0.85

表 19　相似性模式下進行辨識之結果

在 Observation Length 調整為 1000 及 100000、以及調整 Threshold 的值，其辨識結果仍然沒有多大的改變，這邊就不再列出實驗的結果與表格。

而在狀態模式的方法上，這邊都將狀態數設定為 5，而在利用惡意訓練樣本的狀態行為模式對這些狀態進行統計後，惡意樣本所走訪之狀態次數如表 20，而實驗也依序設定惡意狀態數，由 1 到 4 調整，其結果列於表 21。

State Number	Packet Size	IAT

0	82244	934
1	2083	70848
2	327	14
3	1365	14221
4	8	10

表 20　惡意樣本走訪狀態次數之統計

Observation Length:10000	Feature			
	Packet Size		IAT	
# of Malicious State	TPR- FPR	Accuracy	TPR- FPR	Accuracy
1	0.01-0.09	0.73	0.01- 0.08	0.74
2	0.01-0.06	0.75	0.01- 0.01	0.79
3	0.01-0.06	0.76	0.01- 0.01	0.79
4	0.01-0.05	0.76	0.01- 0.01	0.79

表 21　狀態模式之隱藏馬可夫模型辨識結果

參考表 15，可發現即使是單純的 Decision Tree 演算法，其辨識效果都比利用此兩種 Feature 所建立之隱藏馬可夫模型之結果要來的好上許多，也因此，原先想利用隱藏馬可夫模型作為 Self-Learning 分類器之二次過濾工具的構想也就沒有辦法完成。因以這兩種 Feature 所建立之隱藏馬可夫模型，對惡意攻擊近乎沒有偵測能力，也因此，由該馬可夫模型可辨識出之攻擊樣本，先前利用其他演算法之分類器也都有能力找出，而若依照原先構想，將被分類器辨識為惡意樣本之資料，交由隱藏馬可夫模型進行辨識，則大多也僅會被辨識為正常的連線樣本，並無助於進行二次過濾之用，對此結果，應是該馬可夫模型過於簡單，所以無法完整的建構出惡意行為的馬可夫鏈狀態，否則依照[9] 當中所提的惡意連線之特性，封包大小應可作為判斷的 Feature，而要如何定義更深入的 Feature 建立馬可夫模型支用，則還有研究的空間；又或著，也可試著以離散獨立的狀態來看待樣本中的每個封包大小，將其視為獨立的大量 Feature，或是計算一個 Netflow 中重複出現的 Packet 大小，皆是未來可行的研究方向。

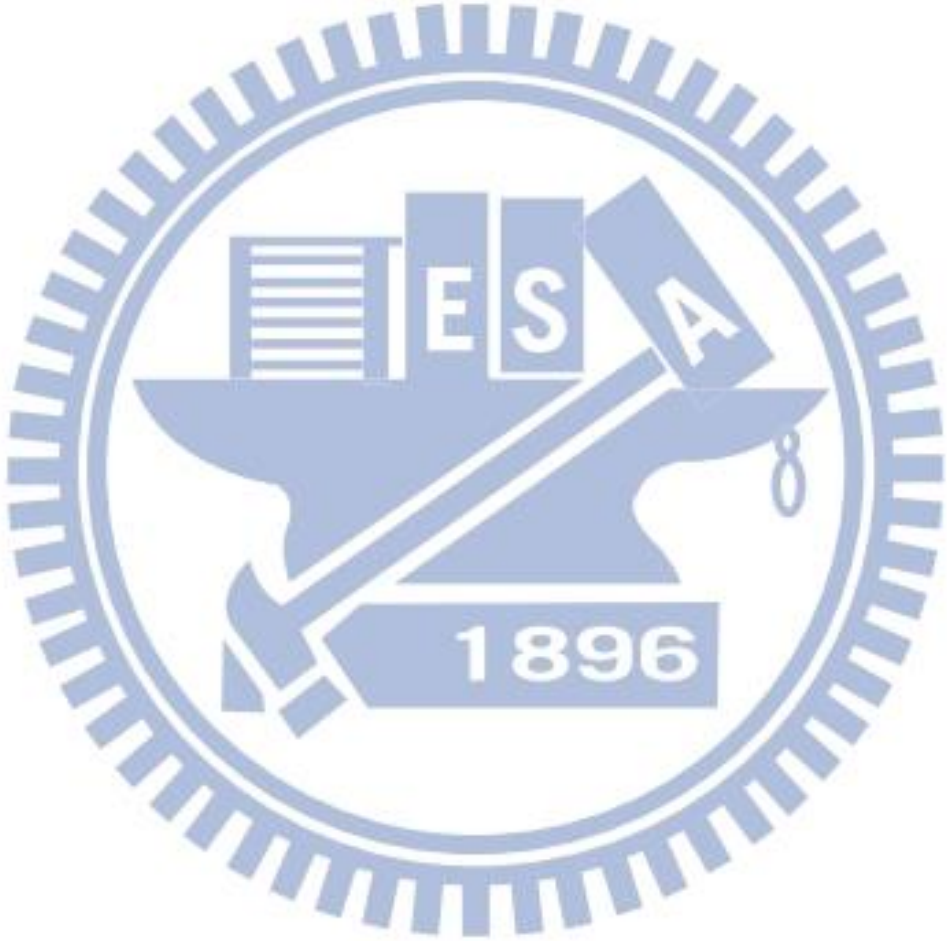
七、 結論(Conclusion)

本論文從自我學習方式的辨識運用，到混合了多層機制以及分群法、加入投票方法來防止最壞情況的產生，再到隱藏馬可夫鏈的運作，嘗試利用各種的方式來針對未知攻擊做到有效的偵測與捕捉，並配合不同的資料集來做不同的實驗結果評估，包含已經提供 Feature 的 NSL-KDD、經過更動後的 Modified NSL-KDD，以及自行從原始 Raw Data 之檔案所轉換的自定義特徵，其中並包含有針對個別樣本之連續性特徵資訊。

其中，自我學習機制的成果最為顯著，對於偵測未知攻擊(Unknown Attack)在各資料集皆具有一定的辨識效率及整體準確率提升，除了對攻擊的偵測率上升不少，其誤判率上升的比率並不大，然而，在混合了多層的機制之後，並未如原先所預期的讓實驗結果更進一步的提升，原因應在於分層反而使得自我學習機制，使惡意樣本的數目及樣本類型增加的優點給抵消掉了，使得每層之訓練集當中的惡意樣本減少，雖然分層理應可以對各類攻擊更針對性地捕捉，但其反而使得一些特徵介於各種攻擊的未知惡意攻擊難以捕捉，反而在不分層的情況下，對於此類特徵較為模糊的未知攻擊，才較有機會辨識成功，但其同時也帶著誤判率提高，使得 False Positive 上升的風險，也因此，為了避免誤判率高，導致自我學習機制所訓練出之分類器之辨識結果過於差勁，投票機制此時就有著不可或缺的重要性，藉由投票機制的參數調整，可決定初始分類器及經過自我學習機制所訓練出的多個分類器其權重差異，並配合辨識門檻的訂定，將可有更彈性化的系統設置。

隱藏馬可夫模型的應用，原先規劃作為自我學習機制分類器的過濾之用，其出發點來自於惡意連線可能皆有著特定大小的封包大小等特性，因此便將 Netflow 中的個別封包大小用來建立隱藏馬可夫模型之用，在經過將連續性 Feature 轉換為 Observation 過後，配合兩種不同辨識方式的使用，希望能夠將自我學習機制當中被判定為惡意攻擊之樣本二次過濾，降低系統的 False Positive，但從封包大小、IAT 等特徵所建立之隱藏馬可夫模型，從實驗結果上來看，無論 Observation 所取的長度，或是惡意狀態的設立，門檻值的訂定，這些參數的調整，其辨識結果仍然離理想值有一定的差距，只要用單純的機器學習演算法如 Decision Tree 即可有更好的辨識結果，代表單單藉由這些由封包的 Feature 作為 Observation 所建立之隱藏馬可夫模型，並無法準確的預測出惡意攻擊的行為模式，若要有更好的辨識結果，應要定義更多的 Feature 來做為隱藏馬可夫模型之 Observation，並更深入的處理 State 之定義，並重新思考該如何判斷樣本為惡意攻擊，單單依照正常連線所建立的模型或是以簡單的狀態定義及門檻值來作為判斷依據，明顯是不足的，必須要在這方面再加以改進。

另外在未來的方向上，除了上述所提到的機器學習方法改進、以及不同的 Feature 定義外，在資料集上，隨著現在攻擊的日新月異，各種不同類型的惡意滲透方式推陳出新，應可配合各種開源的滲透測試工具以及 Honeypot 的設立，側錄更多的惡意連線及攻擊樣本，來拓展測試資料集的多樣性，並更進一步提升評測結果的價值，來取代本實驗所用的資料集。



八、 參考文獻(Reference)

[1] Large unexplained DDoS attack hits USA. http://www.transients.info/2014/05/large-unexplained-ddos-attack-hits-usa.html

[2] S.Seufert and D. O'Brien, "Machine Learning for Automatic Defence against Distributed Denial of Service Attacks" IEEE International Conference on Communications, Glasgow, pp. 1217-1222, 2007.

[3] G. Münz, S. Li, and G. Carle, "Traffic Anomaly Detection Using K-Means Clustering"，2007.

[4] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, " Can machine learning be secure?" Proceedings of the 2006 ACM Symposium on Information, computer and communications security(ASIA CCS), pp. 16-25, Taipei, Taiwan, 2006.

[5] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools" IEEE Communications Surveys & Tutorials, Vol. 16, No. 1, pp. 303-336, 2014.

[6] P. Natesan1, and P. Balasubramanie, "Multi Stage Filter Using Enhanced Adaboost for Network Intrusion Detection" International Journal of Network Security & Its Applications (IJNSA), Vol.4, No.3, pp. 121-135, 2012.

[7] P. Jongsuebsuk, N. Wattanapongsakorn, and C. Charnsripinyo, "Network intrusion detection with Fuzzy Genetic Algorithm for unknown attacks" Internet Conference on Information Networking(ICOIN), Bangkok, 2013.

[8] A. AlEroud and G. Karabatis, "Toward Zero-Day Attack Identification Using Linear Data Transformation Techniques" IEEE 7th International Conference on Software Security and Reliability (SERE), Gaithersburg, MD, 2013.

[9] P. M. Comar, L. Liu, S. Saha, P. N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection" IEEE INFOCOM, pp. 2022-2030, Turin, 2013.

[10] KDD 99 Dataset.
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[11] DARPA Dataset.
http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, ”A detailed analysis of the KDD CUP 99 data set” IEEE Symposium on Computational Intelligence for Security and Defense Applications(CISDA), Ottawa, ON, 2009.

[13] A. Blum and T. Mitchell, ”Combining Labeled and Unlabeled Data with Co-Training” COLT' 98 Proceedings of the eleventh annual conference on Computational learning theory，pp. 92-100, 1998.

[14] V. Ng and C. Cardie, ”Bootstrapping Coreference Classifiers with Multiple Machine Learning Algorithms” Coreference on Empirical Methods in Natural Language Processing (EMNLP), 2003.

[15] S. He and D. Gildea, ”Self-training and Co-training for Semantic Role Labeling: Primary Report”.

[16] J. Zhang, C. Chen, Y. Xiang, and W. Zhou, “Robust network traffic identification with unknown applications” Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security(ASIA CCS), pp. 405-414, Hangzhou, China，2013.

[17] N. B. Amor, S. Benferhat, and Z. Elouedi, ”Naive Bayes vs Decision Trees in Intrusion Detection Systems” ACM symposium on Applied computing, pp. 420-424, 2004.

[18] M. Stamp, ”A Revealing Introduction to Hidden Markov Models”, 2012.

[19] The Baum-Welch algorithm for hidden Markov Models: speed comparison between octave / python / R / scilab / matlab / C / C++,
http://www.math.univ-toulouse.fr/~agarivie/Telecom/code/index.php

[20] Normal Distribution Picture,
http://upload.wikimedia.org/wikipedia/commons/thumb/8/8c/Standard_deviation_diag

ram.svg/525px-Standard_deviation_diagram.svg.png

[21] FlowScan Architecture, http://www.caida.org/tools/utilities/flowscan/arch.xml#refs

[22] Softflowd, http://www.mindrot.org/projects/softflowd/

[23] P. Gogoi1, M. H. Bhuyan1, D.K. Bhattacharyya1, and J.K. Kalita, ”Packet and Flow Based Network Intrusion Dataset” Communications in Computer and Information Science Volume 306, pp. 322-334, 2012.

[24] Weka, http://www.cs.waikato.ac.nz/ml/weka/

[25] Orange, http://orange.biolab.si/

[26] Attribute-Relation File Format (ARFF), http://www.cs.waikato.ac.nz/ml/weka/arff.html

[27] Weka API Documents, http://weka.sourceforge.net/doc.stable/

[28] ROC Curve Picture, http://upload.wikimedia.org/wikipedia/commons/thumb/3/36/ROC_space-2.png/1024px-ROC_space-2.png

[29] Precision and Recall, http://en.wikipedia.org/wiki/Precision_and_recall

[30] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection" Information Sciences: an International Journal, vol. 177, pp. 3799-3821, 2007.

[31] D. Bolzoni, S. Etalle, and P. Hartel, "Poseidon: a 2-tier anomaly-based network intrusion detection system" Information Assurance, 2006. IWIA 2006. Fourth IEEE International Workshop on, London, 2006.

[32] G. Liu and Z. Yi, "Intrusion Detection Using PCASOM Neural Networks" Advances in Neural Networks - ISNN 2006. vol. 3973, 2006.

[33] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative

study of anomaly detection schemes in network intrusion detection" Proceedings of the third SIAM international conference on data mining, 2003.

[34] D. Barbara, N. Wu, and S. Jajodia, "Detecting novel network intrusions using bayes estimators" First SIAM Conference on Data Mining, 2001.

[35] C. Chen, Y. Gong, and Y. Tian, "Semi-supervised learning methods for network intrusion detection" Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on, pp.2603-2608, Singapore, 2008.