IMPROVED DETECTION OF DOS ATTACKS USING INTELLIGENT COMPUTATION TECHNIQUES

J.Visumathi, Dr. K. L. Shunmuganathan

ABSTRACT

IDSs play a principal role in pro-actively detecting intrusions into enterprise-level computer networks, therefore the accuracy with which it performs this vital function is of paramount importance. Many studies have previously been conducted to improve upon proper classification of detections using neural networks and machine learning algorithms. We try to compare the performance of various intelligent computation techniques like Bayesian networks, Naïve Bayesian, Logistic regression, RBF networks, Multi-Layer perception, SVMs with the SMO model, Kth nearest neighbour and Random forest in detecting DoS attack patterns. The data that was used to train and validate these techniques was obtained from the MIT Lincoln lab study into IDSs. The results obtained provide a clear comparison of the individual intelligent computation techniques ability in identifying and classifying attack patterns.

Keywords: Networks, intrusion detection, denial of service, datasets, data mining, Bayesian networks, Naïve Bayesian, Logistic regression, RBF networks, Multi-layer perception, Support vector machines, Sequential minimal optimization, Kth nearest neighbor, Random forest.

1. INTRODUCTION

Over the last decade, numerous processes and functions in various domains have become increasingly dependent on computers fuelled by the growth of the internet. The internet which is built on an open architecture is a dynamic platform that is readily accessible to the public making it truly global in nature. This rapid acceleration of the internet coupled with the increase in the number of enterprises that rely on computer networks to carry out vital functions highlights the paramount need to ensure round-the-clock availability of such services.

Denial-of-Service (DoS) attacks are performed with the intention of disrupting or preventing the availability of a particular service to a user or group of users and vice-versa. DoS attacks and particularly Distributed DoS (DDoS) attacks have gained greater prominence at the turn of this century with the differentiating factor between the two being; DoS attacks are propagated from a single machine or set of machines within the same geo-physical boundary, whereas DDoS attacks are propagated from a multitude of sources scattered among different locations all acting in concert to perform an orchestrated attack. Correspondingly, IDSs have been at the fore-front of defending critical network installations from such attacks and are widely considered to be the keystone for

a well integrated network security setup.

Prior studies in this field have tried to improve the detection ability of IDSs by addressing their pattern classification accuracies, lowering false-alarm rate thresholds and their efficiency through the use of load balancing or distributed parallel computing techniques [1, 2]. A study has also been conducted into improving the response curve of IDSs by moving the primary detection point to the router at the network perimeter [3]. The primary objective of this study is to perform a comparative analysis on a selection of intelligent computation techniques with the aim of achieving improved pattern classification accuracy levels in recognizing various DoS attack patterns using datasets tailored for this experimental process. The algorithms we considered for our study were chosen for their particular ability in handling multi-variate empirical data with highly complex feature sets and their prediction accuracies as previously documented and are listed as - Naive Bayesian, Bayesian network, Logistical regression, SVMs, RBF network, Multilayer perception, SMO, KNN and Random Forest. The results of our experiments and the conclusions that we have drawn from it will provide a clear indication of the performance and abilities of the various intelligent computation techniques chosen for this study.

The contents of this paper have been arranged into five distinct sections to coherently delineate the concepts being stated here. Section 2 details the data-sets that were used for this study. Section 3 provides a concise overview of the data mining process and the intelligent computation techniques that were utilized. Section 4 describes the methodology that was followed for performing analysis and comparison of the datasets. It also provides an elaborate understanding of the experimental setup and any procedures that were used to obtain the results after the experimental phase of the study. Section 5 presents any conclusions that can be inferred from the results of the experimental study along with its current and future implications followed by a list of sources that were used as a reference for this study.

2. DATA-SETS

The data used to perform an evaluation of the various techniques being profiled by this paper consists of four subsets that were derived from a data-set which was originally conceived as part of a 1998-99 DARPA sponsored study at MIT Lincoln Labs to evaluate and classify IDSs relative to their levels of detection accuracy and false alarm rates [4]. The dataset was crafted to provide an emulation of training and testing scenarios allowing the participants of the evaluation to be evaluated against an off-line simulation environment modeled on a real-world production environment. Moreover it allowed creation of an automated control set that can be replicated and repeated without any variation

as often as required during the experimental process.

The four datasets that were prepared to determine the classification accuracy rates of the intelligent computation techniques are – Neptune (11982 data points), Pod (10733 data points), Murf (11982 data points), and Teardrop (11982 data points). Each data point in any of the four datasets represents a TCP/IP connection having specific features that provide various attributes with regard to the connection. There are overall 41 features for each data point in all four datasets and these features can be used to broadly categorize the connections under three formats – intrinsic, content based or traffic based. The data points are classed as positive or negative – where positive points refer to connections where an attack has been propagated and negative points refer to normal data connections. The intelligent computation techniques under evaluation have to accurately determine the number of positive connections among the overall number of connections for each dataset which is then calculated as a percentile, thereby providing a basis for numerical comparison.

Four particular attack scripts were used to individually propagate attacks in each of the four subsets respectively to test the potential effectiveness of the different algorithms at classifying data by recognizing suitable patterns [5]. A short description of the four attack techniques and their respective datasets are provided below:

1.1 Dataset 1 – Neptune

The first dataset Neptune contains 11982 different data points containing 1980 positive points and 10002 negative points with 41 features each. The syn-flood attack that was propagated against the experimental setup in this dataset exploited the unbounded listen state feature for half-open connections in the 3-way TCP handshake protocol, where the client after receiving a SYN-ACK packet from the server after the transmitting the initial SYN packet does not respond back with the ACK packet to complete the connection. This causes the server to allocate resources till the time specified as such awaiting completion of the connection which is not going to happen.

For this particular experiment, the Neptune program was used to send about twenty SYN packets to the server machine on every port from 1 to 1024 every ten minutes causing a blackout of all essential services provided by the victim server. Though this attack's consequences are temporary, the attacker by repeatedly performing it can inundate the victim's machine from achieving future recovery of services by establishing legitimate connections. Depending on the severity of the attack and the vulnerability of the victim, it may cause a system crash or memory loss due to buffer overflow. Some of the defense mechanisms known to prevent such an attack are – ingress filtering to remove spoofed addresses, reducing the SYN-Received timer, increasing the backlog size,

etc.

1.2 Dataset 2 – Pod

The second dataset Pod contains 10733 different data points containing 139 positive points and 10594 negative points with 41 features each. The Ping of death used to propagate attacks in this dataset works by exploiting a design flaw in IP fragment re-assembly process of various OS kernels. The maximum size of an IP packet is 65535 bytes with about 20 bytes allocated to a bare IP header and 8 bytes for the ICMP echo request with the remaining bytes available for data allocation. As the DLL (Layer 2) restricts the maximum size of any transmitted packet to 1500 bytes, large IP packets have to broken down in to smaller fragments to accommodate the MTU size. On receiving all fragments, the receiver then re-assembles them to derive the entire IP packet with each packet being identified by a fragment offset value to determine its position in the sequence of fragments. This can be manipulated to create a malformed packet which is larger than the standard accepted 65535 bytes, thereby causing a buffer overflow, memory loss, system crash, etc [6]. This attack can be avoided by ensuring the packet re-assembly process only accepts IP fragments with offset value and total length equaling the standard size.

1.3 Dataset 3 – Murf

The third dataset Murf contains 11982 different data points containing 1980 positive points and 10002 negative points with 41 features each. The attack that was propagated in this dataset is known as the Smurf attack. It works on the principle of using forged ICMP echo requests sent to an intermediary network broadcast address with the intended victim's network IP address as the source. If the intermediary network accepts ICMP echo requests to the broadcast address then every host that is up on the intermediary network responds to the echo request, thereby sending a flood of responses to the victim's address causing network congestion [7]. The signature pattern of such an attack would have a large no of ICMP responses to the victim's IP but no requests as such from the same IP. This attack can be avoided by preventing a routing device to accept echo requests to broadcast addresses and disabling OSs to respond to ICMP echo requests sent to a broadcast address. Ingress filtering can also be used to prevent networks from becoming part of a Smurf attack.

1.4 Dataset 4 – Teardrop

The fourth dataset Teardrop contains 11982 different data points containing 11003 positive points and 979 negative points with 41 features each. The teardrop attack which is a UDP attack is somewhat similar to Ping of death attacks where it uses malformed IP packets to exploit the fragment offset overlap feature in the IP datagram stack. But if a receiver receives a large number of fragments with an offset value not matching any that were sent before and the

receiver is unable to handle it properly, it leads to a system crash causing a loss of data [8].

3. INTELLIGENT COMPUTATION TECHNIQUES - MODELS & FEATURES

3.1 Data Mining – A Process in Evolution

Data mining is an analytical process that explores large segments of complex or seemingly random data to detect any underlying patterns or systematic relationships between independent or dependent variables. The process consists of two main stages which involves exploration of the dataset to final application of the predictive model to validate the patterns [9, 10].

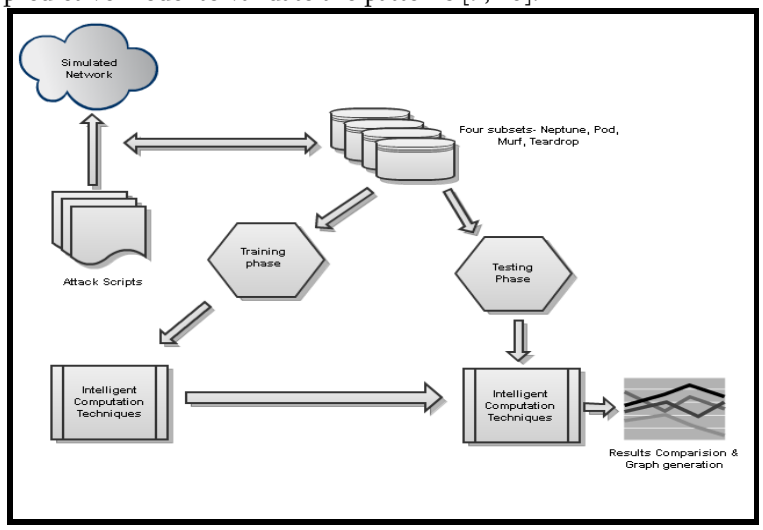

Figure 1: Process Flow Diagram for Experimental Process

Stage I

This initial phase of the process prepares the data for analysis and validation in the next stage. The primary functions performed during this stage are cleansing of data, data transformations, feature selection or selection of subsets. Data cleansing involves removing any junk values such as impossible values or those that are out of range. Data transformation and feature selection breaks down large complex data-sets into more manageable chunks that simplify the number of variables by selecting only those variables that are deemed to be most beneficial in pattern identification.

Stage II

During this stage, the selection of models for analysis of the data to discover the underlying pattern for prediction is undertaken via the process of comparative selection by application of the chosen models on a standard dataset and

obtaining the resultant performance. The patterns can be validated by applying the selected process on similar or new datasets and achieving similar results within an acceptable error margin.

3.2 Bayesian Network

The Bayesian Network can be used to compute joint distribution problems through computation of any number of truth values to variables in linear with the number of nodes being selected. A Bayesian network for a set of variables X_1 to X_n represented by X_i, where $i = 1$ to n

$$P(X_1 = x_1 \wedge X_2 = x_2 \wedge \dots \dots \dots X_{n-1} = x_{n-1} \wedge X_n = x_n) =$$
$$P(X_n = x_n \wedge X_{n-1} = x_{n-1} \wedge \dots \dots \dots X_2 = \wedge X_1 = x_1) =$$
$$= \prod_{i=1}^{n} P(X_i = x_i | (X_{i-1} = x_{i-1}) \wedge \dots (X_1 = x_1))$$
$$= \prod_{i=1}^{n} P(X_i = x_i | Assignment\ of\ Parents\ (X_i))$$

The major conditional requirement for a Bayesian network to model a probability distribution function is that each variable is conditionally independent of all its non-parent descendents in the graph given the value of all its parents [11]. Thus the Bayesian network algorithm can be generalized as follows:

$$P\left(\frac{E_1}{E_2}\right) = \frac{P(E_1 \wedge E_2)}{P(E_2)} = \frac{\sum P_{joint\ entries\ matching\ E_1 \& E_2}}{\sum P_{joint\ entries\ matching\ E_2}} \qquad (1)$$

3.3 Naïve Bayesian

Naïve bayes works on the assumption that all variables of a given class are independent of the values of other variables. This principle is known as class condiotional independence. The equation below will try to provide the probability distribution of the possible values Y for every instance of X.

$$Y^{predict} = \underset{v}{\operatorname{argmax}}\, P(Y = v | X_1 = u_1 \dots X_n = u_n) \qquad (2)$$

Using Bayes theorem;

$$Y^{predict} = Y^{predict} = \underset{v}{\operatorname{arg\,max}}\, P(X_1 = u_1 \dots X_n = u_n | Y = v) P(Y = v) \qquad (3)$$

Equation (3) is possible because the posterior possibility that X belongs to Y as the conditional probabilities of the independent variables are statistically independent. It can further be resolved into

$$Y^{predict} = \underset{v}{\operatorname{argmax}}\, P(Y = v) \prod_{j=1}^{n_y} P(X_j = u_j | Y = v) \qquad (4)$$

Where denominator does not depend on v

3.4 Logistic Regression

The logistic regression model can be used to explain the relation between multiple independent variables and a binary response as generalized below [12]:

$$\log\{\Pr(Y=1|x)\} = \log\left\{\frac{\Pr(Y=1|x)}{1-\Pr(Y=1|x)}\right\} = \beta_0 + x'\beta \qquad (5)$$

Where β_0 is the intercept parameter and β is the vector of the slopes
If the probability of Y changes from 0 to1 its co-efficient is negative. In that case, the probability of response to 1 can be represented as

$$\log\left\{\frac{\Pr(Y=1|x)}{1-\Pr(Y=1|x)}\right\} = -\beta_0 - x'\beta \qquad (6)$$

As such it can be inferred that the regression coefficients for modeling the probability of 1 will have the same magnitude but opposite polarity to the model of 0.

3.5 RBF Networks

RBF networks take a non-linear input and produce a linear output with efficient approximation levels. They find implementation as a two layer feed-forward network with a processing layer between the input and out units called the hidden layer. When applied for pattern classification the inputs correspond to feature entries and the outputs as are labeled as classes with hidden units assigned as sub-classes [13]. The most preferred activation function for RBF networks is the Gaussian function represented as

$$\phi_j(X) = \exp\left[-(-\mu_j)^T \Sigma_j^{-1}(-\mu_j)\right] \qquad (7)$$

For $j = 1,\ldots, L$
Where X is the input feature vector and L is the number of hidden units. μ_j and Σ_j are the mean and covariance matrix of the *jth* Gaussian function respectively.
For pattern classification, the output of the radial basis function is limited to an interval (0, 1) using the sigmoid function.

$$Y_k(X) = \frac{1}{1+\exp\left[-\Psi_k(X)\right]} \qquad (8)$$

For $k = 1,\ldots, M$

3.6 Multi-Layer Perception (MLP) Algorithm

Multi-layer perception algorithm was developed with the singular goal of overcoming the weakness of perception algorithm, i.e., the in-ability to solve problems that are linearly inseparable, as was illustrated through the example of logic functions specifically the XOR function. The best way of implementing this algorithm would be to start by randomizing all weights between the numbers -1 and +1. The MLP algorithm can be summarized as below -

$$x^0 \xrightarrow{W^1 g^1} x^1 \xrightarrow{W^2 g^2} \ldots \xrightarrow{W^L g^L} x^L$$

Where $x^l \in R^{n_l}$ for all $l = 0,\ldots,L$ and W^l is n_l x n_{l-1} matrix for all $l = 1,\ldots,L$
There are $L + 1$ layers of nodes and L layers of synaptic weights. The major steps of MLP algorithm can be considered as

1. Forward pass – The first step of the process is to transform the input vector x^0 into the required output vector x^L. This can be achieved using the following equation:

$$x_i^l = f(u_i^l) = f\left(\sum_{j=1}^{n_{l-1}} W_{ij}^l x_j^{l-1} + b_i^l\right) \qquad (9)$$

For *l = 0 to L.*

2. Error computation – The difference between the desired output d and the actual output x^L can be calculated using the equation:

$$\delta_i^L = f'(u_i^L)(d_i - x_i^L) \qquad (10)$$

3. Backward propagation – After computation of the error, it is applied to the output unit and then propagated backwards using the evaluation of

$$\delta_i^{l-1} = f'(u_i^{l-1}) \sum_{j=1}^{n_l} \delta_j^l W_{ij}^l \qquad (11)$$

Here *l = L to 1.*

Note that the forward and backward passes use the same weights but with the directions reversed. MLP uses two modes for network training – sequential mode and batch mode.

3.7 SVMs

In any predictive learning task, such as classification, both a model and a parameter estimation method should be selected in order to achieve a high level of performance of the learning machine. Then the task of learning amounts to selecting the sought-after model of optimal complexity and estimating parameters from training data [4]. Within the SVMs approach, usually parameters to be chosen are;

a) the penalty term *C* which determines the trade-off between the complexity of the decision function and the number of training examples misclassified;

b) the mapping function

c) the kernel function such that .

3.8 Sequential Minimal Optimization on SVM

Sequential Minimal Optimization algorithm is an optimization algorithm for training Support Vector Machines (SVM). The algorithm gives optimal solution for large quadratic programming (QP) problems, by breaking it into smallest possible quadratic programming (QP) problem and find solutions to these smaller problems [14]. For a training set of n samples the following condition must be satisfied

Minimize $\frac{1}{2}|w^{-1}|$ subject to $c_i(w.x_i-b)\geq 1$ for all $1\leq i\leq n$

where w is normal vector perpendicular to hyper plane which divides two classes, c_i is class of i^{th} data point and x_i is the feature vector of i^{th} data point.

3.9 Kth Nearest Neighbor

K Nearest Neighbor algorithm is an instance based learning algorithm. Classification of a data point is done based on majority voting of its K (K>=1) Nearest Neighbors. During training phase the feature vectors and classes of training samples are stored. In testing phase when a new data point is given, distance between this data point and all the data points in training data points is calculated to find K^{th} nearest data points. The distance metrics can be Euclidean distance, Manhattan distance, Hamming distance etc [15]. New data point will be assigned to most frequent class in these K Nearest data points. We used k=9 in our experiments.

3.10 Random Forests

A random forest is a classifier consisting of a collection of tree structured classifiers $\{h(x,\Theta_k), k=1, \ldots\}$ where $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class of input X [16] .

The common element in random trees is that for the k^{th} tree, a random vector Θ_k is generated, independent of the past random vectors $\Theta_1,\ldots\ \Theta_{k-1}$ but with the same distribution; and a tree is grown using the training set and Θ_k, resulting in a classifier $h(x,\Theta_k)$ where x is an input vector.

4. EXPERIMENTAL METHODOLOGY & RESULTS

The experimental procedure is built on a simple premise or understanding, where data selected to evaluate the different algorithms is used to determine and train intelligent computation models to the effect that when the models are presented with new data, they will be able detect patterns and make a classification based on some preset parameters. The use of probability models to increase the accuracy of these classifications is the main feature of this particular study.

Of the 11982 data points in three datasets, 5092 data points were used for training the models and 6890 data points were used for validating the models after the training phase with all points for both phases being randomly selected. A selection of four attack scripts described in section 3 was used to propagate attacks against the test network. The data points are divided into two classes “positive” and “negative” based on the probability of an attack taking place. Where positive refers to the cases where an attack has taken place and negative refers to normal data. The testing data when applied to the individual models for validating the data points against the two classes will provide an accuracy level which can be compared across the board to determine the best among models.

To provide a better understanding of the results, it is partially represented graphically in the form of ROC curves. The ROC curve plots consist of two axis –

x-axis (negative detection) & y-axis (positive detection). Therefore based on the results of the experiment, ROC curves describing the performance of SVMs in the four data sets are shown in Figures (2 to 5). In perfect circumstances, a ROC curve will immediately rise to the vector point (0, 1) indicating complete detection with no false positives and then trail off to (1, 1). Since most of the other algorithms show near perfect detection tendencies they are not represented here.

Intelligent Computation Technique / Algorithm	Classifier Accuracy Rates (%)			
	Neptune	**Pod**	**Murf**	**Teardrop**
Naïve Bayesian	99.97	95.6	99.87	99.95
Bayesian Network	100	100	100	99.97
Logistic regression	99.95	99.97	99.97	99.97
SVM	98.7	100	99.55	99.87
RBF Network	99.97	99.03	99	99.97
Multilayer Perception	99.77	100	100	99.77

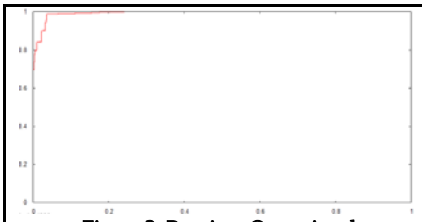

Figure 2: Receiver Operational Characteristic (ROC) curve of Neptune dataset using SVM's

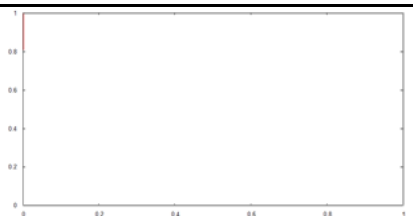

Figure 3: Receiver Operational Characteristic (ROC) curve of Pod dataset using SVM's

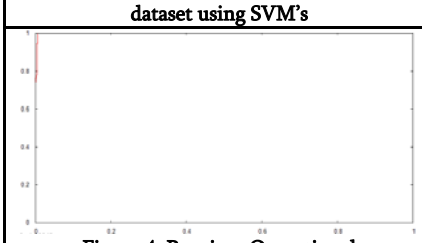

Figure 4: Receiver Operational Characteristic (ROC) curve of Murf dataset using SVM's

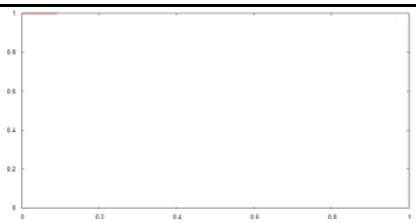

Figure 5: Receiver Operational Characteristic (ROC) curve of Teardrop dataset using SVM's

SMO	100	99.86	100	100
KNN	100	99.97	100	99.97
Random Forest	100	100	100	100

5. CONCLUSIONS

On successful completion of the experimentation process during which we evaluated nine different classifiers against four large and complex data sets, our observations and conclusions can be summed up as given below:

Random forest successfully achieved 100% accuracy across all four data sets, while SMO and Bayesian network follow it with 100% detection accuracy in three different datasets – Datasets 1/3/4 and Datasets 1/2/4 respectively. They both score above 99% in the remaining datasets respectively.

Multilayer perception and KNN bring up the middle order with 100% detection accuracy in two datasets (Datasets 2/3 and datasets 1/3 respectively) and above 99% detection in the remaining two data sets.

The bottom of the pack has a few mixed results with RBF and Logistic regression just making it above the 99% mark in all four datasets. SVM was able to score 100% detection accuracy in dataset 2 and above 99% in datasets 3 and 4. On the other hand Naïve Bayesian was able to score above 99% in datasets 2, 3 and 4. But surprisingly both SVM and Naïve Bayesian reported scores below 99% accuracy rates in single datasets, i.e, 98.7% for the first dataset and 95.6% for the second dataset respectively.

REFERENCES:

1. Kruegal C., Valeur F., Vigna G., Kemmerer R., "Statefull intrusion detection for high speed networks", In proceedings of IEEE Symposium on Security and Privacy, pp 285-294, May 2002
2. Mukkamala S., and Sung. A. H. (2003) A Comparative Study of Techniques for Intrusion Detection. Proceedings of 15th IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society Press, pp 570-579
3. K. Park, and H. Lee, "On the Effectiveness of Router-Based Packet Filtering for Distributed DoS attack and Prevention in Power-Law Internets", Proc. of the SGICOMM, pp. 15-26, 2001
4. S. E. Webster, "The Development and Analysis of Intrusion Detection Algorithms", *S.M. Thesis, Massachusetts Institute of Technology*, 1998
5. K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", *Master's Thesis, Massachusetts Institute of Technology*, 1998.
6. "Internet Protocol Specification", IETF, RFC 791, September 1981
7. CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks http://www.cert.org/advisories/CA-1998-01.html, January 5, 1998
8. Jason Anderson, "An Analysis of Fragmentation Attacks", March 2001
9. "Statistics: Methods and Applications", Statsoft Publications
10. Vladimir V. N. (1995) The Nature of Statistical Learning Theory. Springer
11. Tommi Jaakkola, "Machine Learning: Bayesian networks, Support Vector Machines & Model selection", MIT, 2006
12. Jia Li, "Logistical Regression", Department of Statistics, University of Pennsylvania, 2000~
13. Ying So, "A Tutorial on Logisitc Regression", SAS Institute, 2001
14. [John Platt, "Fast training of support vector machines using sequential minimal optimization," Advances in kernel methods: support vector learning, Pages: 185 – 208, 1999

15. [Harp P.E., "Nearest neighbour pattern classification". IEEE Transactions on Information Theory 13 (1): 21-27 (1967)
16. Leo Breiman, "Random Forests", Machine Learning, pp5-32, Kluwer Academic Publishing, 2001.