Feature Selection for Intrusion Detection using Neural Networks and Support Vector Machines

Srinivas Mukkamala[1] & Andrew H. Sung[1,2]
[1]Department of Computer Science
[2]Institute for Complex Additive Systems Analysis
New Mexico Tech
Socorro, New Mexico 87801
srinivas|sung@cs.nmt.edu

ABSTRACT

Computational Intelligence (CI) methods are increasingly being used for problem solving. This paper concerns using CI-type learning machines for intrusion detection, which is a problem of general interest to transportation infrastructure protection since a necessary task thereof is to protect the computers responsible for the infrastructure's operational control, and an effective *Intrusion Detection System* (IDS) is essential for ensuring network security.

Two classes of learning machines for IDSs are studied: *Artificial Neural Networks* (ANNs) and *Support Vector Machines* (SVMs). We show that SVMs are superior to ANNs in three critical respects of IDSs: SVMs train and run an order of magnitude faster; SVMs scale much better; and SVMs give higher classification accuracy.

We also address the related issue of ranking the importance of input features, which is itself a problem of great interest. Since elimination of the insignificant and/or useless inputs leads to a simplified problem and possibly faster and more accurate detection, feature selection is very important in intrusion detection.

Two methods for feature ranking are presented: the first one is independent of the modeling tool, while the second method is specific to SVMs. The two methods are applied to identify the important features in the 1999 DARPA intrusion data set. It is shown that the two methods produce results that are largely consistent.

We present experimental results that indicate that SVM-based IDSs using a reduced number of features can deliver enhanced or comparable performance. Finally, an SVM-based IDS for class-specific detection is proposed.

1. INTRODUCTION

This paper concerns computer networks intrusion detection and the related issue of identifying important input features for intrusion detection. Intrusion detection is a problem of significance to transportation infrastructure protection owing to the fact that computer networks are at the core of the operational control of much of the nation's transportation. We use two types of learning machines to build Intrusion Detection Systems (IDSs): Artificial Neural Networks or ANNs (*1*) and Support Vector Machines or SVMs (*2*). Since the ability to identify the important inputs and redundant inputs of a classifier results in reduced problem size, faster training and possibly more accurate results, it is critical to be able to identify the important features of network traffic data for intrusion detection in order for the IDS to achieve maximal performance. Therefore, we also study feature ranking and selection, which is itself a problem of great interest in data mining and model construction based on experimental data.

Since most of the intrusions can be uncovered by examining patterns of user activities, many IDSs have been built by utilizing the recognized attack and misuse patterns to develop learning machines (*3*,*4*,*5*,*6*,*7*,*8*,*9*,*10*,*11*). In our recent work, SVMs are found to be superior to ANNs in many important respects of intrusion detection (*12*,*13*,*14*); we will therefore concentrate on SVMs and briefly summarize the results of ANNs.

The data we used in our experiments originated from MIT's Lincoln Lab. It was developed for intrusion detection system evaluations by DARPA and is considered a benchmark for intrusion detection evaluations (*15*).

We performed experiments to rank the importance of input features for each of the five classes (normal, probe, denial of service, user to super-user, and remote to local) of patterns in the DARPA data. It is shown that using only the important features for classification gives good accuracies and, in certain cases, reduces the training time and testing time of the SVM classifier.

In the rest of the paper, a brief introduction to the data we used is given in section 2. In section 3 we describe two methods for ranking the importance of input features. In section 4 we present the experimental results of using SVMs for feature ranking. In section 5 we present the experimental results of using ANNs. In section 6 we summarize our results and give a brief description of our proposed IDS architecture.

2. THE DATA

In the 1998 DARPA intrusion detection evaluation program, an environment was set up to acquire raw TCP/IP dump data for a network by simulating a typical U.S. Air Force LAN. The LAN was operated like a real environment, but being blasted with multiple attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted. Of this database a subset of 494021 data were used in our experiments reported in this paper, of which approximately 20% represent normal patterns, the rest 80% of patterns are attacks belonging to four different categories.

The four different categories of attack patterns are:

A. Denial of Service (DOS) Attacks: A denial of service attack is a class of attacks in which an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. Examples are Apache2, Back, Land, Mail bomb, SYN Flood, Ping of death, Process table, Smurf, Syslogd, Teardrop, Udpstorm.
B. User to Superuser or Root Attacks (U2Su): User to root exploits are a class of attacks in which an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Examples are Eject, Ffbconfig, Fdformat, Loadmodule, Perl, Ps, Xterm.
C. Remote to User Attacks (R2L): A remote to user attack is a class of attacks in which an attacker sends packets to a machine over a network–but who does not have an account on that machine; exploits some vulnerability to gain local access as a user of that machine. Examples are Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock, Xsnoop.
D. Probing (Probe): Probing is a class of attacks in which an attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use this information to look for exploits. Examples are Ipsweep, Mscan, Nmap, Saint, Satan.

3. RANKING THE SIGNIFICANCE OF INPUTS

Feature selection and ranking (*16*,*17*) is an important issue in intrusion detection. Of the large number of features that can be monitored for intrusion detection purpose, which are truly useful, which are less significant, and which may be useless? The question is relevant because the elimination of useless features (the so-called audit trail reduction) enhances the accuracy of detection while speeding up the computation, thus improving the overall performance of an IDS. In cases where there are no useless features, by concentrating on the most important ones we may well improve the time performance of an IDS without affecting the accuracy of detection in statistically significant ways.

The feature ranking and selection problem for intrusion detection is similar in nature to various engineering problems that are characterized by:

- Having a large number of input variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of varying degrees of importance to the output **y**; i.e., some elements of **x** are essential, some are less important, some of them may not be mutually independent, and some may be useless or irrelevant (in determining the value of **y**)
- Lacking an analytical model that provides the basis for a mathematical formula that precisely describes the input-output relationship, $\mathbf{y} = \boldsymbol{F}(\mathbf{x})$
- Having available a finite set of experimental data, based on which a model (e.g. neural networks) can be built for simulation and prediction purposes

Due to the lack of an analytical model, one can only seek to determine the relative importance of the input variables through empirical methods. A complete analysis would require examination of all possibilities, e.g., taking two variables at a time to analyze their dependence or correlation, then taking three at a time, etc. This, however, is both infeasible (requiring 2^n experiments!) and not infallible (since the available data may be of poor quality in sampling the whole input space). In the following, therefore, we apply the technique of deleting one feature at a time (*16*) to rank the input features and identify the most important ones for intrusion detection using SVMs.

3.1 Performance-Based Ranking Method (PBRM)

We first describe a general (i.e., independent of the modeling tools being used), performance-based input ranking methodology: One input feature is deleted from the data at a time; the resultant data set is then used for the training and testing of the classifier. Then the classifier's performance is compared to that of the original classifier (based on all features) in terms of relevant performance criteria. Finally, the importance of the feature is ranked according to a set of rules based on the performance comparison.

The procedure is summarized as follows:

1. compose the training set and the testing set;

for each feature ***do*** the following
2. delete the feature from the (training and testing) data;
3. use the resultant data set to train the classifier;
4. analyze the performance of the classifier using the test set, in terms of the selected performance criteria;
5. rank the importance of the feature according to the rules;

3.2 Performance Metrics

To rank the importance of the 41 features (of the DARPA data) in an SVM-based IDS, we consider three main performance criteria: overall accuracy of (5-class) classification; training time; and testing time. Each feature will be ranked as "important", "secondary", or "insignificant", according to the following rules that are applied to the result of performance comparison of the original 41-feature SVM and the 40-feature SVM:

1. ***If*** *accuracy* decreases ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is important
2. ***If*** *accuracy* decreases ***and*** training time increases ***and*** testing time increases, ***then*** the feature is important
3. ***If*** *accuracy* decreases ***and*** training time decreases ***and*** testing time increases, ***then*** the feature is important
4. ***If*** *accuracy* unchanges ***and*** training time increases ***and*** testing time increases, ***then*** the feature is important
5. ***If*** *accuracy* unchanges ***and*** training time decreases *and* testing time increases, ***then*** the feature is secondary
6. ***If*** *accuracy* unchanges ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is secondary
7. ***If*** *accuracy* unchanges ***and*** training time decreases ***and*** testing time decreases, ***then*** the feature is unimportant
8. ***If*** *accuracy* increases ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is secondary
9. ***If*** *accuracy* increases ***and*** training time decreases ***and*** testing time increases, ***then*** the feature is secondary
10. ***If*** *accuracy* increases ***and*** training time decreases ***and*** testing time decreases, ***then*** the feature is unimportant

According to the above rules, the 41 features are ranked into the 3 types of {Important features}, <Secondary features>, or (Unimportant features), for each of the 5 classes of patterns, as follows:

class 1: {1,3,5,6,8-10,14,15,17,20-23,25-29,33,35,36,38,39,41}, <2,4,7,11,12,16,18,19,24,30,31,34,37,40>, (13,32)
class 2: {3,5,6,23,24,32,33}, <1,4,7-9,12-19,21,22,25-28,34-41>, (2,10,11,20,29,30,31,36,37)
class 3: {1,3,5,6,8,19,23-28,32,33,35,36,38-41}, <2,7,9-11,14,17,20,22,29,30,34,37>, (4,12,13,15,16,18,19,21,3)
class 4: {5,6,15,16,18,32,33}, <7,8,11,13,17,19-24,26,30,36-39>, (9,10,12,14,27,29,31,34,35,40,41)
class 5: {3,5,6,24,32,33}, <2,4,7-23,26-31,34-41>, (1,20,25,38)

3.3 SVM-specific Feature Ranking Method

Information about the features and their contribution towards classification is hidden in the support vector decision function. Using this information one can rank their significance, i.e., in the equation

$$\mathbf{F}(\mathbf{X}) = \Sigma \mathbf{W}_i \mathbf{X}_i + \mathbf{b}$$

The point **X** belongs to the positive class if **F**(**X**) is a positive value. The point **X** belongs to the negative class if **F**(**X**) is negative. The value of **F**(**X**) depends on the contribution of each value of **X** and $\mathbf{W}_i$. The absolute value of $\mathbf{W}_i$ measures the strength of the classification. If $\mathbf{W}_i$ is a large positive value then the i^{th} feature is a key factor for positive class. If $\mathbf{W}_i$ is a large negative value then the i^{th} feature is a key factor for negative class. If $\mathbf{W}_i$ is a value close to zero on either the positive or the negative side, then the i^{th} feature does not contribute significantly to the classification. Based on this idea, a ranking can be done by considering the support vector decision function.

3.4 Support Vector Decision Function Ranking Method (SVDFRM)

The input ranking is done as follows: First the original data set is used for the training of the classifier. Then the classifier's decision function is used to rank the importance of the features. The procedure is:

1. Calculate the weights from the support vector decision function;
2. Rank the importance of the features by the absolute values of the weights;

According to the ranking method, the 41 features are placed into the 3 categories of {Important features}, <Secondary features> or (Unimportant features), for each of the 5 classes of patterns, as follows:

class 1: {1-6,10,12,17,23,24,27,28,29,31-34,36,39}, <11-14,16,19,22,25,26,30,35,37,38, 40,41>, (7-9,15,18,20,21)
class 2: {1-6,23,24,29,32,33}, <10,12,22,28,34-36,38-41>, (7-9,11,13-21,25-27,30,31,37,40)
class 3: {1,5,6,23-26,32,36,38,39}, <2,3,4,10,12,29,33,34>, (7-9,11,13-22,27,28,30,31,35-37,40,41)
class 4: {1-6,12,23,24,32,33}, <4,10,13,14,17,22,27,29,31,34,36,37,39>,
(7-9,11,15,16,18-21,25,26,28,30,35,38,40,41)
class 5: {1,3,5,6,32,33}, <2,4,10,12,22-24,29,31,34,36,37,38,40>, (7,-9,11,13-21,25-28,30,35,39,41)

4. EXPERIMENTS USING SVMs

SVMs are used, in each of the two methods, for ranking the importance of the input features. Once the importance of the input features was ranked, the classifiers were trained and tested with only the important features. Further, we validate the ranking by comparing the performance of the classifier using all input features to that using the important features; and we also compare the performance of a classifier using the union of the important features for all fives classes. (Because SVMs are only capable of binary classifications, we will need to employ five SVMs for the five-class identification problem in intrusion detection. But since the set of important features may differ from class to class, using five SVMs becomes an advantage rather than a hindrance, i.e., in building an IDS using five SVMs, each SVM can use only the important features for that class which it is responsible for making classifications.)

4.1 SVMs Briefly Explained

Support vector machines, or SVMs, are learning machines that place the training vectors in high-dimensional feature space, labeling each vector by its class. SVMs classify data by determining a set of vectors from the training set, called support vectors, which outlines a hyper plane in the feature space (*18,19,20*).

SVMs provide a generic mechanism to fit the surface of the hyper plane to the data through the use of a kernel function. The user may provide a function (e.g., linear, polynomial, or sigmoid) to the SVMs during the training process, which selects support vectors along the surface of this function. The number of free parameters used in the SVMs depends on the margin that separates the two classes but not on the number of input features, thus SVMs do not require a reduction in the number of features in order to avoid over fitting--an apparent advantage in applications such as intrusion detection. Another primary advantage of SVMs is the low expected probability of generalization errors.

There are other reasons that we use SVMs for intrusion detection. The first is speed: as real-time performance is of primary importance to IDSs, any classifier that can potentially run “fast” is worth considering. The second reason is scalability: SVMs are relatively insensitive to the number of data points and the classification complexity does not depend on the dimensionality of the feature space (*18*), so they can potentially learn a larger set of patterns and thus be able to scale better than neural networks. Finally, SVMs give highly accurate classification of the patterns, as will be seen in the next section.

4.2 SVM Performance Statistics

Our results are summarized in the following tables. Table 1 gives the performance results of the five SVMs for each respective class of data. Table 2 shows the results of SVMs performing classification, with each SVM using as input the important features for all five classes. Table 3 shows the results of SVMs performing classification, with each SVM using as input the important and secondary features for each respective class. Table 4 shows the result of SVMs performing classification, with each SVM using as input the union of the important features for each class as obtained from PBR; the union has 30 features. Table 5 shows the results of SVMs performing classification, with each SVM using as input the union of the important features for each class as obtained from the SVDFR ranking; the union has 23 features.

The features identified as important by both ranking methods are described below:

- Duration: Length of the connection made by the destination system to the host system
- Service: Network service used by the destination system to connect to the host system
- Source bytes: Number of bytes sent from the host system to the destination system
- Destination bytes: Number of bytes sent from the destination system to the host system
- Count: Number of connections made to the same host system in a given interval of time
- Service count: Number of connections made to the same service on the same host system in a given interval of time
- Destination host count: Number of connections made by the same destination system to the same host system in a given interval of time
- Destination host service count: Number of connections made by the same destination system to the same service on the same host system in a given interval of time

5. EXPERIMENTS USING NEURAL NETWORKS

This section summarizes the authors’ recent work in comparing ANNs and SVMs for intrusion detection (*12,13,14*). Since a (multi-layer feedforward) ANN is capable of making multi-class classifications, a single ANN (Scaled

Conjugate Gradient Decent), is employed to perform the intrusion detection, using the same training and testing sets as those for the SVMs.

Neural networks are used for ranking the importance of the input features, taking training time, testing time, and classification accuracy as the performance measure; and a set of rules is used for ranking. Therefore, the method is an extension of the feature ranking method described in (*16*) where the cement bonding quality problem was studied. Once the importance of the input feature was ranked, the ANNs are trained and tested with the data set containing only the important features. We then compare the performance of the trained classifier against the original ANN trained with data containing all input features.

5.1 Artificial Neural Networks

Artificial neural network (in the present context, multilayer, feedforward type networks) consists of a collection of highly-interconnected processing elements to perform an input-output transformation. The actual transformation is determined by the set of weights associated with the links connecting elements. The neural network gains knowledge about the transformation to be performed by iteratively learning from a sufficient training set of samples or input-output training pairs. A well-trained network can perform the transformation correctly and also possess some generalization capability.

This section summarizes the use of different neural network training functions for the problem of intrusion detection. Since multi-layer feedforward ANNs are capable of making multi-class classifications, an ANN is employed to perform the intrusion detection, using the same training and testing sets as those for the SVMs. Table 6 gives the description of the different neural network training functions used for detecting intrusions (*21*).

5.2 ANN Performance Statistics

Table 8 gives the comparison of the ANN with all 41 features to that of using 34 important features that have been obtained by our feature-ranking algorithm described in section 3.1.

6. SUMMARY & CONCLUSIONS

A number of observations and conclusions are drawn from the results reported:

- SVMs outperform ANNs in the important respects of
  - **a.** scalability (SVMs can train with a larger number of patterns, while ANNs would take a long time to train or fail to converge at all when the number of patterns gets large)
  - **b.** training time and running time (SVMs run an order of magnitude faster)
  - **c.** prediction accuracy.
- SVMs easily achieve high detection accuracy (higher than 99%) for each of the 5 classes of data, regardless of whether all 41 features are used, only the important features for each class are used, or the union of all important features for all classes are used.

We note, however, that the difference in accuracy figures tend to be very small and may not be statistically significant, especially in view of the fact that the 5 classes of patterns differ in their sizes tremendously. More definitive conclusions can only be made after analyzing more comprehensive sets of network traffic data.

Regarding feature ranking, we observe that

- The two feature ranking methods produce largely consistent results: except for the class 1 (Normal) and class 4 (U2Su) data, the features ranked as Important by the two methods heavily overlap.
- The most important features for the two classes of ‘Normal’ and ‘DOS’ heavily overlap.
- ‘U2Su’ and ‘R2L’, the two smallest classes representing the most serious attacks, each has a small number of important features and a large number of secondary features.
- The performances of (a) using the important features for each class, Table 2, (b) using the union of important features, Table 4 and Table 5, and (c) using the union of important and secondary features for each class, Table 3, do not show significant differences, and are all similar to that of using all 41 features.
- Using the important features for each class gives the most remarkable performance: the testing time decreases in each class; the accuracy increases slightly for one class ‘Normal’, decreases slightly for two classes ‘Probe’ and ‘DOS’, and remains the same for the two most serious attack classes.

Our ongoing experiments include making 23-class (22 attack classes plus normal) feature identification using SVMs, for the purpose of designing a cost-effective and real time intrusion detection tool. Finally, we propose a five-SVM-based intrusion detection architecture as shown in figure 2, where the set of features to be used for each class can be selected by the user to optimize the overall performance of intrusion detection.

7. ACKNOWLEDGEMENTS

Partial support for this research received from ICASA (Institute for Complex Additive Systems Analysis, a division of New Mexico Tech) and IASP capacity building grant are gratefully acknowledged. We would also like to acknowledge many insightful suggestions from Dr. Jean-Louis Lassez that helped clarify our ideas and contributed to our work. Mr. Sanjay Veeramachaneni and Ms. Guadalupe Torres helped perform or repeat some of the experiments and we acknowledge their valuable assistance.

8. REFERENCES

1. Hertz J., Krogh A., Palmer R. G. (1991) *Introduction to the Theory of Neural Computation, Addison –Wesley.*
2. Joachims T. (1998) "Making Large-Scale SVM Learning Practical," LS8-Report, University of Dortmund, LS VIII-Report.
3. Denning D. (1987) "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 2, pp.222-232.
4. Kumar S., Spafford E. H. (1994) "An Application of Pattern Matching in Intrusion Detection," Technical Report CSD-TR-94-013. Purdue University.
5. Ghosh A. K. (1999). "Learning Program Behavior Profiles for Intrusion Detection," In Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring.
6. Cannady J. (1998) "Applying Neural Networks for Misuse Detection," Proceedings of 21st National Information Systems Security Conference, pp.368-381.
7. Ryan J., Lin M-J., Miikkulainen R. (1997) "Intrusion Detection with Neural Networks," in *Advances in Neural Information Processing Systems*, Vol. 10, Cambridge, MA: MIT Press.
8. Debar H., Becke M., Siboni D. (1992) "A Neural Network Component for an Intrusion Detection System," Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, pp.240-250.
9. Debar H., Dorizzi. B. (1992) "An Application of a Recurrent Network to an Intrusion Detection System," Proceedings of the IEEE International Joint Conference on Neural Networks, pp.78-83.
10. Luo J., Bridges S. M. (2000) "Mining Fuzzy Association Rules and Fuzzy Frequency Episodes for Intrusion Detection," *International Journal of Intelligent Systems*, John Wiley & Sons, Vol. 15, No. 8, pp.687-704.
11. Cramer M., et. al. (1995) "New Methods of Intrusion Detection using Control-Loop Measurement," Proceedings of the Technology in Information Security Conference (TISC) '95, pp.1-10.
12. Mukkamala S., Janoski G., Sung A. H. (2001) "Monitoring Information System Security," Proceedings of the 11th Annual Workshop on Information Technologies & Systems, pp.139-144.
13. Mukkamala S., Janoski G., Sung A. H. (2002) "Intrusion Detection Using Neural Networks and Support Vector Machines," Proceedings of IEEE International Joint Conference on Neural Networks, pp.1702-1707.
14. Mukkamala S., Janoski G., Sung A. H. (2002) "Comparison of Neural Networks and Support Vector Machines in Intrusion Detection," Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, 2002, http://www.mts.jhu.edu/~cidwkshop/abstracts.html
15. http://kdd.ics.uci.edu/databases/kddcup99/task.htm.
16. Sung A. H. (1998) "Ranking Importance of Input Parameters of Neural Networks," *Expert Systems with Applications*, Vol. 15, pp.405-41.
17. Lin, Y., Cunningham, G. A. (1995) "A New Approach to Fuzzy-Neural System Modeling," *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 2, pp.190-198.
18. Joachims T. (2000) "SVMlight is an Implementation of Support Vector Machines (SVMs) in C," http://ais.gmd.de/~thorsten/svm_light. University of Dortmund. Collaborative Research Center on "Complexity Reduction in Multivariate Data" (SFB475).
19. Vladimir V. N. (1995) *The Nature of Statistical Learning Theory*. Springer.
20. Joachims T. (2000) "Estimating the Generalization Performance of a SVM Efficiently," Proceedings of the International Conference on Machine Learning, Morgan Kaufman.
21. Demuth H., Beale M. (2000) *Neural Network Toolbox User's Guide*. Math Works Inc. Natick, MA.

List of Tables and Figures:

Tables:

Figures:

TABLE 1 Performance of SVMs Using 41 Features

Class	Training Time (sec)	Testing Time (sec)	Accuracy (%)
Normal	7.66	1.26	99.55
Probe	49.13	2.10	99.70
DOS	22.87	1.92	99.25
U2Su	3.38	1.05	99.87
R2L	11.54	1.02	99.78

TABLE 2 Performance of SVMs Using Important Features

Class	No of Features Identified		Training Time (sec)		Testing Time (sec)		Accuracy (%)	
	PBRM	**SVDFRM**	**PBRM**	**SVDFRM**	**PBRM**	**SVDFRM**	**PBRM**	**SVDFRM**
Normal	25	20	9.36	4.58	1.07	0.78	99.59	99.55
Probe	7	11	37.71	40.56	1.87	1.20	99.38	99.36
DOS	19	11	22.79	18.93	1.84	1.00	99.22	99.16
U2Su	8	10	2.56	1.46	0.85	0.70	99.87	99.87
R2L	6	6	8.76	6.79	0.73	0.72	99.78	99.72

TABLE 3 Performance of SVMs Using Important and Secondary Features

Class	No of Features Identified		Training Time (sec)		Testing Time (sec)		Accuracy (%)	
	PBRM	**SVDFRM**	**PBRM**	**SVDFRM**	**PBRM**	**SVDFRM**	**PBRM**	**SVDFRM**
Normal	39	34	8.15	4.61	1.22	0.97	99.59	99.55
Probe	32	21	47.56	39.69	2.09	1.45	99.65	99.56
DOS	32	19	19.72	73.55	2.11	1.50	99.25	99.56
U2Su	25	23	2.72	1.73	0.92	0.79	99.87	99.87
R2L	37	20	8.25	5.94	1.25	0.91	99.80	99.78

TABLE 4 Performance of SVMs Using Union of Important Features (Total 30)

Class	Training Time (sec)	Testing Time (sec)	Accuracy (%)
Normal	7.67	1.02	99.51
Probe	44.38	2.07	99.67
DOS	18.64	1.41	99.22
U2Su	3.23	0.98	99.87
R2L	9.81	1.01	99.78

TABLE 5 Performance of SVMs Using Union of Important Features (Total 23) as Ranked by SVDFM

Class	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	4.85	0.82	99.55
Probe	36.23	1.40	99.71
DOS	7.77	1.32	99.20
U2Su	1.72	0.75	99.87
R2L	5.91	0.88	99.78

TABLE 6 Description of Different Neural Network Training Functions

Function	Description
Traingd	Basic gradient descent. Slow response, can be used in incremental mode training.
Traingdm	Gradient descent with momentum. Generally faster than traingd. Can be used in incremental mode training.
Traingdx	Adaptive learning rate. Faster training than traingd, but can only be used in batch mode training.
Trainrp	Resilient back propagation. Simple batch mode training algorithm with fast convergence and minimal storage requirements.
Traincgf	Fletcher-Reeves conjugate gradient algorithm. Have smallest storage requirements of the conjugate gradient algorithms.
Traincgp	Polak-Ribiere conjugate gradient algorithm. Slightly larger storage requirements than traincgf. Faster convergence on some problems.
Traincgb	Powell-Beale conjugate gradient algorithm. Slightly larger storage requirements than traincgp. Generally faster convergence.
Trainscg	Scaled conjugate gradient algorithm. The only conjugate gradient algorithm that requires no line search. A very good general purpose training algorithm.
Trainbfg	BFGS quasi-Newton method. Requires storage of approximate Hessian matrix and has more computation in each iteration than conjugate gradient algorithms, but usually converges in fewer iterations.
Trainoss	One step secant method. Compromise between conjugate gradient methods and quasi-Newton methods.
Trainlm	Levenberg-Marquardt algorithm. Fastest training algorithm for networks of moderate size. Has memory reduction feature for use when the training set is large.

TABLE 7 Performance of Different Neural Network Training Functions

Function	No of epochs Trial 1	No of epochs Trial 2	Accuracy (%) Trail 1	Accuracy (%) Trail 2
Traingd	3500	3500	61.70	48.14
Traingdm	3500	3500	51.60	48.14
Traingdx	3500	3500	95.38	92.83
Trainrp	67	66	97.04	95.44
Traincgf	891	891	82.18	82.18
Traincgp	313	274	80.54	78.19
Traincgb	298	256	91.57	83.11
Trainscg	351	303	80.87	95.25
Trainbfg	359	359	75.67	75.67
Trainoss	638	638	93.60	93.60
Trainlm	17	16	76.23	74.04

TABLE 8 Neural Network Results Using all 41 Features and 34 Important Features

Number of features	Accuracy (%)	False positive rate (%)	False negative rate (%)	Number of epochs
41	87.07	6.66	6.27	412
34	81.57	18.19	0.25	27

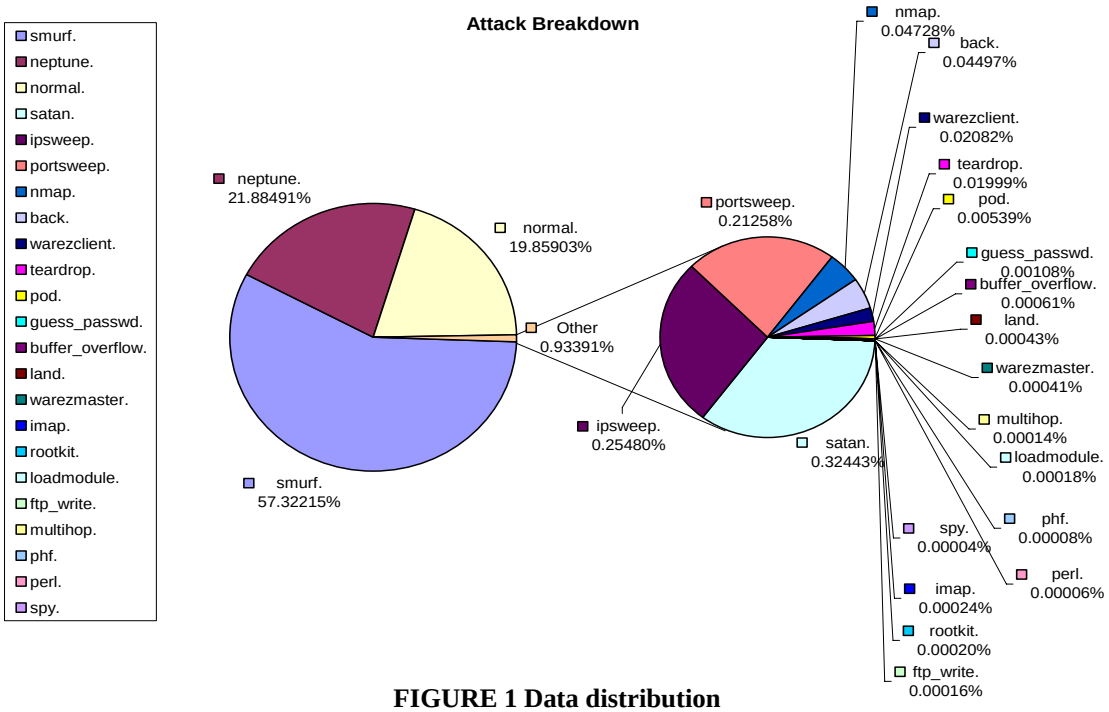

FIGURE 1 Data distribution

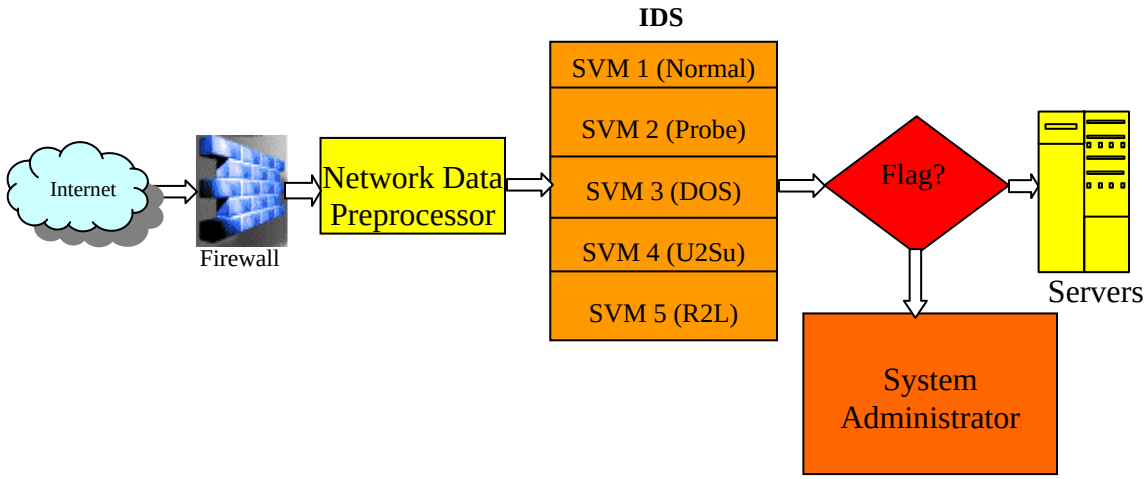

FIGURE 2 Proposed 5 class SVM intrusion detection architecture