Audit Data Reduction for Intrusion Detection

S. Mukkamala, G. R. Tadiparthi, N. Tummala, G. Janoski
{srinivas, gopal, ntummala, silfalco}@cs.nmt.edu
Department of Computer Science
New Mexico Institute of Mining and Technology
Socorro, New Mexico 87801, USA

Abstract:
Intrusion Detection Systems (IDS) have become important and widely-used tools for ensuring netwrok security. Since the amount of audit data that an IDS needs to examine is very large even for a small network, audit data reduction is often a necessary task. To maximize the time performance, scalability, and fast re-training or tuning of an IDS, irrelevant features in audit data must be identified and eliminated from examination by the IDS.

This paper concerns ranking the importance of input features for IDS. We use the DARPA data initially provided for the KDD'99 competition and perform experiments using neural networks (NN) and support vector machines (SVM). To rank the significance of the 41 input features in the data, we first build NN and SVM that achieve a high-level of accuracy. Next, input features are deleted, one at a time, and NN and SVM are trained based on the reduced input. The performance of the NN and SVM are then compared with the original NN and SVM to determine the significance of the deleted feature.

A number of simulation results are presented, including binary classifications (normal and attack) and five-class classifications (normal, and four classes of attacks). It is demonstrated that a large number of the (41) input features are unimportant and may be eliminated, without significantly lowering the performance of the IDS [17].

1. 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 true 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, of which 20% represent normal patterns.

Attack types fall into four main categories:

1. DOS: denial of service
2. R2L: unauthorized access from a remote machine
3. U2R: unauthorized access to local super user (root) privileges
4. Probing: surveillance and other probing

Table 1 below shows 22 different exploits that were used in the intrusion detection evaluation.

Table 1: Attacks in the DARPA evaluation.

Attack Class	**OS: Solaris**	**OS: SunOS**	**OS: Linux**
Denial of Service **Denial of Service**	Apache2 Back Mail bomb Neptune Ping of death	Apache2 Back Mail bomb Neptune Ping of death	Apache2 Back Mail bomb Neptune Ping of death
(cont.)	Process table Smurf Syslogd UDP storm	Process table Smurf Syslogd UDP storm	Process table Smurf Syslogd UDP storm
Remote to User	Dictionary Ftp-write Guest Phf Xlock Xnsnoop	Dictionary Ftp-write Guest Phf Xlock Xnsnoop	Dictionary Ftp-write Guest Imap Named Phf Sendmail Xlock Xnsnoop
User to Super-user	Eject Ffbconfig Fdformat Ps	Load module Ps	Perl Xterm
Probing	Ip sweep Mscan Nmap Saint Satan	Ip sweep Mscan Nmap Saint Satan	Ip sweep Mscan Nmap Saint Satan

2. SVM BASED TRAINNING

In our first set of experiments, the data consists of 14000 randomly generated points, with a number of data from each class in proportion to its size. We used a training set of 7000 data points with, respectively, 41 features and 13 features [16] each. The results are summarized in the following table.

In our second set of experiments, we perform 5-class classification. The (training and testing) data set contains 4562 randomly generated points from the five classes, with the number of data from each class proportional to its size, except that the smallest class is completely included. The normal data belongs to class1 (C1), denial of service belongs to C2, probe belongs to C3, remote to user belongs to C4, user to super user belongs to class C5. We used a training set of 2282 data points with 41 features for five class classification as described in section 3.

The results are summarized in the following table [17]. As can be seen, SVMs demonstrate higher performance than neural networks, in terms of training time (SVM trains at a speed that is an order of magnitude faster than that for neural networks), running time (running 5 SVMs, even serially, for 5-class identification, takes *less* time than running a single neural network for making the same 5-class identification), and scalability (SVMs can train with larger data sets).

Table 2: SVM training results.

Training results	**Experiment 1**	**Experiment 2**
Data set	14000	14000
Training set	7000	7000
# of features	41	13
Kernel	RBF	RBF
Gamma value	0.000001	0.000001
C value	1000	1000
CPU run time	52.02 sec	108.62 sec
# of misclassifications	15	22
# of iterations	11605	23766
Max difference	0.00099	0.00095
# of Support vectors	209 (53 at upper bound)	163 (92 at upper bound)
Liner loss	40.45970	65.25182
Normalization of weight vector	159.75859	203.82591
# of kernel evaluations	3798517	4358680
Training results	**Experiment 3**	**Experiment 4**
Data set	4562	4562
Training set	2282	2282
# of features	41	41
Kernel	RBF	RBF
Gamma value	0.000006	0.000006
C value	1000	1000
CPU run time sec	7.51 sec	13.36 sec
# of misclassifications	0	23
# of iterations	1338	23766
Max difference	0.00995	0.00100
# of Support vectors	174 (5 at upper bound)	157 (132 at upper bound)
Liner loss	0.90083	66.79142
Normalization of weight vector	93.79964	371.32366
# of kernel evaluations	891776	1066511
Training results	**Experiment 5**	**Experiment 6**
Data set	4562	4562
Training set	2282	2282
# of features	41	41
Kernel	RBF	RBF
Gamma value	0.000006	0.000006
C value	1000	1000
CPU run time	14.39 sec	7.22 sec
# of misclassifications	39	1
# of iterations	11063	2263
Max difference	0.00097	0.00095
# of Support vectors	272 (163 at upper bound)	112 (8 at upper bound)
Liner loss	96.14469	2.86755
Normalization of weight vector	273.28483	117.93924
# of kernel evaluations	1208449	881258
Training results	**Experiment 7**	
Data set	4562	
Training set	2282	
# of features	41	
Kernel	RBF	
Gamma value	0.000006	
C value	1000	
CPU run time	1.96 sec	
# of misclassifications	0	
# of iterations	576	
Max difference	0.00079	
# of Support vectors	36 (2 at upper bound)	
Liner loss	0.54599	
Normalization of weight vector	73.38302	
# of kernel evaluations	301750	

2.1 Testing

In our first set of experiments, the test set consists of 7000 data points with 41 features and 13 features. In our second set of experiments five-class classification as described in section 3 the test set consists of 2800 data points. Results are given in table4.

Table 3: SVM testing results.

Testing	Exp 1	Exp 2	Exp 3	Exp 4
Test data set	7000	7000	2200	2200
# of features	41	13	41	41
Accuracy %	99.53	99.52	98.99	99.08
CPU run time sec	1.60	1.06	0.29	0.23
# of mis-classifications	33	35	42	20
Testing	**Exp 5**	**Exp 6**	**Exp 7**	
Test data set	2200	2200	2200	
# of features	41	41	41	
Accuracy %	98.55	98.46	99.65	
CPU run time sec	0.70	0.36	1.96	
# of mis-classifications	33	35	8	

Table 5: Results of the second test set with 41 features and 2200 data points for five-class classification.

	C1	C2	C3	C4	C5	%
C1	536	16	9	1	0	95
C2	2	539	20	0	0	96
C3	4	1	521	42	0	91.7
C4	2	1	0	556	4	98.7
C5	4	2	1	0	19	76
%	97.8	99.6	97	93.9	92	

The top-left entry of Table 6 shows that 536 of the actual "normal" [C1] test set were detected to be normal; the last column indicates that 95 % of the actual "normal" data points were detected correctly. In the same way, for the class 1 [C2] 539 of the actual "attack" test set were correctly detected; the last column indicates that 96% of the actual "C2" data points were detected correctly. The bottom row shows that 97.8% of the test set said to be "normal" indeed were "normal" and 99.6% of the tests set classified, as "C2" indeed belong to C2.

3. NEURAL NETWORK TRAINING

In our experiments, we use a dataset consisting of 14000 randomly generated data points from the 2 classes of attack and normal. From this dataset, we then randomly select a subset of 7000 data for training; and prepare two training sets, with 41 features and 13 features each, respectively. A multi-layer, feed forward network was trained using the scaled conjugate gradient decent algorithm with convergence criterion set to be MSE (mean square error) of 0.001. During the training process of using 41features, the goal was met in 538 epochs with MSE=0.000999; Using 13 features, the goal was reached in 608 epochs with MSE=0.000638. In our other set of experiments, the data consists of 4562 randomly generated points, with a number of data from each class in proportion to its size and the least class completely included. We used a training set of 2282 data points with, 41 features for five class classification as described in section 3. The results are summarized in the following table [17].

Table 6: Neural network training.

Training	Experiment 1	Experiment 2
# of features	41	13
# of data points	7000	7000
Architecture	[41,50,40,1]	[13,40,40,1]
Performance	0.000999	0.00638
Epochs	538	608
CPU time	30 min	38 min
Training	**Experiment 3**	
# of features	41	
# of data points	2282	
Architecture	[41,15,10,1]	
Performance	0.000864	
Epochs	3118	
CPU time	1hr5min	

3.1 Testing the Neural Network

The test set consisting of 7000 data points with 41 features and 13 features. The one with 41 features received 99.48% accuracy and the one with 13 features received 99.41%. The following table gives a comparison of the neural network detection performance using 41 and 13 features [17]. In our experiment number 3 for five-class classification as described in section 3 the test set consists of 2800 data points. Results are given in the following table.

Table 7 Neural network testing binary classification.

Training	Experiment 1	Experiment 2
# of features	41	13
# of data points	7000	7000
Architecture	[41,50,40,1]	[13,40,40,1]
Performance	99.48%	99.41%
Training	**Experiment 3**	
# of features	41	
# of data points	2200	
Architecture	[41,15,10,1]	
Performance	92.6%	

Table 8 Neural network testing for five class classification

	C1	C2	C3	C4	C5	%
C1	547	17	0	0	0	97
C2	20	528	8	1	4	94.1
C3	0	5	476	17	73	83.8
C4	1	10	0	552	0	98.4
C5	0	0	6	6	12	48
%	99.2	99.8	86.2	97.5	28	

4. FEATURE RANKING

We used the method of deleting one feature at a time to rank the importance of each feature towards the over all efficiency and effectiveness, this was done in order to develop an cost effective and efficient intrusion detection system.

4.1 Experiments

We used neural networks for ranking the effectiveness. Here the same architecture [41,15,10,1] was used; and depending on the accuracy achieved on the test sets, unequal weighted effectiveness (C1-5%, C2-20%, C3-10%, C4-35%, C5-30%) and equally weighted effectiveness (each class 20%) of classification was determined. Using the unequal weighted effectiveness and equal weighted effectiveness as the basis, the features were ranked. After the features are ranked, we performed experiments by deleting the least significant features and then compared the unequal weighted effectiveness and equal weighted effectiveness to the experiment using all the 41 features. The table below shows the performance achieved by deleting that particular feature, based on the performance metrics importance of a particular feature can be derived.

Table 9 Performance of the neural networks after deleting a particular feature

#	**Feature deleted**	**Unequally Weighted Effectiveness**	**Equally Weighted Effectivenes s**
1	duration	71.7 %	80.1 %
2	protocol type	65.4	78
3	service	78.9	85.6
4	flag	84.3	88.0
5	src_bytes	86.4	89.8
6	dst_bytes	67.2	80.2
7	land	80.6	86.0
8	wrong_fragme nt	85.4	88.5
9	urgent	77.0	81.0
10	hot	70.04	81.2
11	num_failed_lo gins	82.2	87.4
12	logged in	70.1	79.9
13	num_compro mised	65.0	77.1
14	root_shell	65.9	78.6
15	su_attempted	67.2	77.8
16	num_root	66.3	77.8
17	num_file_creat ions	66.6	78.1
18	num_shells	76.9	82.7
19	num_access_fi les	70.8	76.6
20	num_outbound _cmds	65.5	77.9
21	is_host_login	65.5	77.9
22	is_guest_logn	67.8	77.9
23	count	64.4	77.1
24	srv_count	65.3	77.1
25	serror_rate	80.1	85.1
26	srv_serror_rate	69.0	78.8
27	rerror_rate	80.2	85.1
28	srv_rerror_rate	64.3	76.4
29	same_srv_rate	77.5	82.8
30	diff_srv_rate	67.3	78.3
31	srv_diff_host_ rate	77.1	84.5
32	dst_host_count	84.4	87.7
33	dst_host_srv_c ount	70.3	80.4
34	dst_host_same _srv_rate	81.7	84.4
35	dst_host_diff_sr v_rate	85.4	87.5
36	dst_host_same _src_port_rate	63.5	75.9
37	dst_host_srv_d iff_host_rate	63.2	75
38	dst_host_serro r_rate	66.7	78
39	dst_host_srv_s error_rate	64.1	76.7

40	dst_host_rerror_rate	65.1	75.8
41	dst_host_srv_rerror_rate	72.1	82.1

Considering performance as the basis we found that feature numbers 2,6,13,14,15,16,17,21,22,23,24,26,28,30,36,37,38,39, 40 were important for detecting the attack and normal patters for five-class classification. Tables below show the experimental results on the reduced features and on the one with all the 41 features.

Table 10 Performance matrix with all 41 features

	C1	C2	C3	C4	C5	%
C1	523	40	0	0	0	92.9
C2	19	529	7	2	4	94.3
C3	0	4	472	13	79	83.1
C4	1	3	14	545	0	96.8
C5	0	0	2	6	17	68
%	96.3	91.8	95.4	96.3	17	

Table 11 Performance matrix with 19 most important features

	C1	C2	C3	C4	C5	%
C1	434	129	0	0	0	77.1
C2	6	491	17	26	21	87.5
C3	50	32	461	15	10	81.2
C4	2	13	4	534	10	94.8
C5	0	6	0	0	19	76
%	88.2	73.2	95.6	92.9	31.7	

The weighted effectiveness improved when the least significant features were removed, but there was a decrease in un-weighted effectiveness.

Effectiveness	41features	19features
Weighted effectiveness	67.9	69.8
Unweighted effectiveness	79.4	76.3

5. CONCLUSIONS

We have performed a number of experiments to measure the performance of support vector machines and neural networks in intrusion detection, using the DARPA data for intrusion evaluation. Classifications were performed on the binary (attack / normal), as well as five-class classifications.

Both SVMs and neural networks deliver highly accurate (99% and higher) performance, with SVMs showing slightly better results. Further, when a reduction is performed to reduce the 41 features to the 13 most significant, both SVMs and neural networks again were able to train to deliver accurate results for binary classification. In terms of the five-class classification, we found using only 19 most important (of the 41) features, the change in accuracy was statistically insignificant. But the reduction in features can be expected to reduce the cost of detection and the overhead of the intrusion detection as a whole.

Our ongoing experiments include making 23-class (22 specific attacks and normal) feature identification using SVMs and neural networks, for designing an cost-effective and real time intrusion detection tool.

ACKNOWLEDGEMENTS

Partial support for this research received from ICASA (Institute for Complex Additive Systems Analysis, a division of New Mexico Tech) is gratefully acknowledged. The fourth author also acknowledges her partial support received from Sandia National Laboratories under the Rio Grande Educational Initiative. We also would like to thank Dr. Jean-Louis Lassez, David Duggan, and Bob Hutchinson for insightful conversations regarding this research.

REFERENCES

[1] Ryan J., Lin M-J., Miikkulainen R. (1998) *Intrusion Detection with Neural Networks*, Advances in Neural Information Processing Systems 10, Cambridge, MA: MIT Press.

[2] Kumar S., Spafford E.H. (1994) *An Application of Pattern Matching in Intrusion Detection*, Technical Report CSD-TR-94-013, Purdue University.

[3] 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, pp. 15:687-703.

[4] Demuth H., Beale M. (2000) *Neural Network Toolbox User's Guide*, Math Works, Inc. Natick, MA.

[5] Sung A.H. (1998) *Ranking Importance of Input Parameters Of Neural Networks*, Expert Systems with Applications, pp. 15:405-411.

[6] 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.

[7] 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.

[8] Debar H., Dorizzi B. (1992) *An Application of a Recurrent Network to an Intrusion Detection System*, Proceedings of the International Joint Conference on Neural Networks. pp. 78-483.

[9] Denning D. (1987) *An Intrusion-Detection Model*, IEEE Transactions on Software Engineering, Vol SE-13, No 2.

[10] Ghosh A.K. (1999). *Learning Program Behavior Profiles for Intrusion Detection*, USENIX.

[11] Cannady J. (1998) *Artificial Neural Networks for Misuse Detection*, National Information Systems Security Conference.

[12] Vladimir V.N. (1995) *The Nature of Statistical Learning Theory*, Springer.

[13] 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).

[14] Joachims T. (1998) *Making Large-Scale SVM Learning Practical*. LS8-Report, University of Dortmund, LS VIII-Report.

[15] Joachims T. (2000) *Estimating the Generalization Performance of a SVM Efficiently*, Proceedings of the International Conference on Machine Learning, Morgan Kaufman.

[16] http://kdd.ics.uci.edu/databases/kddcup99/task.htm.

[17] S. Mukkamala, G.Kakarla, A.H. Sung, S.Veeramachaneni (2002) *Intrusion Detection: Comparison of Support Vector Machines and Neural Networks*, IASTED Artificial Intelligence and Soft Computing conference (submitted).