Detecting Insider Attacks Using Non-negative Matrix Factorization

Jan Platoš, Václav Snášel, Pavel Krömer
Department of Computer Science
VŠB-Technical University of Ostrava
Ostrava, Czech Republic
{*jan.platos,vaclav.snasel,pavel.kromer.fei*}*@vsb.cz*

Ajith Abraham
Machine Intelligence Research Labs (MIR Labs), USA
ajith.abraham@ieee.org

***Abstract*—It is a fact that vast majority of attention is given to protecting against external threats, which are considered more dangerous. However, some industrial surveys have indicated they have had attacks reported internally. Insider Attacks are an unusual type of threat which are also serious and very common. Unlike an external intruder, in the case of internal attacks, the intruder is someone who has been entrusted with authorized access to the network. This paper presents a Non-negative Matrix Factorization approach to detect inside attacks. Comparisons with other established pattern recognition techniques reveal that the Non-negative Matrix Factorization approach could be also an ideal candidate to detect internal threats.**

***Keywords*-non-negative matrix factorization, intrusion detection**

I. Introduction

One of the important threats to computer systems has traditionally been the insider attacks. These class of attackers have a substantial amount of knowledge about the network architecture, files, systems etc. It has become a practice that many organizations only focus on protecting the network from external intruders and as a result insider attacks may not be discovered for a long period. Internal intruders can plant trojan horses or browse through the network file system, overload the system, cause a system crash etc. While browsing of unauthorized files violates the confidentiality, trojan horses are a threat to both the integrity and confidentiality of data/system and overloading/crashing directly affects the availability of the network/system. Unfortunately, these type of attacks can be extremely difficult to detect or protect against, without deploying a proper detecting and prevention mechanism.

Though different computer or network systems may have different definitions of security, early attempts of the protection mechanisms include authentication or identification, encryption, access control, etc. The goal of these mechanisms is to prevent unauthorized users from compromising the data confidentiality, data and communications integrity, and assurance against denial-of-service. It has been noticed that such prevention-based techniques cannot assure the security of the systems being protected. For example, in the year 2000, the so-called Distributed Denial of Service (DDoS) attacks stopped several major commercial sites, including Yahoo and CNN, from functioning normally, though they were protected by prevention-based techniques. Intrusion Detection Systems (IDS) were proposed to complement the prevention-based security measures. An intrusion is defined to be a violation of the security policy of the system; intrusion detection thus refers to the mechanisms that are developed to detect the violation of the system security policy. Intrusion detection is not introduced to replace the prevention-based techniques such as authentication and access control; instead, it is intended to be used along with the existing security measures and detect the actions that bypass the security control of the system. Thus, intrusion detection is usually considered as a second line of defense for computer and network systems.

Intrusion detection is defined to be the problem of identifying users or hosts or programs that are using a computer system without authorization and those who have legitimate access to the system but are abusing their privileges. Several types of intrusion detection systems are in use [1], [2], [3], [4], [5], [6]. Configuring an IDS to detect internal attacks can be difficult. Part of the IDS challenge lies in creating a good recognition engines. The reason the recognition engine needs to be different is due to the fact that different network users require a different amount of access to different services, servers, and systems for their work. Once any attack patterns/behavior are identified, the system administrators will be able to identify any network users who pose a threat to network or system security.

Rest of the paper is organized as follows. In Section 2, we present some theoretical background of non-negative matrix factorization approach. Experiment details are provided in Section 3 followed by conclusions in the last Section.

II. Non-negative Matrix Factorization

Since the amount of audit data that an IDS needs to examine is very large even for a small network, analysis is difficult even with computer assistance because extraneous features can make it harder to detect suspicious behavior patterns. Complex relationships exist between the features, which are difficult for humans to discover. IDS must therefore reduce the amount of data to be processed. This is very important if real-time detection is desired. Therefore, some form of data

reduction is required for IDS. Reduction can occur in one of several ways. Data that is not considered useful can be filtered, leaving only the potentially interesting data. Data can be grouped or clustered to reveal hidden patterns; by storing the characteristics of the clusters instead of the data, overhead can be reduced. Finally, some data sources can be eliminated using feature selection.

In complex classification domains, some data may hinder the classification process. Features may contain false correlations, which hinder the process of detecting intrusions. Further, some features may be redundant since the information they add is contained in other features. Extra features can increase computation time, and can impact the accuracy of IDS. Feature selection improves classification by searching for the subset of features, which best classifies the training data. Various approaches have been used for feature selection for the design of IDS [1], [6].

Matrix factorization or factor analysis is an important task helpful in the analysis of high dimensional real world data. There are several well known methods and algorithms for factorization of real data but many application areas including information retrieval, pattern recognition and data mining require processing of binary rather than real data see [7], [8], [9].

Non-negative Matrix Factorization (NMF) is really a class of decompositions whose members are not necessarily closely related to each other [10], [11]. They share the property that are designed for datasets in which attribute values are never negative - and its does not make sense for the decomposition matrices to contain negative values either. A side-effect of this non-negativity property is that the mixing of components that we have seen is one way to understand decompositions can only be additive.

With the standard vector space model, a set of data S can be expressed as a matrix V, where m is the number of attributes and n is the number of documents in S. Each column V_j of V is an encoding of a document in S and each entry v_{ij} of vector V_j is the value of i-th term with regard to the semantics of V_j, where i ranges across attributes. The NMF problem is defined as finding an approximation of V in terms of some metric (e.g., the norm) by factoring V into the product WH of two reduced-dimensional matrices W and H. Each column of W is a basis vector. It contains an encoding of a semantic space or concept from V and each column of H contains an encoding of the linear combination of the basis vectors that approximates the corresponding column of V. Dimensions of W and H are $m \times k$ and $k \times n$, where k is the reduced rank. Usually k is chosen to be much smaller than n. Finding the appropriate value of k depends on the application and is also influenced by the nature of the collection itself. Common approaches to NMF obtain an approximation of V by computing a (W, H) pair to minimize the Frobenius norm of the difference $V - WH$. The matrices W and H are not unique. Usually H is initialized to zero and W to a randomly generated matrix where each $W_{ij} > 0$ and these initial values are improved with iterations of the algorithm described in [12].

For any given matrix V, matrix W has k columns or basis vectors that represent k clusters, matrix H has n columns that represent n documents. A column vector in H has k components, each of which denotes the contribution of the corresponding basis vector to that column or document. The clustering of documents is then performed based on the index of the highest value of k for each document. For document i ($i = 1 \ldots n$), if the maximum value is the j-th entry ($j = 1 \ldots k$), document i is assigned to cluster j. Thus, NMF can be used to organize data collections into partitioned structures or clusters directly derived from the nonnegative factors. Potential applications include the monitoring, tracking and clustering of semantic features (topics) and can be used for intrusion detection.

III. Experimental Results

The data for the experiments was prepared by the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Labs [13]. The original data contains 744 MB data with 4,940,000 records. The data set has 41 attributes for each connection record plus one class label. Some features are derived features, which are useful in distinguishing normal connection from attacks. These features are either nominal or numeric.

Some features examine only the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. These are called same host features. Some features examine only the connections in the past two seconds that have the same service as the current connection and are called same service features. Some other connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. These are called host-based traffic features.

Our experiments have three phases namely data reduction, a training phase and a testing phase. In the data reduction phase, important variables for real-time intrusion detection are selected by feature selection. In the training phase, the matrix factorization method is used to construct a model using the training data to give maximum generalization accuracy on the unseen data. The test data is then passed through the saved trained model to detect intrusions in the testing phase. The data set for our experiments contains randomly generated 11982 records having 41 features [14]. The 41 features are labeled as in order as *A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, AA, AB, AC, AD, AF, AG, AH, AI, AJ, AK, AL, AM, AN, AO* and the class label is named as *AP*.

The training and test set used in the experiments comprises of 5092 and 6890 records respectively [1].

Several variants of NMF was tested. We tried 12, 17, 19, 25, 33 and 41 variables and processed 10, 50, 100, 500 and 1000 iterations for each number of variables. The results are illustrated in Tables I, II, III, IV and V. Since the matrices W and H were randomly initialized all experiments were repeated 5 times.

As evident, for 10 iterations we obtained 90% accuracy and 97% for 50 iterations and almost 100% after 500 iterations. The best ratio between time and precision is achieved for 50 and 100 iterations. Table V depicts the results for 1000 iterations. The results after 1000 iterations are not so good, because almost the same results were achieved after 100 and 500 iterations and we need less than 3% and 33% of time respectively.

IV. Conclusions

This paper presented a Non-negative Matrix Factorization approach to detect inside attacks. Empirical results reveal that the proposed NMF approach is efficient. It is only through detecting attacks/patterns and keeping logs uncorrupted that insider attacks can be minimized. Although insider attacks pose some unique challenges for security administrators, our research reveals that they can be easily detected by a well designed IDS. It is also important that IDS itself must be protected against attacks.

References

[1] S. Chebrolu, A. Abraham, and J. P. Thomas, "Hybrid feature selection for modeling intrusion detection systems," in *ICONIP*, ser. Lecture Notes in Computer Science, N. R. Pal, N. Kasabov, R. K. Mudi, S. Pal, and S. K. Parui, Eds., vol. 3316. Springer, 2004, pp. 1020–1025.

[2] S. Chavan, K. Shah, N. Dave, S. Mukherjee, A. Abraham, and S. Sanyal, "Adaptive neuro-fuzzy intrusion detection systems," in *ITCC (1)*. IEEE Computer Society, 2004, pp. 70–74.

[3] S. Mukkamala, A. H. Sung, and A. Abraham, "Modeling intrusion detection systems using linear genetic programming approach," in *IEA/AIE*, ser. Lecture Notes in Computer Science, R. Orchard, C. Yang, and M. Ali, Eds., vol. 3029. Springer, 2004, pp. 633–642.

[4] S. Mukkamala, A. Sung, and A. Abraham, "Intrusion detection using ensemble of soft computing paradigms," in *Third International Conference on Intelligent Systems Design and Applications, Intelligent Systems Design and Applications, Advances in Soft Computing*. Springer Verlag, Germany, 2003, pp. 239–248.

[5] S. Mukkamala, A. H. Sung, A. Abraham, and V. Ramos, "Intrusion detection systems using adaptive regression splines," in *ICEIS (3)*, 2004, pp. 26–33.

[6] V. Snasel, J. Platos, P. Kromer, and A. Abraham, "Matrix factorization approach for feature deduction and design of intrusion detection systems," in *FOURTH INTERNATIONAL SYMPOSIUM ON INFORMATION ASSURANCE AND SECURITY, PROCEEDINGS*, M. Rak, A. Abraham, and V. Casola, Eds., 2008, pp. 172–179, 4th International Symposium on Information Assurance and Security, Napoli, ITALY, SEP 08-10, 2008.

[7] D. Húsek, P. Moravec, V. Snásel, A. A. Frolov, H. Rezanková, and P. Polyakov, "Comparison of neural network boolean factor analysis method with some other dimension reduction methods on bars problem," in *PReMI*, ser. Lecture Notes in Computer Science, A. Ghosh, R. K. De, and S. K. Pal, Eds., vol. 4815. Springer, 2007, pp. 235–243.

[8] P. Moravec and V. Snasel, "Dimension reduction methods for image retrieval," in *ISDA '06: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 1055–1060.

[9] V. Snásel, D. Húsek, A. A. Frolov, H. Rezanková, P. Moravec, and P. Polyakov, "Bars problem solving - new neural network method and comparison," in *MICAI*, 2007, pp. 671–682.

[10] L. Eldén, *Matrix Methods in Data Mining and Pattern Recognition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007.

[11] D. Skillicorn, *Understanding Complex Datasets: Data Mining with Matrix Decomposition*, ser. Data Mining and Knowledge Discovery. Chapman & Hall/CRC, May 2007.

[12] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2000, pp. 556–562. [Online]. Available: citeseer.ist.psu.edu/lee01algorithms.html

[13] "Mit lincoln laboratory." [Online]. Available: http://www.ll.mit.edu/IST/ideval/

[14] "Kdd cup 99 intrusion detection data set." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz

Table I
EMPIRICAL RESULTS FOR 10 ITERATIONS

	Teaching				Testing			
Var.	Acc.[%]	Time[s]	FP[%]	FN[%]	Acc.[%]	Time[s]	FP[%]	FN[%]
12	72.66	0.4	27.28	0.06	80.23	0.34	19.72	0.05
17	78.15	0.62	21.77	0.07	87.92	0.55	12.01	0.07
19	81.69	0.74	18.23	0.09	90.68	0.62	9.22	0.10
25	79.86	0.98	20.03	0.11	87.51	0.82	12.44	0.06
33	82.38	1.39	17.52	0.10	90.56	1.11	9.39	0.05
41	84.64	1.79	15.23	0.13	90.32	1.52	9.61	0.07

Table II
EMPIRICAL RESULTS FOR 50 ITERATIONS

	Teaching				Testing			
Var.	Acc.[%]	Time[s]	FP[%]	FN[%]	Acc.[%]	Time[s]	FP[%]	FN[%]
12	87.22	2.79	12.66	0.12	92.63	4.17	7.28	0.09
17	89.81	4.19	10.12	0.07	94.46	7.48	5.46	0.08
19	89.76	4.32	10.13	0.12	93.54	6.23	6.34	0.12
25	91.08	6.04	8.85	0.07	91.52	8.72	8.40	0.07
33	94.49	9.46	5.39	0.12	96.27	11.5	3.57	0.16
41	95.31	13.27	4.56	0.13	96.87	17.77	3.00	0.13

Table III
EMPIRICAL RESULTS FOR 100 ITERATIONS

	Teaching				Testing			
Var.	Acc.[%]	Time[s]	FP[%]	FN[%]	Acc.[%]	Time[s]	FP[%]	FN[%]
12	90.20	6.11	9.68	0.12	94.68	8.6	5.22	0.10
17	90.07	11	9.86	0.07	94.64	17.66	5.27	0.09
19	89.44	11.41	10.46	0.10	94.37	17.05	5.54	0.09
25	92.36	15.49	7.56	0.08	95.90	23.48	4.00	0.10
33	94.12	22.04	5.79	0.09	93.99	31.09	5.87	0.15
41	95.25	27.94	4.65	0.09	97.12	35.93	2.74	0.14

Table IV
EMPIRICAL RESULTS FOR 500 ITERATIONS

	Teaching				Testing			
Var.	Acc.[%]	Time[s]	FP[%]	FN[%]	Acc.[%]	Time[s]	FP[%]	FN[%]
12	87.63	61.48	12.29	0.08	93.37	172.09	6.54	0.09
17	88.17	84.15	11.72	0.11	92.75	215.15	7.19	0.06
19	89.15	104.58	10.79	0.06	94.51	249.7	5.40	0.09
25	91.88	135.88	8.04	0.08	92.18	334.69	7.73	0.09
33	96.31	164.71	3.53	0.15	97.62	371.67	2.21	0.17
41	99.60	231.9	0.20	0.20	99.66	574.89	0.12	0.22

Table V
EMPIRICAL RESULTS FOR 1000 ITERATIONS

	Teaching				Testing			
Var.	Acc.[%]	Time[s]	FP[%]	FN[%]	Acc.[%]	Time[s]	FP[%]	FN[%]
12	84.16	191.66	15.76	0.09	92.21	416.08	7.70	0.08
17	89.10	263.78	10.82	0.08	93.77	644.63	6.15	0.08
19	87.82	306.83	12.11	0.07	93.71	736.58	6.21	0.07
25	91.29	371.79	8.63	0.08	94.10	904.07	5.82	0.08
33	95.54	439.17	4.31	0.15	96.27	1187.12	3.57	0.16
41	98.97	609.13	0.83	0.19	98.44	1585.49	1.34	0.22