Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2011, Vol. 3, No. 4

Anomaly Base Network Intrusion Detection by Using Random Decision Tree and Random Projection

A Fast Network Intrusion Detection Technique

Virendra Raghuwanshi

M tech (CSE) Scholar, Department of Computer Science & Engineering

Oriental Institute of Science & Technology, Bhopal-462021 Madhya Pradesh, India

E-mail: virendra.ex59@gmail.com

Mahendra Singh Sisodia

Assistant Professor, Department of Computer Science & Engineering

Oriental Institute of Science & Technology, Bhopal-462021 Madhya Pradesh, India

E-mail: mahendrasisodia@oriental.ac.in

Received: December 10, 2011 Accepted: December 28, 2011 Published: January 31, 2012

DOI: 10.5296/npa.v3i4.1342 URL: http://dx.doi.org/10.5296/npa.v3i4.1342

Abstract

Network Intrusion Detection Systems (NIDSs) have become an important component in network security infrastructure. Currently, many NIDSs are rule-based systems whose performances highly depend on their rule sets. Unfortunately, due to the huge volume of network traffic, coding the rules by security experts becomes difficult and time-consuming. Since data mining techniques can build network intrusion detection models adaptively, data mining-based NIDSs have significant advantages over rule-based NIDSs. Network and system security is of paramount importance in the present data communication environment. Hackers and intruders can create many successful attempts to cause the crash of the networks and web services by unauthorized intrusion. New threats and associated solutions to prevent these threats are emerging together with the secured system evolution. Network Intrusion Detection Systems are one of these solutions. The main function of NIDSs is to protect the resources from threats. It analyzes and predicts the behaviors of users, and then these behaviors will be considered an attack or a normal behavior. We use Random projection and Random Tree to detect network intrusions.

Keywords: Data Mining, Network Intrusion Detection System, Random Decision Tree, Random Projection, Detection Rate, Confusion Matrix, False Alarm.

1. Introduction

In recent years, Network Intrusion Detection System has become one of the hottest research areas in Computer Security. It is an important detection technology and is used as countermeasure to preserve data integrity and system availability during an intrusion. A network intrusion detection system monitors the activities of a given environment and decides whether these activities are malicious (intrusive) or legitimate (normal) based on system integrity, confidentiality and the availability of information resources. The network intrusion detection system collects information about the system being observed. This collected audit data is processed by the detector. The detector eliminates unnecessary information from the audit data and then makes a decision to evaluate the probability that these activities can be considered as a sign of an intrusion. Network intrusion detections defined to be the process of monitoring the events occurring in a computer system or network and noticeably different from normal system activities and thus detectable. A network intrusion detection system (NIDS) is a program that analyzes what happened or what has happened during an execution and tries to find indications that the computer has been misused. An NIDS does not eliminate the use of preventive mechanism but it works as the last defensive mechanism in securing the system. In section 2 we give a brief introduction to network intrusion detection system. In section 3, give the related work. Section 4 explains the proposed work. In section 5, we perform the result analysis and finally in section 6 give the conclusion.

2. Network Intrusion Detection System

When an intruder attempts to break into an information system or performs an action not legally allowed, we refer to this activity as an intrusion [1, 2]. Intruders can be divided into two groups, external and internal. The former refers to those who do not have authorized access to the system and who attack by using various penetration techniques. The latter refers to those with access permission who wish to perform unauthorized activities. Intrusion techniques may include exploiting software bugs and system misconfigurations, password cracking, sniffing unsecured traffic, or exploiting the design flaw of specific protocols [1]. A Network intrusion detection System is a system for detecting intrusions and reporting them accurately to the proper authority. Network intrusion detection systems are usually specific to the operating system that they operate in and are an important tool in the overall implementation an organization’s information security policy [2], which reflects an organization's statement by defining the rules and practices to provide security, handle intrusions, and recover from damage caused by security breaches. There are two generally accepted categories of network intrusion detection techniques: misuse detection and anomaly detection. Misuse detection refers to techniques that characterize known methods to penetrate a system. These penetrations are characterized as a ‘pattern’ or a ‘signature’ that the NIDS look for. The pattern/signature might be a static string or a set sequence of actions. System responses are based on identified penetrations. Anomaly detection refers to techniques that define and characterize normal or acceptable behaviors of the system (e.g., CPU usage, job execution time, system calls). Behaviors that deviate from the expected normal behavior are considered intrusions [3, 4]. NIDSs can also be divided into two groups depending on where they look for intrusive behavior: Network-based IDS (NIDS) and Host-based IDS. The former refers to systems that identify intrusions by monitoring traffic through network devices (e.g. Network Interface Card, NIC). NIDS are deployed on strategic point in network infrastructure. The NIDS can capture and analyze data to detect known attacks by comparing patterns or signatures of the database or detection of illegal activities by scanning traffic for anomalous activity. NIDS are also referred as "packet-sniffers", because it captures the packets passing through the of communication mediums .A host-based IDS monitors file and process activities related to a software environment associated with a specific host. Some host-based IDSs also listen to network traffic to identify attacks against a host [3, 4]. One example is known as blocking NIDS, which combines a host based IDS with the ability to modify firewall rules [5].

3. Related Work

Papadimitriou [6] use random projection in the preprocessing of textual data, prior to applying LSI. They present experimental results on an artificially generated set of documents. In their approach, the columns of the random projection matrix are assumed strictly orthogonal, but actually this need not be the case, as we shall see in our experiments. Kaski [7,8] has presented experimental results in using the random mapping in the context of the WEBSOM1 system. Kurimo [9] applies random projection to the indexing of audio documents, prior to using LSI and SOM. Kleinberg [10] and Indyk and Motwani [11] use random projections in nearest-neighbor search in a high dimensional Euclidean space, and also present theoretical insights. Dasgupta [12, 13] has used random projections in learning high-dimensional Gaussian mixture models. Other applications of random projection include e.g. [14,15].The problems of dimensionality reduction and similarity search have often been addressed in the information retrieval literature, and other approaches than random projection have been presented. Ostrovsky and Rabani [16] give a dimension reduction operation that is suitable for clustering. Agrawal et al. [17] map time series into frequency domain by the discrete Fourier transform and only retain the first few frequencies. Keogh and Pazzani [18] reduce the dimension of time series data by segmenting the time series into sections and indexing only the section means. Agarwal et al. [19] index market basket data by a specific signature table, which easens the similarity search. Wavelet transforms ([20, 21] etc.) are a common method of signal compression.

4. Proposed Work

A network based intrusion detection system monitor and analyze network traffics, and use multiple sensors for detecting intrusions from internal and external networks. Network intrusion detection system analyzes the information gathered by the sensors, and returns a synthesis of the input of the sensors to system administrator or intrusion prevention system. System administrator carries out the prescriptions controlled by the intrusion detection system. In our proposed network intrusion detection system, two algorithm namely random project and random decision tree are used shown in figure 1.

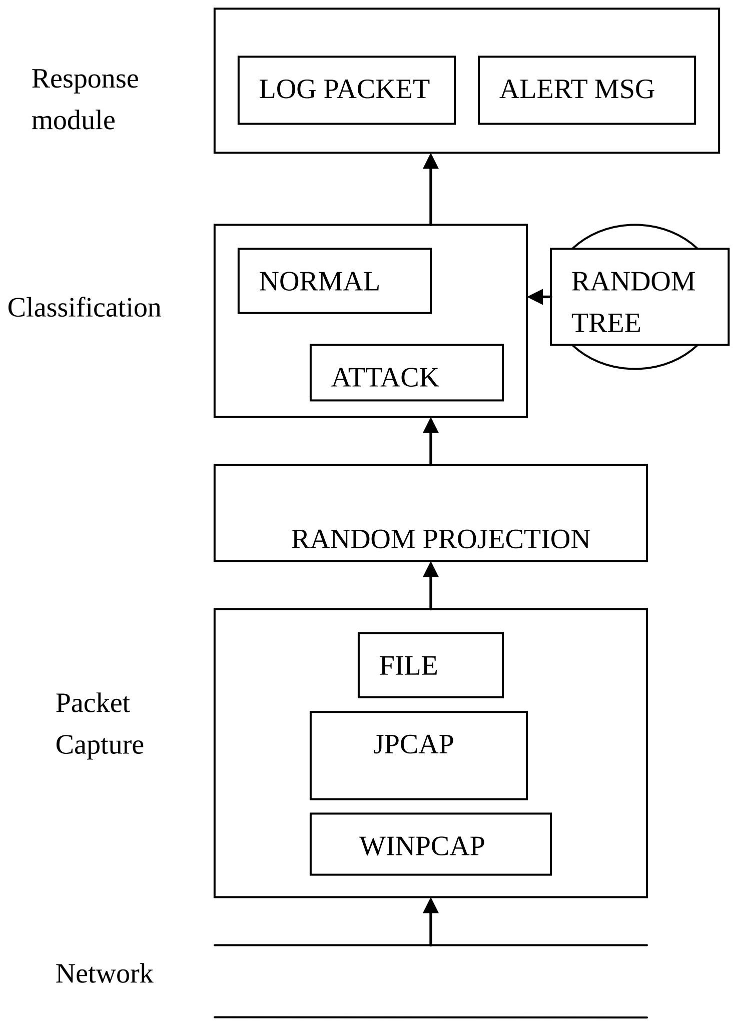

Figure 1. Proposed Network Intrusion Detection System

In our network intrusion detection system, we are using two algorithm namely random project and random decision tree. Random projection used for dimension reduction and decision tree used for classifier of reduced data. The description of Random Projection and Random Decision Tree is given below.

4.1 Random Projection

In our network intrusion detection system we have used random projection for dataset reduction. For testing the NIDSs system we have used KDDCup99 dataset [22]. Initially in KDDCup99 dataset there is 41 attributes. When we apply the random project on KDDCup99 dataset, it reduces 41 attributes into 10 attributes.

In random projection, the original d-dimensional data is projected to a k-dimensional (k << d) subspace through the origin, using a random K × d matrix R whose columns have unit lengths. Using matrix notation where is the original set of N d-dimensional observations, is the projection of the data onto a lower k-dimensional subspace.

$$X^{RP}_{K\times N} = R_{K\times d} X_{d\times N} \tag{1}$$

The key idea of random mapping arises from the Johnson Linden Strauss lemma [23]: if points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances between the points are approximately preserved. For a simple proof of this result, see [24,25]. Random projection is computationally very simple: forming the random matrix R and projecting the d × N data matrix X into k dimensions is of order O(dkN), and if the data matrix X is sparse with about c nonzero entries per column, the complexity is of order O(ckN) [15]. Strictly speaking, (1) is not a projection because R is generally not orthogonal. A linear mapping such as (1) causes significant distortions in the data set if R is not orthogonal. Orthogonalizing R is unfortunately computation-ally expensive. Instead, we can rely on a result presented by Hecht-Nielsen [26]: in a high-dimensional space, there exists a much larger number of almost orthogonal than orthogonal directions. Thus vectors having random directions might be sufficiently close to orthogonal, and equivalently would approximate an identity matrix. In our experiments, the mean squared difference between and an identity matrix was about 1/k per element .When comparing the performance of random projection to that of other methods of dimensionality reduction, it is instructive to see how the similarity of two vectors is distorted in the dimensionality reduction. We measure the similarity of data vectors either as their Euclidean distance or as their inner product. In the case of image data, Euclidean distance is a widely used measure of similarity. Text documents, on the other hand, are generally compared according to the cosine of the angle between the document vectors; if document vectors are normalized to unit length, this corresponds to the inner product of the document vectors. We write the Euclidean distance between two data vectors X_1 and X_2 in the original large-dimensional space as$\| X_1 - X_2 \|$. After the random projection, this distance is approximated by the scaled Euclidean distance of these vectors in the reduced space:

$$\sqrt{d/K} \quad \| R_{X1} - \quad R_{X2} \| \tag{2}$$

Where d is the original and k the reduced dimensionality of the data set. The scaling term $\sqrt{d/K}$ takes into account the decrease in the dimensionality of the data: according to the Johnson Linden Strauss lemma, the expected norm of a projection of a unit vector onto a random subspace through the origin is $\sqrt{K/d}$ [23].The choice of the random matrix R is one of the key points of interest. The elements of Rare often Gaussian distributed, but this need not be the case. Achlioptas [27] has recently shown that the Gaussian distribution can be re-placed by a much simpler distribution such as

$$rij = \sqrt{3} * \begin{cases} +1 & \text{with probality } \frac{1}{6} \\ 0 & \text{with probality } \frac{2}{3} \\ -1 & \text{with probality } \frac{1}{6} \end{cases} \tag{3}$$

In fact, practically all zero mean, unit variance distributions of would give a mapping that still satisfies the Johnson-Lindenstrauss lemma. Achlioptas' result means further computational savings in database applications, as the computations can be performed using integer arithmetics. In our experiments we shall use both Gaussian distributed random matrices and sparse matrices (3), and show that Achlioptas' theoretical result indeed has practical significance. In con-text of the experimental results, we shall refer to RP when the projection matrix is Gaussian distributed and SRP when the matrix is sparse and distributed according to (3). Otherwise, the shorthand RP refers to any random projection.

4.2 Random Decision Tree

Random decision tree a supervised machine learning algorithm used to classify multiple dimension data efficiently. In our proposed NIDS random decision tree used for classification of reduced KDDCup99 dataset [22] that is output of random projection. Random decision tree classify KDDCup99 dataset into normal or attack.

In most machine learning algorithms, the best approximation to the target function is assumed to be the "simplest" classifier that fits the given data, since more complex models tend to over fit the training data and generalize poorly. Ensemble methods such as Boosting and Bagging [9] combine multiple "base" classifiers to obtain new classifiers. It has been observed that ensemble methods can have significantly lower generalization error than any of the base classifiers on which they are based [28]. The base classifiers used in ensemble methods are usually "conventional" classifiers such as decision trees produced by C4.5, which are computationally expensive. The final step of combining these base classifiers can also be computationally intensive. However, Fan et al. [29] argue that neither of these steps (creating the classifiers and combining them) need be computationally burdensome to obtain classifiers with good performance. They present a fast and scalable ensemble method that performs better than the base classifiers, and frequently as well as the well-known ensemble classifiers. Counter intuitively, their ensemble classifier uses base classifiers that are created from randomly chosen decision trees, in which attributes for decision tree nodes are chosen at random instead of using a carefully defined criterion. The structure of the decision tree (that is, which attributes are in which internal nodes of the decision tree) is determined even before any data is examined. Data is then examined to modify and label the random tree. The end result based on creating an ensemble of random decision trees is an algorithm that scales well for large databases. The algorithm to build a single random decision tree is shown in Figure 1. The algorithm as presented works only for categorical attributes, though it can easily be extended to continuous-valued attributes by choosing random thresholds for the chosen attribute. The algorithm recursively creates the structure of the tree (Build Tree Structure), and then updates the statistics (Update Statistics, Add Instance) at the leaves by “filtering” each training instance through the tree. Each leaf node of the tree holds T counters, α[1], . . . , α[T], where T is the number of possible labels for training instances. After all the examples have been incorporated into the tree, the algorithm prunes away all internal and leaf nodes that did not encounter any of the examples in the training set. The running time of the algorithm is linear in the size of the database. The random decision tree classifier is an ensemble of such random decision trees. When a test instance needs to be classified, the posterior probability is output as the weighted sum of the probability outputs from the individual trees (see Figure 3). There are two important parameters to be chosen when using this ensemble method, namely (i) the height h of each random tree, and (ii) the number N of base classifiers.

```
Algorithm Random Decision Tree (RDT)
  Input: D, the training set, and
  X, the set of attributes.
  Output: A random decision tree R
  R = Build Tree Structure(X)
  Update Statistics(R, D)
  Prune sub trees with zero counts
  Return R
  Subroutine Build Tree Structure(X)
  If X = Φ then
  Return a leaf node
  Else
    Randomly choose an attribute F as testing attribute
  Create an internal node r with F as the attribute Assume F has m valid values.
  For i = 1 to m do
    $c_i$ = Build Tree Structure(X − {F})
    Add $c_i$ as a child of r
```

```
End for
End if
Return r

Subroutine Update Statistics(r, D)
For each x in D do
            Add Instance(r, x)
End for
Subroutine Add Instance(r, x)
        If r is not a leaf node then
            Let F be the attribute in r
            Let c represent the child of r that corresponds to the
value of F in x
     Add Instance(c, x)
      Else
            /* r is a leaf node */
            Let t be the label of x
            Let $\alpha[t]$ = # of t-labeled rows that reach r
            $\alpha[t] \leftarrow \alpha[t] + 1$
             End if
```

Figure 2. Random Decision Tree Algorithm

Using Simple combinatorial reasoning, Fan et al. [29] suggest that a good choice for the height is $h = m/2$, where m denotes the number of attributes. They also find that a value for N as low as 10 gives good results.

The advantage of creating a random tree is its training efficiency as well as its minimal memory requirements. The algorithm uses only one pass over the data to create a random decision tree. In a series of papers, Fan et al. [30], [31] show that the random decision tree algorithm is simple, efficient and accurate. They surmise that the reason for the superiority Of their ensemble method is that it optimally approximates for each example its true probability of being a member of a given class—that is, the random decision tree ensembles form efficient implementations of Bayes Optimal Classifiers.

```
Algorithm Classify
 Input: { $R_1 ... R_N$ }, an ensemble of RDTs, and
                              x, the row to be classified.
 Output: Probabilities for all possible labels
     For a tree $R_i$ let $\ell_i$ be the leaf node reached by x
     Let $\alpha_i[t]$ represent the count for label t in $\ell_i$

     $P(t/x) = \sum_{i=N}^{N} \alpha_i[t] / \left( \sum_{r} \sum_{i=1}^{N} \alpha_i \quad [T] \right)$

     Return probabilities for all t
```

Figure 3. Computing the probability for each possible label for a test instance.

5. Experiment and Results

For our experiments we are using of KDD CUP 99 dataset [22]. For our experiment we have selected 7473 instances randomly from KDD CUP 1999 dataset.

Table 1. Description of input KDDCup99 dataset.

Types of Attacks	Number of Instances
Normal	1091
DoS	4633
Prob2	1057
U2R	93
R2L	1057

KDD CUP 1999 dataset contains 41 fields as an attributes and 42nd field as a label. The 42nd field can be generalized as Normal, DoS, Probing, U2R, and R2L. The performances of each method are measured according to the Detection Rate and False Positive Rate using the following expressions:

$$Detection \quad Rate = \frac{TP}{TP + FP}$$

$$False \quad Alarm = \frac{FP}{FP + TN}$$

Where, FN is False Negative, TN is True Negative,TP is True Positive, and FP is False

Positive

The detection rate is the number of attacks detected by the system divided by the number of attacks in the data set. The false positive rate is the number of normal connections that are misclassified as attacks divided by the number of normal connections in the data set.

Table 2. Correctly and incorrectly classified instance by RPRDT and naïve bayes.

Instances Classification Type	RPRDT	Naïve Bayes
Correctly Classified Instances	7197 (96.3067 %)	4221 (56.4833 %)
Incorrectly Classified Instances	276 (3.6933 %)	3252 (43.5167 %)

In our approach Random projection using Random Decision Tree, the Correctly Classified Instances 7197 (96.3067 %) and Incorrectly Classified Instances is 276 (3.6933 %) which is much better and accurate in comparison to the Correctly Classified Instances 4221 (56.4833 %) and Incorrectly Classified Instances is 3252 (43.5167%) in naïve bayes shown in table 2.

A “Confusion Matrix” is sometimes used to represent the result of , as shown in Table 3 & Table 4 .The Advantage of using this matrix is that it not only tells us how many got misclassified but also what misclassifications occurred. For our model we get the following confusion matrix. Tables 3 showing the Result of Naïve Bayes classification [32] & Table 4 showing the Result of Random Decision Tree Using Random Projection.

Table 3. Confusion Matrix for Naïve Bayes.

Predicted Classes / Actual Classes	Dos	U2R	R2L	Probe	Normal
Dos	2954	180	29	1286	184
U2R	23	14	2	48	6
R2L	96	14	40	442	7
Probe	41	0	0	1016	0
Normal	503	9	1	381	197

Table 4. Confusion Matrix for Random Projection Using Random Decision Tree.

Predicted Classes / Actual Classes	Dos	U2R	R2L	Probe	Normal
Dos	4614	2	2	5	10
U2R	2	73	10	1	7
R2L	6	6	498	4	85
Probe	6	1	3	1045	2

Predicted Classes / Actual Classes	Dos	U2R	R2L	Probe	Normal
Normal	8	9	102	5	967

In table 5 show our approach give much better detection rate and low false alarm as compare to naïve bayes classification. In each category our method perform better than naïve bayes classification.

Table 5. Detection Rate and false alarm rate by Random Projection Using Random Decision Tree and Naïve Bayes.

Classification Algorithm / Attack Classes	Random Projection via Random Decision Tree		Naïve Bayes	
	Detection Rate	*False Alarm Rate*	*Detection Rate*	*False Alarm Rate*
Dos	0.996	0.008	0.638	0.233
U2R	0.785	0.002	0.151	0.028
R2L	0.831	0.017	0.067	0.005
Probing	0.989	0.002	0.961	0.336
Normal	0.886	0.016	0.181	0.031
Weight Avg.	0.963	0.009	0.565	0.198

In Figure 4 shows our approach give much better detection rate as compare to Naïve Bayes classification. In each category our method performs better than naïve bayes classification.

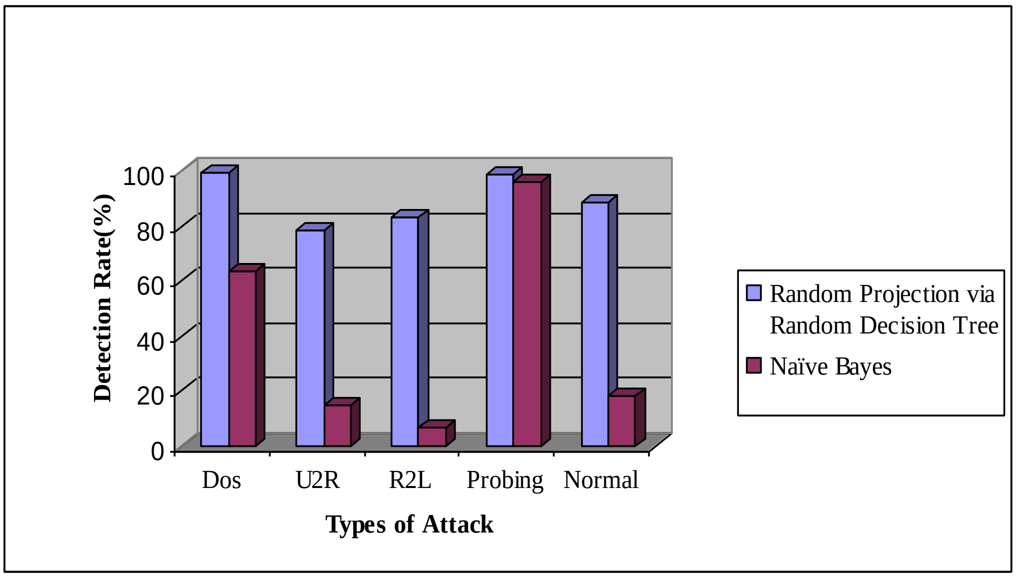

Figure 4. Comparison of Detection Rate of Random Projection Via Random Decision Tree and Naïve Bayes.

In Figure 5 show our approach give low false alarm as compare to Naïve Bayes classification. In each category our method performs better than naïve bayes classification other than R2L.

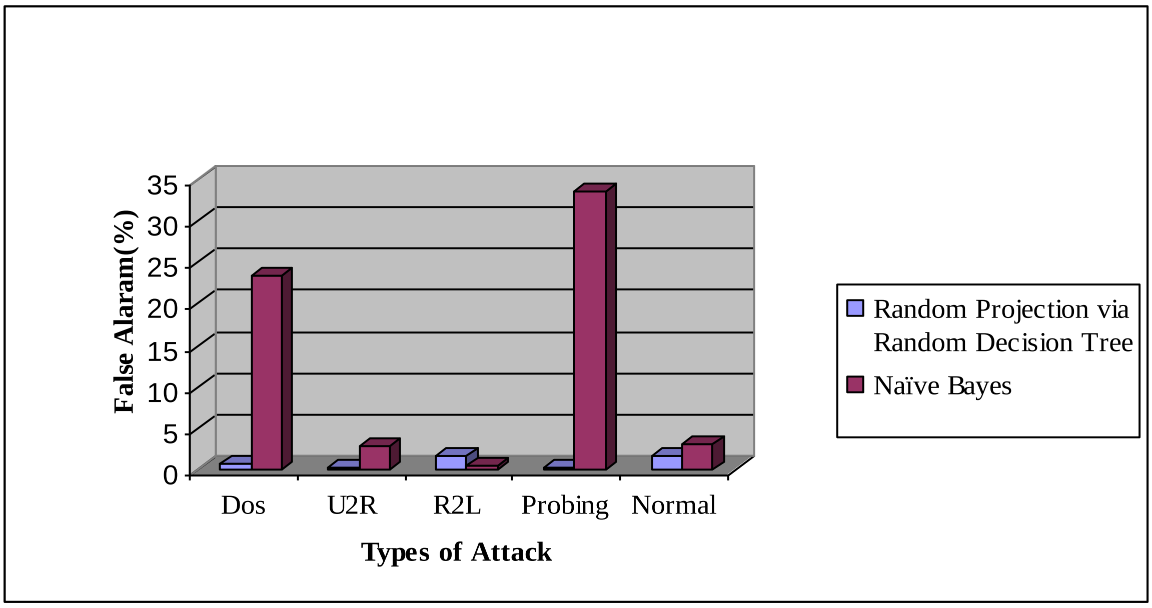

Figure 5. Comparison of False Alarms of Random Projection Via Random Decision Tree and Naïve Bayes.

6. Conclusion

We have presented new and promising experimental results on random projection in dimensionality reduction of high dimensional real world data sets. When comparing different methods for dimensionality reduction, the criteria are the amount of distortion caused by the method and its computational complexity. Our results indicate that random projection preserves the similarities of the data vectors well even when the data is projected to moderate numbers of dimensions; the projection is yet fast to compute. We conclude that random projection is a good alternative to traditional, statistically optimal methods of dimensionality reduction that are computationally infeasible for high dimensional data. Random projection does not suffer from the curse of dimensionality, quite contrary to the traditional methods. The advantage of creating a random tree is its training efficiency as well as its minimal memory requirements. The algorithm uses only one pass over the data to create a random decision tree.

References

[1] 1. Graham, Robert, "FAQ: Network Intrusion Detection Systems." Mar. 21, 2000. RobertGraham.com Homepage. Robert Graham., URL:http://www.robertgraham.com/pubs/network-intrusion-detection.html

[2] Jones, Anita. K. and Robert. S. Sielken.. "Computer System Intrusion Detection: A Survey." Technical Report. Department of Computer Science, University of Virginia, Charlottesville, Virginia. 2000.
[3] Bezroukov, Nikolai, "Intrusion Detection (general issues)." Softpanorama: Open Source Software Educational Society. 19 July 2003. URL: http://www.softpanorama.org/Security/intrusion_detection.shtml,
[4] McHugh, John, "Intrusion and Intrusion Detection." Technical Report. CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, 2001.
[5] Miller, Brad. L. and Michael J. Shaw, "Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization." In Proceedings of IEEE International Conf. on Evolutionary Computation, pp. 786-791. Nagoya University, Japan, 1996.
[6] C.H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis", In Proc. 17th ACM Symposium on the Principles of Database Systems, pages 159–168, 1998.
[7] S. Kaski, "Data exploration using self-organizing maps", In Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series, number 82. 1997. Dr.Tech. thesis, Helsinki University of Technology, Finland.
[8] S. Kaski, "Dimensionality reduction by random mapping", In Proc. Int. Joint Conf. on Neural Networks, volume 1, pages 413–418, 1998.
[9] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A New Explanation for the Effectiveness of Voting Methods," The Annals of Statistics, vol. 26, no. 5, pp. 1651–1686, 1998.
[10] J.M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions", In Proc. 29th ACM Symposium on Theory of Computing, pages 599–608, 1997
[11] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality", In Proc. 30th Symposium on Theory of Computing, Dallas, Texas, USA, May 23-26, 1998., pages 604–613.
[12] S. Dasgupta, "Learning mixtures of Gaussians", In 40th Annual IEEE Symposium on Foundations of Computer Science, pages 634–644, 1999.
[13] S. Dasgupta, "Experiments with random projection", Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence. June 30- July 3 2000, Stanford, CA, USA.
[14] R. I. Arriaga and S. Vempala, "An algorithmic theory of learning: robust concepts and random projection", In Proc. 40th Annual Symposium on Foundations of Computer Science, New York City, NY, October 17-19, 1999. Pages 616–623.
[15] S. Vempala, "Random projection: a new approach to VLSI layout", In Proc. 39th Annual Symposium On Foundations of Computer Science. 8-11 Nov 1998. Palo Alto, CA , USA, http://dx.doi.org/10.1109/SFCS.1998.743489
[16] R. Ostrovsky and Y. Rabani, "Polynomial time approximation schemens for geometric k-clustering", In Proc. 41st Symposium on Foundations of Computer Science, (FOCS 2000), 12-14 November 2000, Redondo Beach, California, USA. Pages 349–358. http://dx.doi.org/10.1109/SFCS.2000.892123
[17] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence

databases", In Proc. 4th Int. Conf. of Data Organization and Algorithms, Chicago, Illinois, USA, October 13-15, 1993. Pages 69–84.

[18] E. J. Keogh and M. J. Pazzani, "A simple dimensionality reduction technique for fast similarity search in large time series databases", In 4th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, PADKK 2000, Kyoto, Japan, April 18-20.

[19] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A new method for similarity indexing of market basket data", In Proc. Int. Conf. on Management of data ACM SIGMOD 99, Philadelphia, Pennsylvania, USA. June 1-3, 1999, pages 407–418.

[20] A. Graps, "An introduction to wavelets", IEEE Computational Science and Engineering, 2(2):50–61,1995.

[21] M. Sonka, V. Hlavac, and R. Boyle, "Image processing, analysis, and machine vision", PWS Publishing, 1998.

[22] KDDcup99, Dataset: kdd.ics.uci.edu/databases/kddcup99/kddcup99.htm

[23] W.B. Johnson and J. Lindenstrauss, "Extensions of Lipshitz mapping into Hilbert space", In Conference in modern analysis and probability, volume 26 of Contemporary Mathematics, pages 189–206. Amer. Math. Soc., 1984.

[24] P. Frankl and H. Maehara, "The Johnson Lindenstrauss lemma and the sphericity of some graphs", Journal of Combinatorial Theory, Ser. B, 44:355–362, 1988.

[25] S. Dasgupta and A. Gupta, "An elementary proof of the Johnson-Lindenstrauss lemma", Technical Report TR-99-006, International Computer Science Institute, Berkeley, California, USA, 1999.

[26] R. Hecht-Nielsen, "Context vectors: general purpose approximate meaning representations self-organized from raw data", In J.M. Zurada, R.J. Marks II, and C.J. Robinson, editors, Computational Intelligence: Imitating Life, pages 43–56. 1994.

[27] D. Achlioptas, "Database-friendly random projections", In Proc. ACM Symposium on the Principles of Database Systems, Santa Barbara, California, USA, May 21-23, 2001, Pages 274–281.

[28] M. Kurimo, "Indexing audio documents by using latent semantic analysis and SOM" In E. Oja and S. Kaski, editors, Kohonen Maps, pages 363–374. Elsevier, 1999.

[29] W. Fan, H. Wang, P. Yu, and S. Ma, "Is random model better? on its accuracy and efficiency", The Third IEEE International Conference on Data Mining 2003, p. 51-58. http://dx.doi.org/10.1109/ICDM.2003.1250902

[30] W. Fan, "On the optimality of probability estimation by random decision trees," in Nineteenth National Conference on Artificial Intelligence (AAAI-04), San Jose, California, July25-29, 2004., pp. 336–341.

[31] W. Fan, E. Greengrass, J. McCloskey, P. Yu, and K. Drummey, "Effective estimation of posterior probabilities: Explaining the accuracy of randomized decision tree approaches," Fifth IEEE International Conference on Data Mining, 27-30 Nov. 2005, pp. 154–161. http://dx.doi.org/10.1109/ICDM.2005.54

[32] Panda & Patra, "Network Intrusion Detection using naïve bayes" IJCSNS International Journal of Computer Science and Network Security, Vol.7 No.12, December 2007