Network Intrusion Detection in High Dimensional Space

Stanislav Marcek* Martin Drozda† Gabriel Juhas* Fedor Lehocki*
* Slovak University of Technology, Faculty of Electrical Engineering and Information Technology,
Ilkovičova 3, 81219 Bratislava, Slovakia.
† Leibniz University of Hannover, Simulation and Modeling Group, Faculty of Electrical Engineering and Computer Science,
Welfengarten 1, 30167 Hannover, Germany.
Email: stanislav.marcek@stuba.sk, drozda@sim.uni-hannover.de

***Abstract*—We propose and evaluate an approach for network intrusion detection in high dimensional space. This approach is based on an approximate solution to the nearest neighbor problem. Our evaluation is based on the KDD'99 data set, a Yahoo web spam data set and another set used in the NIPS'03 feature selection challenge. The approximate approach shows that good performance in terms of detection rate and false positives rate can be achieved.**

I. Introduction

Network intrusion has become a widespread problem with a very negative impact on the trustworthiness and reliability of Internet services. According to a Symantec report [1], there were more than 1,6 million malicious code threats reported in 2008. These threats are often in the form of viruses, worms or trojans. Some virus families consist of several thousands or even tens of thousand distinct viruses that were created by virus kits, a specialized software aimed at automated virus creation [2].

Best current practices for malicious code detection are based on elaborate manual malicious code analysis [2]. Even though a manual code analysis can produce a signature for an efficient detection, with the increasing number and sophistication of malicious code threats, it becomes more and more challenging to produce them. Additionally, this approach fails to address the problem from a broader perspective that could lead to an increased survivability of information systems.

In the following, we introduce the locality sensitive hashing as an underlying mechanisms in the supervised k-nearest neighbor classification algorithm. Running experiments with several (benchmark) data sets, we argue about the efficiency and feasibility of such a classification. In investigating this approach, our motivation is based on the fact, that with the increasing complexity and sophistication of network intrusions, the classical approach based on signature detection will become in the future more and more challenging, if not impossible. We evaluate the efficiency of this approach in terms of detection rate and false positives rate.

This documents is organized as follows. In Sec. II we introduce the locality sensitive hashing. Additionally, we introduce an approximate k-nearest neighbors classifier. In Sec. III we discuss the benchmark data sets that we used. In Sec. IV we introduce our experimental setup. In Sec. V we report our experimental results. In Sec. VI we compare the performance of our approach with the KDD'99 winner. In Sec. VII we discuss the related work. In Sec. VIII we conclude.

II. Locality Sensitive Hashing

Datar et al. [3] recently proposed a locality sensitive hashing scheme for finding an approximate nearest neighbor in high dimensional space. The goal of the (R, c)-near neighbor problem is to report a point within the distance cR from a query q, if there is a point in the data set P within distance R from q. A key technique in solving the (R, c)-near neighbor problem is locality sensitive hashing.

Definition 1: A family of hash functions $\mathcal{H} = h : S \to U$ is called $(R, cR, p1, p2)$-sensitive, if for any two points $v, q \in \Re^d$, where $\Re$ is the set of real numbers and d is the dimensionality

- if $d(v - q) \leq R$ then $Pr[h(q) = h(p)] \geq p_1$,
- if $d(v - q) \geq cR$ then $Pr[h(q) = h(p)] \leq p_2$.

$d(v - q)$ is the distance of points v and q under l^p norm. In order for a locality-sensitive hashing (LSH) family to be useful, it has to satisfy $p_1 > p_2$.

The algorithm proceeds as follows. Define a function family $\mathcal{G} = \{g : S \to U^k\}$ such that $g(v) = (h_1(v), ..., h_k(v))$. For an integer L, choose independently and uniformly at random L functions $g_1, ..., g_L$ from $\mathcal{G}$. During preprocessing, store each $v \in P$ in the bucket $g_1(v), ..., g_L(v)$. To process a query q, search all buckets $g_1(q), ..., g_L(q)$. If any of the points found in these buckets lies within distance R from q, return Yes and v_j, else return No.

Parameters k and L are chosen so that with some constant probability it holds:

- if there exists v such that $d(v, q) \leq R$ then $g_j(v) = g_j(q)$ for some $j = 1, ..., L$, and
- the total number of collisions of q with points v such that $d(v, q) \leq cR$ is less than $3L$.

Then $k = log_{1/p_2}\ n$ and $L = n^\rho$, $\rho = \frac{ln\ 1/p1}{ln\ 1/p2}$, where $n = |P|$. The results in [3] show that for the norm l_2, the parameter $\rho = 0.1 - 0.2$ for the approximation factor c being in the range $4 - 8$.

In [3] the following locality sensitive family is proposed.

Definition 2: For a fixed a and b, the hash function is $h_{a,b}(v) = \frac{a.v+b}{r}$, where a is a d-dimensional vector with entries chosen independently from a p-stable distribution and b is a real number chosen uniformly from the range $[0, r]$. The $(.)$ operator is the vector dot product operator. Finally, r is a parameter that dissects the real line into segments of equal size.

Definition 3: A distribution $\mathcal{D}$ over $\mathcal{R}$ is called p-stable, if there exists $p \geq 0$ such that for any n real numbers $x_1, ..., x_n$ and i.i.d. variables $X_1, ..., X_n$ with distribution $\mathcal{D}$, the random variable $\sum_i x_i X_i$ has the same distribution as the variable $(\sum_i |x_i|^p)^{-p} X$, where X is a random variable with distribution $\mathcal{D}$. Examples of a p-stable distribution are the Cauchy distribution or the Gaussian distribution.

We used the algorithm implementation called E2LSH introduced in [4]. This implementation takes advantage of several optimization geared towards "average case data sets". Most notably, it benefits from the fact that the functions $g_1, ..., g_L$ are not independent. The query time, if decomposed into two terms is $T_g = O(dk\sqrt{L})$ and $T_c = O(d.\#collisions)$. T_g is the cost of computing L functions g_j and T_c is the cost of computing the mutual distances between the query q and all points ($\#collisions$) found in the buckets $g_1(q), ..., g_L(q)$. The algorithm requires $O(nL)$ memory.

We use the above described algorithm for (approximate) k-nearest neighbors classification. In the k-nearest neighbors classification algorithm [5], a test sample is classified after its k-nearest neighbors by majority rule. This means, the k-nearest neighbors of a test sample are computed and the class label of the majority of the computed neighbors is assigned to the test sample. In our approach, we consider two cases: (i) the test sample is classified after its approximate nearest neighbor and (ii) the test sample is classified after two approximate nearest neighbor. In the latter case it is necessary that both of the two approximate nearest neighbors share the same label. If this is not the case, we mark the test sample as "undecided". This approach was introduced in order to control the false positives rate.

III. The Datasets

We tested our approach on three data sets. We must note that suitable data sets having (i) high dimensionality, (ii) a high number of samples and (iii) relevance to intrusion detection are scarce. Additionally, to facilitate the whole process, *labeled* data sets are preferable to make the experiments straightforward. In a labeled data set, the labels of samples are determined by whether a given sample represents normal behavior or an intrusion.

The first data set is the KDD'99 data set used in a data classification challenge within the "Fifth International Conference on Knowledge Discovery and Data Mining" [6]. The second one is a Yahoo! web spam data set. The third one, Madelon, is an artificial dataset, which was part of the NIPS 2003 (Neural Information Processing Systems) feature selection challenge.

The KDD dataset consists of almost five million labeled data in 22 attacks and a one normal label. The data set is divided into five basic classes according to the attack type:

- normal,
- probe - surveillance and other probing,
- denial of service (DOS),
- user-to-root (U2R) - unauthorized access with superuser (root) privileges
- remote-to-local (R2L) - unauthorized access from a remote machine.

Each data item has 41 attributes. It is based on real data traffic measurements in a local area network. This data set was then extended with samples that represent attacks from one of the above four classes. There were 22 different attack types.

The goal of the competition was to come up with a good classifier that can distinguish between (i) normal traffic and attacks (good/bad classification), (ii) normal traffic and the four attack classes, and (iii) normal traffic and a specific attack type. The data set was split into two subsets: training and test sets.

- The *training data set* was available to the participants long before the challenge.
- The *test data set* was then used to test a classifier computed with the training set. In order to make the challenge more realistic, a few attacks not present in the training set were included in the test set.

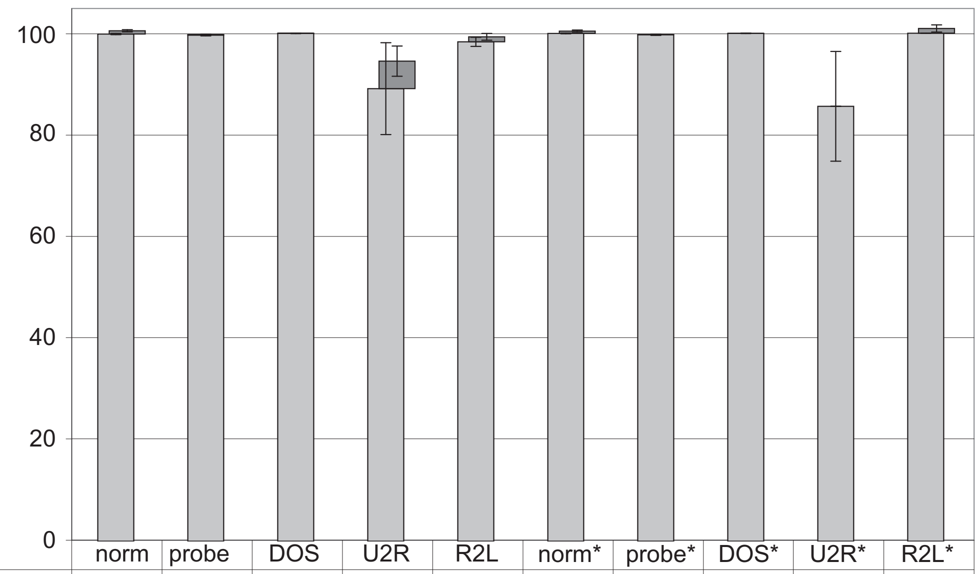
The Yahoo! WEBSPAM-UK2007 collection [7] is based on a crawl of the .uk domain done in May 2007, and labeled by a group of volunteers. The collection includes 114,529 hosts out of which 6,479 are labeled. The Web spam datasets are provided to advance research on Web spam detection. This data set is also provided in precomputed form with 138 attributes. This dataset has only 3 labels: spam, nospam and undecided.

The third data set, Madelon [8], represents a two-class classification problem with continuous input variables. The problem is multivariate and highly non-linear. It contains data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the 2 classes (corresponding to the +-1 labels). A number of distractor features called 'probes' having no predictive power was added. The order of the features and patterns were randomized. This dataset contains 4,400 items, but only 2,600 labeled. The number of attributes is 500.

IV. Experimental Setup

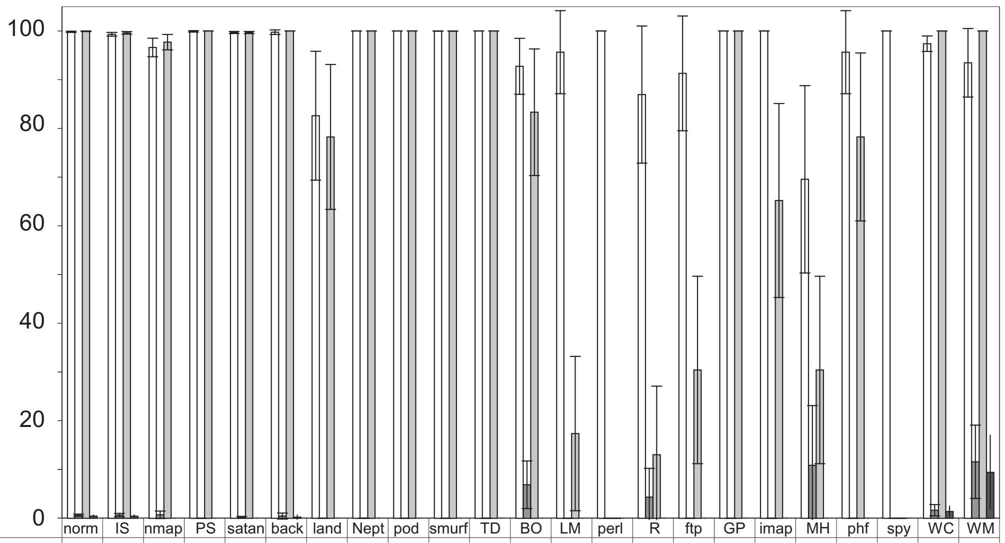
Our experiments were done in three phases:

1) Assigning numerical values to nominal features. Data preprocessing for the E2LSH application.
2) Classification with E2LSH.
3) Statistical evaluation of the results.



	norm	probe	DOS	U2R	R2L	norm*	probe*	DOS*	U2R*	R2L*
■ FP rate	0.639	0.098	0.025	5.411	0.986	0.457	0.020	0.008	0	0.934
□ Det rate	99.84	99.63	99.96	89.09	98.30	99.95	99.70	99.97	85.61	100.00
UN rate	0.265	1.053	0.183	0.758	0.505	0.457	2.965	0.309	61.364	15.993

Fig. 1. KDD data set: detection rate and FP rate for the normal and 4 intrusion classes.



	norm	IS	nmap	PS	satan	back	land	Nept	pod	smurf	TD	BO	LM	perl	R	ftp	GP	imap	MH	phf	spy	WC	WM
□ Det rate	99.84	99.32	96.62	99.92	99.69	99.76	82.61	100	100	99.99	100	92.75	95.65	100	86.96	91.30	100	100	69.57	95.65	100	97.39	93.48
■ FP rate	0.639	0.670	0.712	0	0.202	0.458	0	0.022	0	0	0	6.884	0	0	4.348	0	0	0	10.87	0	0	1.649	11.59
□ Det rate*	99.95	99.62	97.71	100	99.69	100	78.26	100	100	99.99	100	83.33	17.39	0	13.04	30.43	100	65.22	30.43	78.26	0	100	100
■ FP rate*	0.457	0.432	0	0	0	0.229	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.395	9.420

Fig. 2. KDD data set: detection rate and FP rate for the normal and 22 intrusion types. {ipsweep, nmap, portsweep, satan} belong to the probe intrusion class. {back, land, neptune, pod, smurf, teardrop} belong to the DOS intrusion class. {buffer overflow, loadmodule, perl, rootkit} belong to the U2R intrusion class. {ftp write, guess password, imap, multihop, phf, spy, warezclient, warezmaster} belong to the R2L intrusion class.

A. Data Preprocessing

The nominal features in any of the data sets were converted to a numerical representation. This was done by parsing the corresponding file and assigning monotonically increasing integer values to nominal features. All numerical features were subsequently normalized into the [0.0,1.0] range. Then the data format was converted to the E2LSH format. If necessary, we filtered out features that do not change in range, i.e. always take upon the same value.

B. Classification

As we already mentioned, our classification approach is based on the k-nearest neighbors classification with $k = 1$. We extended this approach by requiring a double vote during the classification process, i.e. a feature vector would be assigned

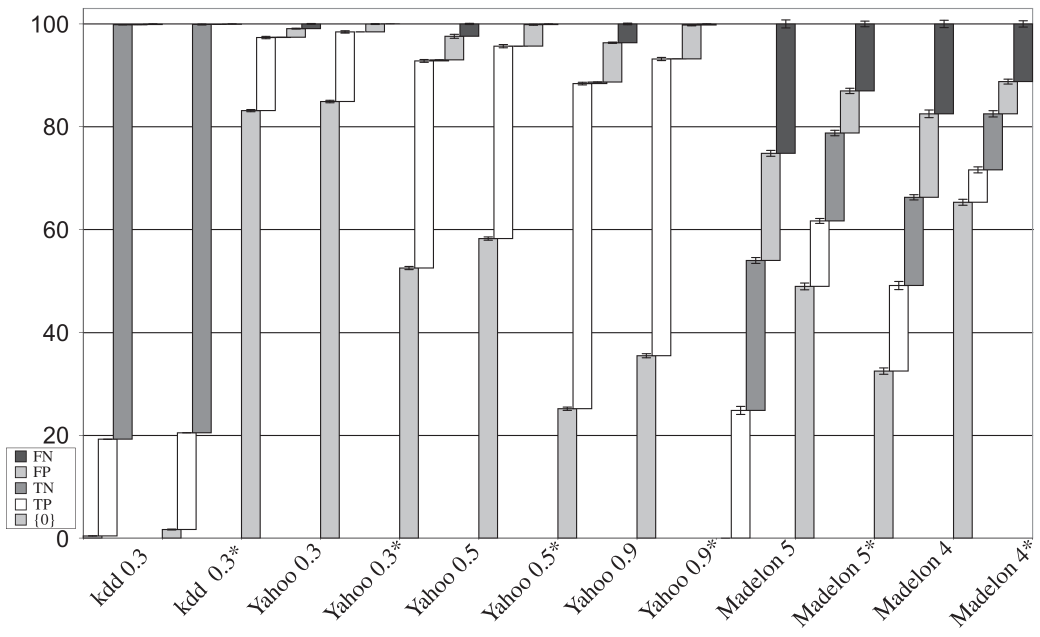


Fig. 3. Comparison of results for the KDD, Yahoo and Madelon data sets.

a label, if both of its two nearest neighbors share the same label.

For computing the approximate nearest neighbor the E2LSH tool is used. This tool automatically computes the most suitable values for the k and L parameters. This is done by minimizing the sum $T_g + T_c$. The (dissection) parameter r was left at its default value 4. The approximation parameter c was set to 1.1. Upon success, for a query q the tool outputs the list of approximate nearest neighbors. The Cauchy distribution was used as a p-stable distribution. We only considered the l^2 norm, i.e. the Euclidean norm.

C. Data Evaluation

The classification procedure was evaluated in terms of the detection rate, the false positives rate (FP rate) and the undecided rate (UN rate). The UN rate reflects the fact that not all vectors could be predicted because i) there was no neighbor within the distance R from the query q or ii) the prediction based on the two nearest neighbors would not coincide.

Definition 4: $Det\ rate = \frac{c_{c_j}}{n_{c_j}} \times 100.0\%$

Definition 5: $FP\ rate = \frac{FP_{c_j}}{FP_{c_j} + c_{c_j}} \times 100.0\%$

Definition 6: $UN\ rate = \frac{UN_{c_j}}{n_{c_j}} \times 100.0\%$

where c_j is a class label. n_{c_j} is the number of vectors (samples) labeled with the class c_j. c_{c_j} is the number of vectors that are correctly predicted by the induction algorithm (k-nearest neighbors algorithm) as belonging to the class c_j. FP_{c_j} is the number of samples incorrectly predicted as belonging to c_j. 95% confidence intervals ($CI_{95\%}$) were computed for each measure. UN_{c_j} is the number of "undecided" vectors belonging to the class c_j. When comparing performance for the three different data sets KDD'99, Yahoo and Madelon, we report directly in terms of false positives (FP), true positives (TP), false negatives (FN) and true negatives (TN).

For performance estimation, we used stratified holdout validation [9]. A holdout validation depends on the division of the data set into a training set and a test set. In random subsampling, the holdout method is repeated k times, with $k = 20$ in our experiments. The distribution of classes was in both the training and the test sets preserved. The detection rate, the FP rate and their confidence intervals are then computed.

Now we describe how the holdout validation was applied to the various data sets. The size of the KDD data set was $494,021$ (we used a subset of the KDD data set). From this data set $1,000$ vectors were randomly chosen to belong to the test data set. The other two sets, Yahoo and Madelon, although, having a large dimensionality were small in size (low number of samples). Their sizes were $6,479$ and $2,600$, respectively. For the Yahoo data set, the training and test set sizes were $4,275$ and $2,204$, respectively. The Madelon data set was divided into training and test data sets of the size $2,000$ data and 600, respectively.

The training data sets constitute in the case of the k-nearest neighbors classification, the data set that is stored in a database. The test data set constitutes the set of queries.

In case of the KDD data set, we did three types of predictions:

- Good vs bad, i.e. we predicted whether a given sample represents a normal behavior or an intrusion.
- Good vs intrusion categories, i.e. we predicted whether a given sample represents a normal behavior or an intrusion. If it represents an intrusion, the intrusion class

membership was also output. There were 4 different intrusion categories (see Section III).

- Good vs intrusion types. The same as above but we attempted to predict the exact intrusion type. There were 22 different intrusion types (see Section III).

V. Performance Evaluation

The results are reported in Figures 1, 2 and 3. The results based on two approximate nearest neighbors are appended with a $*$ sign. Under the label "$\{0\}$" in Fig. 3 are shown the "undecided" cases for the two nearest neighbor classification. The 95% confidence intervals are depicted as error bars. The results in Figures 1, 2 and 3 are based on the *training* KDD data subset.

The Fig. 1 depicts the results for the KDD data set for both classification by the nearest neighbors and the two nearest neighbors. The detection rate in the former case is in the range $98-99\%$ with the exception of the U2R class with the detection rate of 89% and the FP rate of 5.4%. Using the two nearest neighbors procedure the FP rate is at the same level (there is no statistically significant difference), except the U2R class. In this case the FP rate is decreased to 0%.

Notice that the FP rate in Fig. 1 also decreases because many samples fall into the "undecided" category. It can be seen that 61% instances of U2R* and 16% instances of R2L* are undecided. The is also the reason why the detection rate decreases for some classes; there are many samples that were marked as "undecided". In the case of the U2R* class, the detection performance, if compared to U2R, decreases by about 3.5%.

The Fig. 2 depicts the results for the 22 intrusion types. The detection rate lies in the range $70-100\%$. The FP rate is in the range $0-13\%$. The extended procedure with the two nearest neighbors had in some cases a statistically significant effect on the FP rate. The intrusion class "spy" has only 2 instances. Since they were both misclassified, the detection rate for this class is 0%.

The Fig. 3 compares the detection performance for the three data sets that we used. The tests were done with several different values for the R parameter. The labels on the x-axis are in the format "datafile + R". It can be stated that for the KDD set, the nearest neighbor classification performed well. For the other two data sets, Yahoo and Madelon, the performance is significantly worse. This is due to the fact that the data sets do not have a high enough number of samples. This can be seen by the rather high number of "undecided" cases. Often there was no neighbor within the distance R from the initial query point.

VI. Comparison with the KDD'99 Winner

Table I shows the confusion matrix of the KDD'99 competition winner [10]. A confusion matrix contains information about actual and predicted classifications done by a classification algorithm. The winning method was based on a set of C5 decision trees and a classification error minimization. Tables III, IV and V show our results using 1-neighbor classification for three different values of the R parameter. The results in the tables are based on the *test* KDD data subset.

Within the KDD test set, the classes contain the following numbers of samples: normal 60,593 (19.48%), probe 4,166 (1.34%), DOS 229,853 (73.9%), U2R 228 (0.07%) and R2L 16,189 (5.2%), together 311,029 labeled samples.

In the KDD competition, the average cost ξ for each confusion matrix was computed. The average cost of the winning entry was 0.2331. The average cost is defined as a sum of the products of confusion matrix and cost matrix (described in [6]) divided by the number of total predicted samples. More formally:

Definition 7: $\xi = \frac{\sum_{i,j} a_{i,j} b_{i,j}}{\sum_{i,j} a_{i,j}}$

where $a_{i,j}$ is the number of samples from the confusion matrix and $b_{i,j}$ is the corresponding entry from the cost matrix; $0 \leq i, j \leq 4$ (there were 4 intrusion classes and one normal class). The average cost computation in conjuction with an approximate nearest neighbor classification is biased due the fact that some samples were excluded, since there was no neighbor within radius R. For example, in the U2R class, there were almost 92% samples excluded (for $R = 0.1$). This points out that only the normal and DOS classes were suitable for classification with the approximate nearest neighbor classifier.

Interestingly, one of the entries in the KDD'99 cup was based on a 1-nearest neighbor classifier [11]. The results are shown in Table II. Our approach performs better, since we only considered neighbors with a radius R from the query. This however precluded classification of some samples.

VII. Related Work

Sabhnani and Serpen [12] compared nine machine learning algorithms with respect to intrusion detection efficiency. They concentrated on the detection of the U2R and R2L intrusion classes of the KDD'99 data set. They concluded that none of the investigated approaches could significantly improve the detection performance with respect to these two intrusion classes. Their detection rate was about 30% for U2R and 10% for R2L.

Bouzida and Gombault [13] presented a method based on Principal component analysis (PCA). The goal of the method is to reduce the complexity of the representation space before a learning algorithm is applied. Their tests are based on the KDD'99 data set. The reported detection rate (using a variant of PCA) for the normal, probe, DOS, U2R and R2L classes were 99.00%, 68.80%, 97.25%, 6.58% and 0.01%, respectively.

Amor et al. [14] investigated the performance of a decision tree classifier and the Bayes classifier with the KDD'99 data set. Their detection rate was about $91-92\%$ in average, when classification with respect to the five classes was done. The worst result was reported for the R2L class, 1% and 9% when using a decision tree and the Naive Bayes classifier, respectively.

The Kernel miner tool based method by I. Levin won the second place in the KDD'99 contest [15]. He constructed a

TABLE I
THE WINNER RESULT OF KDD'99 [6]; $\xi = 0.2331$.

		Predicted [%]				
		normal	probe	DOS	U2R	R2L
Actual [%]	normal	99.45	0.4	0.13	0.01	0.01
	probe	12.27	83.32	4.42	0	0
	DOS	2.31	0.58	97.12	0	0
	U2R	73.68	8.77	0	13.16	4.39
	R2L	89.73	1.82	0	0.05	8.4

TABLE II
THE RESULTS OF THE BEST 1-NEAREST NEIGHBOR CLASSIFIER OF KDD'99; $\xi = 0.2523$.

		Predicted [%]				
		normal	probe	DOS	U2R	R2L
Actual [%]	normal	99.55	0.35	0.09	0	0
	probe	16.73	75.01	8.21	0	0.05
	DOS	2.67	0.03	97.29	0	0
	U2R	91.67	2.19	0.44	3.51	2.19
	R2L	97.5	1.9	0.01	0	0.59

TABLE III
E2LSH RESULT OF KDD'99 WITH $R = 0.1$; $\xi = 0.1767$.

		Predicted [%]					
		normal	probe	DOS	U2R	R2L	UN rate
Actual [%]	normal	99.8	0.13	0.05	0.01	0.01	3.43
	probe	0.44	99.43	0.13	0	0	45.27
	DOS	2.28	0	97.72	0	0	1.45
	U2R	89.47	0	0	5.26	5.26	91.67
	R2L	95.05	0	0.02	0.08	4.86	31.56

TABLE IV
E2LSH RESULT OF KDD'99 WITH $R = 0.3$; $\xi = 0.2329$.

		Predicted [%]					
		normal	probe	DOS	U2R	R2L	UN rate
Actual [%]	normal	99.72	0.2	0.07	0.01	0.01	0.65
	probe	5.35	93.69	0.95	0	0	34.54
	DOS	2.36	0	97.64	0	0	1.12
	U2R	76.19	2.38	0	11.9	9.52	81.58
	R2L	95.45	0.01	0.01	0.27	4.26	3.02

TABLE V
E2LSH RESULT OF KDD'99 WITH $R = 0.5$; $\xi = 0.2377$.

		Predicted [%]					
		normal	probe	DOS	U2R	R2L	UN rate
Actual [%]	normal	99.63	0.25	0.1	0.01	0.01	0.27
	probe	9.78	88.3	1.92	0	0	28.8
	DOS	2.44	0.01	97.55	0	0.01	0.85
	U2R	48.87	39.1	0	9.02	3.01	41.67
	R2L	95.27	0.08	0.01	0.27	4.37	1.52

set of locally optimal decision tree from which a subset was used in class prediction. The average cost of his solution was $\xi = 0.2356$.

Miheev et al. [16] won the third place in the KDD'99 competition. Their solution was based on combining expert knowledge with voting decision trees. The average cost of their solution was $\xi = 0.2367$.

Depren et al. proposed a hybrid Intrusion Detection System architecture [17] with several modules: an anomaly detection module, a misuse detection module and a decision support system. They used the J.48 decision tree algorithm to classify the attacks. The KDD'99 data set was used for performance estimation. In their experiments, they achieved the detection rate of $98.96 - 99.90\%$ and a false positives rate of $0.1 - 1.01\%$. They classified with respect to all intrusion types. They however only used a 10% subset of the KDD'99 data set with 199,677 samples.

Kim and Park [18] applied Support Vector Machines (SVM) for classifying the intrusion classes of the KDD'99 data set. For the normal and the four intrusion classes they reported the following results in terms of detection rate. Normal class 99.3%, probe 36.65%, DOS 91.6%, U2R 12% and R2L 22%.

Shyu et al. [19] used a principal component classifier for classification of the KDD'99 data set. For the normal and intrusion classes, they reported the detection rate of 99.08% and 98.94%, respectively. The FP rate was about 0.5%.

Song et al. presented an approach based on Genetic Programming in [20]. The detection rate that they reported, with respect to the four intrusion classes, was 95.6%. Adjusting various parameters, they were able to decrease the FP rate, however, at the cost of a lower detection rate.

The results reported in this and the previous section are summarized in Tab. VI. The FP rate reported is the FP rate of the "any intrusion" class, where the four intrusion classes are merged together. Occasionally, we cannot report some measures, since they were not computed by the authors. In some cases, the authors used in their experiments various input parameters in order to control the detection performance. We chose to include into Tab. VI the results that offer a balanced detection performance with respect to the detection and FP rates.

VIII. CONCLUSIONS

We evaluated a novel approach to intrusion detection in high dimensional space. This approach is based on locality sensitive hashing, a recent approach for approximate nearest neighbor computation due to Datar et al. [3]. We tested this approach on three different data sets. Most notably, we used the KDD'99 data set that is a standard benchmark for testing intrusion detection algorithms.

Our results show solid detection performance that is comparable to the KDD'99 competition winner, if the three classes with the highest number of samples (normal, probe, DOS) are considered. Since the applied approximate approach only outputs the nearest neighbor within the distance R from the query point, some samples stayed unclassified (undecided). The undecided rate was in the range $1 - 4\%$. However, for several specific intrusion classes, this rate was rather high. For example, for the U2R intrusion class and $R = 0.1$, the undecided rate was about 92%. This points out that the task of correct classification is for some samples in the test KDD data set rather tricky.

In general, as it could be seen on the Yahoo and Madelon

TABLE VI
COMPARISON OF KDD'99 BASED CLASSIFICATION.

	Det. rate [%]					FP rate [%]	Note
	normal	probe	DOS	U2R	R2L		
KDD'99 Winner	99.45	83.3	97.1	13.2	8.4	25.38	B. Pfahringer [10]
KDD'99 2nd place	99.42	84.5	97.4	11.8	7.3	26.05	I. Levin [15]
KDD'99 with 1-nearest neigh.	99.55	75.0	97.3	3.5	0.6	27.46	Ch. Elkan [11]
Naive Bayes	97.68	88.3	96.7	11.0	8.7	26.48	Amor et al. [14]
C4.5 on PCA7 projection	99.39	75.8	97.2	5.7	0.1	27.72	Bouzida and Gombault [13]
C4.5 on PCA2 projection	99.00	66.8	97.3	6.6	0.0	27.88	Bouzida and Gombault [13]
SVM IDS	99.30	36.7	91.6	12.0	22.0	–	Kim and Park [18]
GP 8-basic 90-difficulty 10-age	99.70	48.5	95.6	10.1	0.2	–	Song et al. [20]
Hybrid IDS	99.82	99.90				0.70	Depren et al. [17], used only *training* subset
PCC 1%	99.08	98.94				0.46	Shyu et al. [19], used only *training* subset
E2LSH with R=0.3 (training set)	99.84	99.6	99.9	93.9	98.4	0.04	0.42% samples undecided, used only *training* subset
E2LSH with R=0.1	99.80	93.4	97.7	5.3	4.9	21.22	4.06% samples stayed undecided
E2LSH with R=0.3	99.72	93.7	97.6	11.9	4.3	25.47	1.63% samples stayed undecided
E2LSH with R=0.5	99.63	88.3	97.6	9.0	4.4	25.96	1.18% samples stayed undecided

data sets, the investigated type of classification is less suitable for data sets or classes with a low number of samples.

Additionally to 1-nearest neighbor classification, we also evaluated classification based on 2-nearest neighbors. The goal was to gain some insight on the robustness of the underlying approximate nearest neighbor classification with respect to the FP rate. We observed an improvement in the FP rate control, however only for the U2R class.

We conclude that even though, the results based on the approximate approach are not significantly better than that of the KDD'99 winner (based on the C5 decision tree), the computational complexity is in our case notably decreased thanks to the applied locality sensitive hashing. Our approach could be thus useful for classification of high rate data traffic such as the Internet traffic.

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) under the grant no. SZ 51/24-2 (Survivable Ad Hoc Networks – SANE). This work was done while Stanislav Marcek was visiting the Simulation and modeling group at the Leibniz University of Hannover. Stanislav Marcek acknowledges the support from the grants AV 4/0103/06, APVV-0618-07 and VEGA 1/0872/08.

REFERENCES

[1] Symantec, *Global Internet Security Threat Report*, vol. 14, 2009.
[2] P. Szor, *The art of computer virus research and defense*. Addison-Wesley Professional, 2005.
[3] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM New York, NY, USA, 2004, pp. 253–262.
[4] A. Andoni and P. Indyk, *E2lsh 0.1 user manual*, 2005. [Online]. Available: http://web.mit.edu/andoni/www/LSH/index.html
[5] E. Alpaydin, *Introduction To Machine Learning*. MIT Press, 2004.
[6] "Kdd-cup 1999," 1999. [Online]. Available: http://www.sigkdd.org/kddcup
[7] "Yahoo! webspam-uk2007," 2008. [Online]. Available: http://barcelona.research.yahoo.net/webspam/datasets/uk2007/
[8] "Machine learning repository," 2008. [Online]. Available: http://mlr.cs.umass.edu/ml/datasets.html
[9] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International Joint Conference on Artificial Intelligence*, vol. 14, 1995, pp. 1137–1145.
[10] B. Pfahringer, "Winning the KDD99 classification cup: Bagged boosting," *ACM SIGKDD Explorations Newsletter*, vol. 1, no. 2, pp. 65–66, 2000.
[11] C. Elkan, "Results of the KDD'99 classifier learning," *ACM SIGKDD Explorations Newsletter*, vol. 1, no. 2, pp. 63–64, 2000.
[12] M. Sabhnani and G. Serpen, "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context," in *Proceedings of International Conference on Machine Learning: Models, Technologies, and Applications*, vol. 1. Las Vegas, Nevada, USA: Springer-Verlag, January 2003, pp. 2009–215.
[13] Y. Bouzida and S. Gombault, "Eigenconnections to Intrusion Detection," in *In 19 th IFIP International Information Security Conference (SEC2004*. Kluwer Academic Publishers, 2004, pp. 241–258.
[14] N. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," *Proc. of the 2004 ACM symposium on Applied computing*, pp. 420–424, 2004.
[15] I. Levin, "KDD-99 classifier learning contest LLSoft's results overview," *ACM SIGKDD Explorations Newsletter*, vol. 1, no. 2, pp. 67–75, 2000.
[16] V. Miheev, A. Vopilov, and I. Shabalin, "The MP13 approach to the KDD'99 classifier learning contest," *SIGKDD Explor. Newsl.*, vol. 1, no. 2, pp. 76–77, 2000.
[17] O. Depren, M. Topallar, E. Anarim, and M. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," vol. 29, no. 4, pp. 713–722, 2005.
[18] D. Kim and J. Park, "Network-Based Intrusion Detection with Support Vector Machines," vol. 2662/2003, pp. 747–756, 2003.
[19] M. Shyu, S. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM03)*, 2003, pp. 172–179.
[20] D. Song, M. Heywood, and A. Zincir-Heywood, "Training genetic programming on half a million patterns: an example from anomaly detection," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 225–239, 2005.