Naive Bayesian Networks in Intrusion Detection Systems

Nahla Ben Amor[1], Salem Benferhat[2] and Zied Elouedi[1]

[1] Institut Supérieur de Gestion Tunis, 41 Av. de la liberté, 2000 Le Bardo, Tunisie
nahla.benamor@gmx.fr, zied.elouedi@gmx.fr
[2] CRIL - CNRS, Univ. d'Artois, Rue Jean Souvraz SP 18 62307 Lens, Cedex, France
benferhat@cril.univ-artois.fr

Abstract. In this paper, we present and analyze an approach to intrusion detection using *naive Bayes network*. We have used a set of benchmark data from KDD'99 which are appropriate to evaluate an intrusion detection system. We consider three levels of attack granularities depending on whether dealing with whole attacks, or grouping them in four main categories or just focusing on normal and abnormal behaviours. Moreover, we consider different ways for handling continuous variables. The different experimental results show that naive Bayesian networks, even having a simple structure, can provide efficient and accurate classifiers to detect intrusions.

keywords: Intrusion detection, Naive Bayesian networks, Learning

1 Introduction

Intrusion detection in the context of information systems is regarded as a set of attempts to compromise a computer network resource security. There are two general approaches to intrusion detection [1]:

- *Anomaly detection:* based on the detection of an anomaly in a user behavior. The idea is that each user has a certain profile within the system that will not be changed a lot in time. Then, this profile is expected to be 'normal' and consequently any significant deviation will be considered as an anomaly. Several Intrusion Detection Systems based on this approach are developed like IDES [12], NIDES [11].
- *Misuse detection:* also named signature detection since in this case any intrusion can be described by its signature characterized by the values of its features. Systems based on this approach use different models like rule based expert system e.g. NIDES [11], or state transition analysis e.g. STAT [6], or a more formal pattern classification e.g. IDIOT [9].

Recently, Valdes [15] has performed a new approach to intrusion detection based on Bayesian networks. Bayesian networks [7, 13] are tools to reason with uncertain information in the probability theory framework. They use direct acyclic graphs to represent causal relations, and conditional probabilities (of each node given its parents) to express uncertainty of causal relations. Valdes [15] uses a simple form of Bayesian networks, called naive Bayesian networks, composed of two levels: one root node which represents a session class (normal and different kinds of attacks), and several leaf nodes, each of them contain a feature of a connection. This method goes by the name of naive Bayes, because it *naively* assumes a strong independence relation: features are independent in the context of a session class. Such assumption is not always true, and may have a negative influence on the inferred results. Naive Bayesian networks have several advantages due to their simple structure. In particular, the construction of naive Bayesian networks is very simple, and the inference (classification) is achieved in a linear time (while the inference in Bayesian networks with a general structure is known to be NP-complete [3]. Moreover, the construction of naive Bayesian networks is incremental, in the sense that it can be easily updated (namely, it is always easy to consider and take into account new cases in hand).

The aim of this paper is to provide experimental results showing that naive Bayesian networks, with their simple structure and their strong independence assumptions, can be very competitive. Experimental results obtained in this paper use KDD'99 [8] intrusion data sets. There exist several recent works using these data [14, 15]. However, our experimental results give a new perspective. In particular different levels of classifications results are considered. Indeed, as we will see there are different ways to use naive Bayesian networks. These different ways are explained by the fact that on one hand there are several attack granularities, and on the other hand there are several possibilities for handling continuous variables.

In this paper, experimentations are performed according to three levels of attack granularities depending on whether dealing with whole attacks, or grouping them in four main categories or just focusing on normal and abnormal behaviours. Moreover for continuous variables we will analyze two cases: Gaussian distributions and kernel density estimation. Lastly, some results based on discretizing continuous domains of some features have been achieved. The immediate way of using Bayesian networks is to consider all attacks (namely, without any grouping of elementary attacks), and Gaussian distributions (this what usually is used for continuous variables). As we will see, experimental results show that this immediate way of using naive Bayesian networks is not satisfactory. For instance, replacing Gaussian distributions by kernel density estimation leads to a significant increasing of PCC (Percent of Correct Classification).

The rest of the paper is organized as follows: Section 2 gives a brief refresher on naive Bayesian networks. Section 3 first presents KDD'99 dataset, then gives the general schema of the experimental studies. Sections 4, 5 and 6 present experimental results focusing respectively on the whole attacks, four categories of attacks and normal/abnormal behaviours. Section 7 presents the positive effect of discritizing some variables. Lastly, related works and concluding discussions are reported in Section 8.

2 Naive Bayesian networks

Bayesian networks are one of the most widely used graphical models to represent and handle uncertain information [7, 13]. Bayesian networks are specified by two components:
- A *graphical component* composed of a directed acyclic graph (DAG) where vertices represent events and edges are relations between events.
- A *numerical component* consisting in a quantification of different links in the DAG by a *conditional* probability distribution of each node in the context of its parents.

Naive Bayesian networks [10] are very simple Bayesian networks which are composed of DAGs with only one parent, representing the *unobserved* node, and several children, corresponding to *observed* nodes, with the strong assumption of independence among child nodes in the context of their parent.

Naive Bayesian networks are appropriate to deal with classification problems [5]. In fact, classification is ensured by considering the parent node to be a hidden variable stating to which class each object in the database should belong and child nodes represent different attributes specifying this object.

Hence, in presence of a training set we should only compute the conditional probabilities since the structure is unique. This computation can be summarized as follows:
- conditional probabilities for *discrete* attributes probabilities are computed from frequencies by counting how many times each attribute-value pair occurs with each value of the parent node.
- *continuous* attributes are usually handled by assuming that they have a

Gaussian (i.e. *normal*) probability distribution which supposes a continued values of attributes. Thus, for each class value c_i and each continuous attribute A_k, we should compute the mean μ and the standard deviation σ. Using these two values we can compute a *probability density function* for any value a_k of A_k.

The Gaussian-distribution assumption for numeric attributes can be considered as a restriction of naive Bayesian networks since some attributes are not normally distributed.

Thus we can use a generalization of this method by using the *kernel density estimation* [4] which is a non-parametric density estimates for classification. This estimates does not assume any particular distribution for the attribute values and it is based on localizing for each target point a_k the observations close to it via a weighting function or kernel $K_\sigma(a_k, a_i)$ which assigns a weight to each a_i based on its distance from a_k. The more popular choice for K_σ is the Gaussian kernel $K_\sigma = \phi(\mid a_i - a_k \mid / \sigma)$ where σ is the standard deviation. Another alternative for continuous variables is to simply discretize them.

Once the network is quantified, it is able to classify any new object giving its attributes' values using the Baye's rule expressed by:

$$P(c_i \mid A) = \frac{P(A \mid c_i) \cdot P(c_i)}{P(A)} \tag{1}$$

where c_i is a possible value in the session class and A is the total evidence on attributes nodes. The evidence A can be dispatched into pieces of evidence, say $a_1, a_2, ..., a_n$ relative to the attributes $A_1, A_2, ..., A_n$, respectively. Since naive Bayesian networks work under the assumption that these attributes are independent (giving the parent node C), their combined probability is obtained as follows:

$$P(c_i \mid A) = \frac{P(a_1 \mid c_i) \cdot P(a_2 \mid c_i) \cdot ... \cdot P(a_n \mid c_i) \cdot P(c_i)}{P(A)} \tag{2}$$

Note that there is no need to explicitly compute the denominator $P(A)$ since it is determined by the normalization condition. Therefore, it is sufficient to compute for each class c_i its likelihood, i.e. $P(a_1 \mid c_i) \cdot P(a_2 \mid c_i) \cdot ... \cdot P(a_n \mid c_i) \cdot P(c_i)$ to classify any new object characterized by its attributes' values $a_1, a_2, ..., a_n$.

3 Experimental data

3.1 Description of KDD'99 data set

The data used in this paper are those proposed in the KDD'99 for intrusion detection [8] which are generally used for benchmarking intrusion detection problems. They set up an environment to collect TCP/IP dump raws from a host located on a simulated military network. Each TCP/IP connection is described by 41 discrete and continuous features and labeled as either normal, or as an attack, with exactly one specific attack type. Attacks fall into four main categories:

- *Denial of Service Attacks (DOS)*
- *User to Root Attacks (U2R)*
- *Remote to User Attacks (R2L)*
- *Probing*

3.2 Different experimental study cases

We handle 10% of the whole KDD'99 dataset, corresponding to 494019 training connections, and 311029 testing connections.

Note that in the training set the number of U2R instances is very small (0.01%). In order to guarantee their minimal learning, we have duplicated them until having 0.22% training instances. This has a very small influence on U2R instances, and no influence on other instances.

The strategy behind different experimentations presented in this paper is based on the following points:

- Three levels of attack granularities: we can focalize on three cases relative to different attacks in order to handle :

- *Whole-attacks*: all attack classes presented by KDD dataset in addition to the normal situation.
- *Five-classes*: the four attack categories (i.e. DOS, R2L, U2R, Probing). Note that there are 19.65% (resp. 79.07%, 0.23%, 0.22%, 0.83%) of normal (resp. DOS, R2L, U2R,Probing) training connections and 19.48% (resp. 73.90%, 5.21%, 0.07%, 1.34%) of normal (resp. DOS, R2L, U2R, Probing) testing connections.
- *Two-classes*: i.e. Normal and Abnormal by grouping all attacks in the same class (i.e. Abnormal).

- Gathering attacks: in the five-class and two-class cases, there are two strategies to gather results either before or after classification:

- *Gathering before classification*: the idea is to slightly modify the dataset by grouping attacks belonging to the same attack category (i.e. DOS, R2L, U2R or Probing) or by grouping them in a unique class i.e. abnormal.

– *Gathering after classification*:
  - For the five classes, the training set remains unchanged. However, each connection classified into one of the 38 attacks is assigned to the one of the four categories it belongs to.
  - For the two classes, there are two strategies: either we do not modify the training set and each connection classified into one of the 38 attacks is simply labeled as abnormal, or we first modify the training set by gathering attacks into four categories, then each connection classified in one of these categories will be labeled as abnormal.

- Discretization: some attributes, characterizing connections, are obviously discrete such as the protocol type, others are obviously continuous such as the duration [8]. This is not the case for all continuous attributes since, in some cases, the type is ambiguous. Typically, some attributes are labeled as continuous but, indeed, take a finite number of values, e.g, percent of connections to the same service having "SYN" errors. Thus, the idea is to discretize such attributes in order to test its incidence on the classification results.

In each of the studied cases, the evaluation of classification efficiency is based on the *Percent of Correct Classification* (PCC) of the instances belonging to the testing. Besides, we will use a particular form of the PCC relative to each class, named the *recall* criterion which indicates the percentage of correctly recognized connections for each class.

4 Focusing on all attacks

Table 1 presents the PCC values of the training and the testing sets according to the whole-class case. It shows that naive Bayesian networks using the kernel Gaussian estimator are completely in accordance with the training set which means that this latter is coherent, i.e., almost all training instances characterized by the same attributes' values belong to the same class. This behaviour is also kept with the classification phase. This is not the case when we use the Gaussian distribution assumption since we have a gap of 7% in the learning phase. This gap becomes larger in the classification phase (about 17%).

5 Focusing on the four categories of attacks

In order to better select the strategy allowing to handle the four major attack categories, we have grouped attacks belonging to the same attack class together. This is done before and after classification. For the latter,

Table 1. PCC's in the whole-attack case

TRAINING SET	TESTING SET
Gaussian distribution	
92.92%	74.38%
Kernel Gaussian estimator	
99.64%	91.13%

we use results of the whole-attack case by summing the number of occurrences relative to each attack category (i.e. DOS, R2L, U2R, Probing). Table 2 gives PCC's relative to these two experimentations.

Table 2. PCC's relative to five classes (values between parentheses are relative to gathering whole-attacks results into five classes after classification)

TRAINING SET	TESTING SET
Gaussian distribution	
92.17% (93.09%)	78.17% (79.29%)
Kernel Gaussian estimator	
98.85% (99.67%)	90.83% (91.40%)

Similarly to the whole-attack case, in both learning and classification phases, naive Bayesian networks based on the kernel estimator present better results than those using the Gaussian distribution. Note that it is slightly better to gather results after classification rather than before it both when using the kernel estimator and the Gaussian distribution.

Table 3. Recall relative to five classes (values between parentheses are relative to gathering whole-attacks results into five classes after classification)

Class	*Gaussian distribution*	*Kernel Gaussian estimator*
Normal (60593)	94.12% (64.15%)	96.97% (98.54%)
DOS (229853)	79.21% (88.43%)	96.25% (96.40%)
R2L (16189)	0.56% (4.40%)	0.02% (0.16%)
U2R (228)	23.68% (11.84%)	1.75% (3.07%)
Probing (4166)	89.73% (90.09%)	60.37% (70.84%)

Table 3 provides the recall criterion (the percentage of correctly classified instances). As it can be seen, in the kernel Gaussian estimations case, for each category, it is better to gather after classification than before classification. This is not the case with Gaussian distributions, since for example normal connections are largely well-classified before classification than after classification.

Indeed, if the gathering is made before classification, the value of 78.17% is due to the misclassification of the instances belonging to the dominant class DOS (only 79.21% of DOS connections are well-classified). Whereas if gathering is made after classification, the value of 79.29% is explained by the misclassification of normal connections which are also important (only 64.15% of them are well-classified). Moreover, Table 3 shows that R2L and U2R connections are always misclassified whenever the assumption used to treat continuous variables. This is due to the fact that the proportions, in the training set, of U2R and R2L attacks are very low (0.22% for U2R and 0.23% for R2L). One idea to make better these results is to more duplicate these instances in the training set.

In fact, within naive Bayesian networks, when a class is presented by a low number of training instances, then it leads to a weak learning regarding this class and consequently to a misclassification of testing connections really belonging to it. Hence, we can have new testing instances really belonging to U2R and R2L attacks, but characterized by attributes' values which deviate from those characterizing these two classes in the training set. These instances are not already learned in the construction phase and their resulting class when applying the inference mechanism are generally wrong.

To illustrate this, a thorough analysis of training connections belonging to U2R shows that the *flag* attribute appears with the value SF while in the majority of testing connections it takes the value REJ which never appears in the learning set with U2R attacks. Thus in the learning phase the conditional probability of REJ in the context of U2R will be equal to zero (i.e. $P(REJ \mid U2R) = 0$). Thus, testing connections pertaining, effectively, to U2R but presenting the value REJ in the *flag* attribute will be missclassified.

Lastly, Table 3 also shows that for classes which are lowly represented in the training set (U2R, R2L, Probe) it is better to use Gaussian distributions instead than kernel Gaussian estimator.

6 Focusing on normal and abnormal connections

In this section, we emphasize on normal behaviour. For this purpose, we have studied PCC's and recall values by focusing on normal connections over the abnormal ones. We first consider the case where we gather all attacks before classification, then the case where gathering is made after classification using results on the whole-class case and those relative to the five-class case. Induced results are summarized in Table 4, and the recall criterion is provided in Table 5.

Table 4. PCC's relative to the normal and abnormal connections (values between parentheses are relative to gathering whole-attacks and five classes results into two classes after classification)

TRAINING SET	TESTING SET
Gaussian distribution	
98.18% (93.20%, 96.63%)	91.52% (84.02%, 92.04%)
Kernel density	
98.75% (99.69%, 98.91%)	91.45% (91.75%, 91.55%)

Table 5. Recall relative to the normal and abnormal classes (values between parentheses are relative to gathering whole-attacks and five classes results into two classes after classification)

Class	*Gaussian distribution*	*Kernel density*
Normal (60593)	97.54% (64.15%, 94.12%)	98.48% (98.54%, 96.97%)
Abnormal (250436)	90.06% (88.83%, 91.54%)	89.75% (90.11%, 90.24%)

Contrary to previous experimentations, Table 4 shows that the gap between naive Bayesian networks based on the kernel Gaussian and those using the Gaussian distribution before gathering is significantly reduced since the PCC is almost the same within the two approaches and presents a very good rate.

Furthermore, all the PCC's for both approaches (the kernel Gaussian and the Gaussian distribution) presented in Table 4 are high, except when using the Gaussian distribution gathering attacks after classification of the five class case. In such a case, the PCC is a little bit worse than the others. This is explained by the low recall of normal connections given in Table 5. Indeed in this case, we use the original dataset (containing the whole attacks) and only 64.15% of normal connections are well-classified which explains the low value of global PCC for naive Bayes networks using the Gaussian distribution in the whole-attack case (i.e. 74.38%). This value reflects a bad learning of normal connections since only 65.69% of them are well-classified in the training set which means that the induced naive Bayes network does not give a *faithful* representation of the normal training connections.

7 Discretization

The idea behind this experimentation is to refine the analysis of continuous attributes in order to handle them in an appropriate manner specific to their values. In particular, we have noticed that there are 15 attributes defined as percentages. These attributes where labeled continuous, but by analyzing the KDD datasets we have noticed that they take at maximum 101 values (from 0.00 to 1.00), thus they can be considered as discrete and their discretization can be directly done by enumerating their domain values.

Since previous experimentations show that results obtained using kernel estimator are largely better than those given when using the Gaussian distribution, this section only gives results of the best strategy, namely when using kernel density estimator. Results induced by this experimentation are summarized in Table 6.

Table 6. PCC's after discretization (values between parentheses are relative to gathering whole-attacks and five classes results into two classes after classification)

WHOLE ATTACKS	FIVE CLASSES	NORMAL AND ABNORMAL
91.20%	91.48% (92.10%)	91.45% (92.69%, 92.40%)

This table shows an increasing (even small) of the PCC in almost all the experimentation cases. Besides, a deep analysis of the recall criterion relative to the five-class case (see Table 7) shows that the recall relative to each of the five categories (i.e. normal, DOS, R2L, U2R, Probing) has increased.

8 Related work and concluding discussions

Table 8 summarizes PCC's relative to the major experimentations performed in this paper.

The major deduced remarks are the followings:

- Table 8 shows that for KDD'99 dataset the standard handling of continuous variables, by means of Gaussian distributions, is not appropriate since it gives worst results.

Table 7. Recall relative to five classes after discretization (values between parentheses are relative to gathering whole-attacks results into five classes after classification)

Normal (60593)	96.64% (97.68%)
DOS (229853)	96.38% (8.66%)
R2L (16189)	7.11% (4.40%)
U2R (228)	11.84% (10.96%)
Probing (4166)	78.18% (88.33%)

Table 8. Summary Table: PCC's on the testing set (values between parentheses are relative to gathering whole-attacks and five classes results into two classes after classification)

WHOLE ATTACKS	FIVE CLASSES	NORMAL AND ABNORMAL
Gaussian distribution		
74.38%	78.17% (79.29%)	91.52% (84.02%, 92.04%)
Kernel Gaussian estimator		
91.13%	90.83% (91.40%)	91.45% (91.40%, 91.73%)
Kernel density with discretization		
91.20%	91.48% (**92.10%**)	91.45% (**91.75%**, 91.55%)

- Table 8 shows that in general, when we focus on five classes and normal/abnormal behaviours, it is better to gather elementary attacks, in their respective classes, after classification step rather than before it.

- According to Table 8, we can see that the best strategies in all experimentations are those using the kernel density estimator and discretization. Moreover, under these strategies, the classification quality is not considerably affected when we deal with all attacks, five classes or only two classes.

- A deep analysis shows that there are some cases where results with naive Bayesian networks are equal or slightly better than those of the winning strategy [8] which is based on a mixture of bagging and boosting decision tree technique. For instance the recalls relative to U2R and Probing connections are better. These interesting results are obtained by applying different strategies such as using the kernel density estimator for handling continuous variables and also discretizing some of them.

Different experimental results, presented in this paper, confirm conclusions obtained by Valdes [15] showing that naive Bayesian networks can well perform in the intrusion detection field and can be competitive with *sophisticated* classification methods such as decision trees [2, 8]. Of course, globally naive Bayesian networks perform a little bit worse than these techniques. However, from a computation point of view, their construction is largely faster. Another line of future research will be to study the effect of limiting the features to those that are relevant on the treatment of continuous variables.

References

1. S. Axelsson. Intrusion detection systems: a survey and taxonomy. In *Technical report 99-15*. March 2000.
2. N. Ben Amor, S. Benferhat, Z. Elouedi, and K. Mellouli. Decision trees and qualitative possibilistic inference: Application to the intrusion detection problem. In *Proceedings of European Conference of Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'2003)*, pages 419–431, Danemark, 2003.
3. G. F. Cooper. Computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.
4. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Hardcover, 2000.
5. N. Friedman and M. Goldszmidt. Building classifiers using bayesian networks. In *Proceedings of American Association for Artificial Intelligence Conference (AAAI'96)*, Portland, Oregon, 1996.
6. K. Ilgun, R. A. Kemmerer., and P. A. Porras. Probability propagation. *IEEE Transactions on Software Engineering*, 21(3):181–199, 1995.
7. F. V. Jensen. *Introduction to Bayesien networks*. UCL Press, University college, London, 1996.
8. KDD. http://kdd.ccs.uci.edu/databases/kddcup99. 1999.
9. S. Kumar and E. H. Spafford. A software architecture to support misuse intrusion detection. In *Proceedings of the 18th National Information Security Conference*, pages 194–204, 1995.
10. P. Langley, W. Iba., and K. Thompson. Decision making using probabilistic inference methods. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence (UAI'92)*, pages 399–406, San Mateo, CA, 1992.
11. T. Lunt. Detecting intruders in computer systems. In *Proceedings of the Sixth Annual Symposium and Technical Displays on Physical and Electronic Security*, 1993.
12. T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Gravey. A real-time intrusion detection expert system (ides). In *Technical report*. Computer Science Laboratory, SRI International, CA, 1992.
13. J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmman, San Francisco (California), 1988.
14. P. Portier and J. Froment-Curtil. Data mining techniques for intrusion detection. In *Technical report, University of Texas at Austin*. 2000.
15. A. Valdes and K. Skinner. Adaptive model-based monitoring for cyber attack detection. In *Proceedings of Recent Advances in Intrusion Detection (RAID 2000)*, pages 80–92, Toulouse, France, 2000.