Ad Hoc Networks xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Intrusion detection in MANET using classification algorithms: The effects of cost and model selection

Aikaterini Mitrokotsa [a,*], Christos Dimitrakakis [b]

[a] Security and Cryptography Laboratory (LASEC), School of Computer and Communication Sciences, EPFL, Station 14, CH-1015 Lausanne, Switzerland
[b] Artificial Intelligence Laboratory (LIA), School of Computer and Communication Sciences, EPFL, Station 14, CH-1015 Lausanne, Switzerland

A R T I C L E I N F O

Article history:
Received 20 November 2011
Received in revised form 9 April 2012
Accepted 16 May 2012
Available online xxxx

Keywords:
Intrusion detection
Classification algorithms
Cost-sensitive classification
Mobile Ad-hoc Networks (MANETs)

A B S T R A C T

Intrusion detection is frequently used as a second line of defense in Mobile Ad-hoc Networks (MANETs). In this paper we examine how to properly use classification methods in intrusion detection for MANETs. In order to do so we evaluate five supervised classification algorithms for intrusion detection on a number of metrics. We measure their performance on a dataset, described in this paper, which includes varied traffic conditions and mobility patterns for multiple attacks. One of our goals is to investigate how classification performance depends on the problem *cost matrix*. Consequently, we examine how the use of uniform versusweighted cost matrices affects classifier performance. A second goal is to examine techniques for tuning classifiers when unknown attack subtypes are expected during testing. Frequently, when classifiers are tuned using cross-validation, data from the same types of attacks are available in all folds. This differs from real-world employment where unknown types of attacks may be present. Consequently, we develop a *sequential* cross-validation procedure so that not all types of attacks will necessarily be present across all folds, in the hope that this would make the tuning of classifiers more robust. Our results indicate that weighted cost matrices can be used effectively with most statistical classifiers and that sequential cross-validation can have a small, but significant effect for certain types of classifiers.

1. Introduction

Mobile Ad hoc Networks (MANETs) present many important advantages and have been employed in a broad range of applications such as emergency services [16], pollution monitoring [6] and vehicular networks [31]. MANETs are dynamic peer-to-peer networks, which employ multi-hop information transfer without requiring an *a priori* infrastructure. Due to their nature, they have unique security requirements. We must guard not only against usual attacks such as denial of service, but also against selfish and malicious nodes more generally. While intrusion prevention can be used as a first line of defence, these types of attacks cannot be prevented directly. In addition, intrusion detection can be used as a mechanism for indicating possible security failures in the system.

A simple way to perform intrusion detection is to use a classifier in order to decide whether some observed traffic data is "normal" or "abnormal". In the simplest case, the classification objective is to minimise the probability of error. However, in problems such as that of intrusion detection, authentication and fraud detection, the goal is not simply to predict the class with highest probability, but to actually take the decision with the lowest expected cost. For example, in intrusion detection, the cost of having an undetected attack is usually much more severe than triggering a false alarm. In *cost-sensitive* classification, decisions are made in order to minimise the expected cost, rather than the probability of error. The concept of

* Corresponding author.
E-mail addresses: katerina.mitrokotsa@epfl.ch (A. Mitrokotsa), christos.dimitrakakis@epfl.ch (C. Dimitrakakis).

1570-8705/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2012.05.006

cost-sensitive classification has been already investigated in wired networks [26].

This paper examines how to properly use classification methods in intrusion detection for MANETs. We perform a comparison of the performance of four well known classifiers. We extend our previous approach [25] in wireless ad hoc networks, where we only investigated simple classification, to cost-sensitive classification, i.e. making classification decisions that minimise the expected cost, rather than the probability of misclassification, and measure its effectiveness for each classifier under consideration. We also address a common problem in intrusion detection applications: the fact that in real world deployment, the data distribution can be very different from that in the training set. This will for example be the case if new attacks are seen which were not present in the training data. In order to do this, we use a variant of the well-known k-fold cross validation method. This method partitions the training dataset in k parts, and iteratively trains the classifier on $k-1$ parts, while keeping the remainder for validation. This can be used to select classifier parameters that will also have good performance in unseen data. While normally the partition is random, in this paper we also examine a *sequential* partition. Due to our method of data collection, this guarantees that most attacks will only be present in some folds. Consequently, this mimics real-world conditions more accurately, since some attacks will never be present during training. Ultimately, this enables us to make a less optimistic tuning of the classifiers' hyper-parameters and hopefully to a more robust performance. The cross-validation and hyper-parameter selection method is described in detail in Section 4.2. We compare this *sequential* cross-validation method with standard *random* cross-validation both in terms of classification error and in terms of expected cost.

More precisely, the contributions of this paper are the following: (a) Firstly, we perform a thorough comparison of four well-known classification algorithms for intrusion detection in Mobile Ad hoc Networks (MANETs), for both simple and cost-sensitive classification. (b) Secondly, we investigate how the performance of the classifiers is affected if the tuning of the hyper-parameters has been performed with random or sequential cross-validation.

All experiments are reported on datasets produced via extended simulations; consequently the ground truth is always known. We perform an unbiased comparison, whereby we tune the hyperparameters of all five models, using a proper experimental protocol, where the algorithms are tuned before seeing any actual test data.

In all cases, we compare the performance of the classification models under different traffic conditions, including: the mobility of the network, the number of malicious nodes, the sampling interval time (i.e. the amount of time statistics are collected before the classifier makes a decision) and the type of attacks. For the performance comparison we use five well-known and efficient classification algorithms (i.e. MultiLayer Perceptron (MLP), Linear classifier, Naïve Bayes classifier, Gaussian Mixture Model (GMM), Support Vector Machines (SVMs)). For the performance comparison with and without cost-sensitive classification we use four of them since the employment of cost-sensitive classification in Support Vector Machines (SVMs) is not a proper probabilistic model. We have selected features from the network layer for MANET and we investigate the performance of the classification algorithms for four types of attacks (i.e. Black Hole, Forging, Packet Dropping and Flooding attacks).

The remainder of the paper is organised as follows. Section 2 describes the related work while Section 3 describes the quality metrics used for the comparison of the employed classification models. Section 4 describes the simulation environment and the experimental results, while Section 5 concludes the paper.

2. Related work

Intrusion detection is a mature field in network security. While there are many possible approaches, such as rule-based systems [37] and anomaly detection systems [28], this paper focuses particularly on systems based on classification algorithms. In particular, we investigate how these algorithms can be employed most appropriately.

Classification algorithms have been extensively used for intrusion detection, and especially for wired networks. The amount of work reported on for classification-based intrusion detection in wireless ad hoc networks is more limited.

More specifically, Zhang and Lee [39] proposed the first (high-level) Intrusion Detection System (IDS) approach specific for ad hoc networks. They proposed a distributed and cooperative anomaly-based IDS, which provides an efficient guide for the design of IDS in wireless ad hoc networks. They focused on an anomaly detection approach based on routing updates on the Media Access Control (MAC) layer and on the mobile application layer.

Huang and Lee [15] extended their previous work by proposing a cluster-based IDS, in order to combat the resource constraints that MANETs face. They use a set of statistical features that can be derived from routing tables and they apply the classification decision tree induction algorithm C 4.5 in order to detect "normal" versus "abnormal" behaviour. The proposed system is able to identify the source of the attack, if the identified attack occurs within one-hop.

Deng et al. [7] proposed two distributed intrusion detection approaches, based on a hierarchical and a completely distributed architecture respectively. The intrusion detection approach used in both of these architectures focuses on the network layer and it is based on a Support Vector Machine (SVM) classification algorithm. They use a set of parameters derived from the network layer and suggest that a hierarchically distributed approach may be a more promising solution versus a completely distributed intrusion detection approach. Liu et al. [19] proposed a completely distributed anomaly detection approach. They used MAC layer data to profile the behaviour of mobile nodes and then applied cross-feature analysis [14] on feature vectors constructed from the training data. Bose et al. [2] proposed a cooperative and distributed intrusion detection system that uses data from the MAC, routing and application layers, coupled with a Bayesian classifier.

Cabrera et al. [4] use an ensemble of classifiers obtained by training multiple C 4.5 classifiers and evaluate them on a MANET network for two types of attacks. Abdel-Fattah et al. [1] use the Conformal Predictor k-nearest neighbour and the Distance based Outlier Detection (CPDOD) algorithms to perform intrusion detection in MANETs against three types of attacks while Shim et al. [32] have used a cluster analysis technique in order to detect Sinkhole attacks in MANETs [12].

In standard classification problems the classification decision is selected in order to minimise the probability of error. However, in many problems, some errors are more serious than others. In such cases, each type of error is assigned a *cost*. The goal then is not to predict the most probable class but instead to take the decision that minimises the expected cost.

For instance, this requirement appears in authentication problems. For most authentication systems, the cost of unauthorized access and is much greater than that of wrongly denying access to legitimate users. In the same vein, in intrusion detection systems raising a false alarm has a significantly lower cost than allowing an undetected intrusion. In such cases it is wiser to take the classification decision that has the minimum expected cost rather than the decision with the lowest error probability. Although this problem has been studied a lot, especially in the domain of optimal statistical decisions [21], it has been largely ignored in the field of intrusion detection. A number of other work has considered cost-sensitive intrusion detection. Fan et al. [11] and Pietraszek et al. [30] both use wrapper algorithms (Meta-Cost [9] and Weighted [36] respectively), in conjunction with RIPPER [5]. The work of Lee et al. [17] employs a decision mechanism which directly compares response cost with decision cost and either logs the detected intrusion or performs an appropriate response. However, the detection is implicitly assumed to be infallible, and so they do not actually consider the expected cost. A more general view of cost-sensitive intrusion response is taken in [34], which clearly shows an advantage for cost-sensitive decisions, as expected. An active-learning cost-sensitive approach, employing Metacost, is examined by [20]. Finally, Ghodratnama et al. [12] have performed cost-sensitive classification using an approach similar to the one described in our previous work [25], based on the use of a k-nearest neighbour (KNN) classifier to estimate probabilities. Most of the above papers report results on KDD database since they investigate the use of cost-sensitive classification for the problem of intrusion detection in wired networks.

In our own previous work [25], we have investigated the problem of cost-sensitive classification in wired networks, where we employed statistical classifiers to minimise the expected cost. The goal of the current paper is to measure the flexibility and robustness of a set of probabilistic methods under different costs and different network conditions. Firstly, we extend our previous work to wireless networks. Secondly, we examine the impact of different cross-validation mechanisms on the robustness of the classifiers. Finally, we see how much the expected cost and decision errors change in response to changes in the dataset, classifiers and cost matrix employed. We find that, while some methods are well-behaved, others exhibit a large sensitivity.

In this paper, we perform a comparative analysis of classification algorithms for the problem of intrusion detection in MANETs. We use the same experimental protocol (tuning of hyper-parameters) and training/testing data-sets generated through varying traffic conditions (regarding the mobility of the network, the number of malicious nodes and the types of attacks) in order to examine the robustness of well-known classification algorithms; something quite important since the previous approaches investigate only a few of those classifiers and performance comparisons cannot be implemented since different datasets are used each time. Additionally, the goals of our paper are: (a) to investigate if different cross-validation methods might improve the performance of the classifiers when unknown types of attacks appear in the test set and (b) how *weighted* classification may improve the accuracy of classifiers for the problem on intrusion detection in MANETs.

3. Intrusion detection using classification

We employ statistical classification algorithms to order to perform intrusion detection in MANETs. Such algorithms have the advantages that they are largely automated, that they can be quite accurate, and that they are rooted in statistics. For that reason, they are prime candidates for use in cost-sensitive classification problems. After training, they can be used for detection with arbitrary cost matrices. They have extended applications including intrusion detection in wired networks [18], they have been extensively studied, both theoretically and experimentally, and used in many applications with a high degree of success.

3.1. Intrusion detection model

The IDS architecture we assume is composed of multiple local IDS agents, which are responsible for detecting possible intrusions locally. However, during the training phase, we collect data locally, merge it and then use it to adapt the classifier models offline. During the testing phase, the resulting classification rule is transmitted to the local IDS agents, who then perform detection independently.

3.2. Cost-sensitive classification

All classifier models are trained so as to predict the probability of every class, given an observation. When using the models to make decisions, some types of erroneous classification decisions may be more important to us than others. This can be modelled by specifying a set of cost for each type of error. This cost matrix can be used with any classifier that has been trained to predict class probabilities.

Given a specification of costs for correct and incorrect predictions, the class decision should be the one that leads to the lowest expected cost, where the expectation is computed using the conditional probability of each class given

Table 1
C_W a weighted cost matrix for the 5-class problem. N denotes normal traffic, while $A_1, A_2, \ldots$, denote different types of attacks.

Class	Decision				
	N	A_1	A_2	A_3	A_4
N	0	1	1	1	1
A_1	10	0	1	1	1
A_2	10	1	0	1	1
A_3	10	1	1	0	1
A_4	10	1	1	1	0

the example, according to our model.[1] More formally, for a set Ω of k classes let a $k \times k$ matrix C such that $C(i,j)$ is the expected cost of predicting class i when the true class is j. If $i = j$ then the decision is correct, while if $i \neq j$ the decision is incorrect. Furthermore, let Y, H be random variables denoting the actual and hypothesised class labels. For any observations $x \in X$ the optimal decision will be the class i that minimises a loss function equal to the expected cost:

$$L(x,i) = E[C|X = x, H = i] \equiv \sum_{j\in\Omega} \mathbb{P}(Y = j|X = x)C(i,j) \qquad (1)$$

where $\mathbb{P}(X|Y)$ denotes the conditional distribution of class labels given an observation, according to our model. In this framework, all that is necessary is a model that can estimate this probability. The cost-sensitive decision-making function $f:S \to \Omega$ would then simply choose the definition i that minimises the expected cost given the decision and the example.[2] More formally,

$$f(x) = \arg\min_{i\in\Omega} L(x,i). \qquad (2)$$

The form of the cost matrix C will depend on the actual application. In general, it is reasonable to choose the diagonal entries equal to zero, i.e. $C(i,j) = 0$ for $i = j$, since correct classification normally incurs no cost. The other entries specify the cost of incorrectly misclassifying an example of class j as belonging to class i. They should be non-negative if the diagonal is zero, i.e. $C(i,j) \geqslant 0$ for $i \neq j$. Note that when this is equal to *1*, the cost measure is the same as the *classification error* (CE) measure.

As an example, consider a cost matrix C for two classes, positive (1) and negative (2). The cost of a *false positive* is $C(1,2)$ and we can set $C(1,1) = C(2,2) = 0$, i.e. a correct classification will have no cost. For intrusion detection applications, it is common to refer to attacks as positive and normal instances as negative example.

A commonly used cost matrix is the one that measures the *classification error* (CE):

$$C_1(y,y') = \begin{cases} 0, & \text{if } y = y' \\ 1, & \text{if } y \neq y' \end{cases} \qquad (3)$$

However, a *false negatives* (FN) is usually considered a worse kind of error than a *false positive* (FP). The matrix C should reflect that, by having $C(1,2) \geqslant C(2,1)$. For this reason, in this paper also considers the weighted cost matrix C_W, shown in Table 1.

These matrices shall be used in two different ways. Firstly, to make decisions based on the class probabilities given by the classifier, as described in this section. Secondly, to evaluate the quality of the decisions taken, will be described in the following section.

3.3. Algorithmic comparisons and quality metrics

When comparisons are made between algorithms, it is important to use the same measure of quality. For a given classification algorithm $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the observation space and $\mathcal{Y}$ is the set of classes, a common measure of quality is the expected value of the cost C measured over an independent test set D,

$$\widehat{E}(C|D) = \frac{1}{|D|}\sum_{d\in D} C(f(x_d), y_d), \qquad (4)$$

where x_d is the observation of example d and y_d is its class.

In this paper we shall perform the evaluation using two cost matrices. Firstly, C_1, the uniform cost matrix, which measures the *classification error* (CE). Secondly, C_w, (shown in Table 1), the weighted cost matrix, which measures the *weighted classification error* (WCE).

However, in much of the literature, the *Detection Rate* (DR) and the *False Alarm* (FA) rate are used instead:

$$\mathrm{DR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \quad \mathrm{FA} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} \qquad (5)$$

where TP, TN, FP, FN, denote the number of true (TP, TN) and false (FP, FN) positives and negatives respectively. The goal of an effective intrusion detection approach is to reduce to the largest extent possible the *False Alarm* rate (FA) and at the same time to increase the *Detection Rate* (DR). Since this is not usually possible, a trade-off between the two quantities is often sought instead. While such a trade-off may be automatically accomplished through the use of an appropriate cost matrix,[3] in this paper we will only use these qualities as a secondary alternative comparison metric.

3.4. Classification models

The computation of class probabilities is model-dependent. Ideally one would assume a Bayesian viewpoint and consider a distribution over all possible models in a set of models, however in this case we will only consider point distributions in model space, i.e. a single parameter vector in the parameter space. While this can cause problems with overfitting, we will use cross-validation to avoid this potential pitfall. All the models we use require labelled

[1] The implicit dependency on some model m can be made explicit by conditioning everything on the model. Then the expectation would be written $E[C|x,f,m]$ and the conditional class probability $P(y|x,m)$.

[2] Which of course is not necessarily identical to the decision with the minimum error probability. Furthermore, this framework is easily extensible to the case where the set of decisions differs from the set of class labels.

[3] Let the expected cost be $E[C] = qP(H = 1|C = 2)P(C = 2) + rP(H = 2|C = 1)P(C = 1) = q(1 - \mathrm{DR})P(C = 2) + r\mathrm{FA}\ P(C = 1)$, where 2 denotes a positive example. Setting $r = 1/P(C = 1)$ and $q = k/P(C = 2)$ we obtain a cost function minimising $\mathrm{FA} - k\mathrm{DR}$, with k being a free parameter specifying the trade-off we are interested in.

training data for their creation. During this training phase, we do not make use of the cost matrix, but find the model parameters that minimise the classification error by cross-validation the training dataset. After training, each model under consideration returns $\mathbb{P}(Y = y|X = x)$, the probability that the class is y, given that the current observation is x. During evaluation, the decision to be taken will depend on the cost matrix used. Thus, we plug in the return class probability to (1), to calculate the loss of each classification decision, and then we pick the decision with the minimum expected loss, according to (2).

The classification models we have used for intrusion detection are: the MultiLayer Perceptron (MLP), the Linear model, the Gaussian Mixture Model (GMM), the Naïve Bayes model and the SVM model. Some background information about these models is given in Appendix A for completeness.

4. Experiments

In order to examine the performance of the classification algorithms, we conducted a series of experiments under varying conditions. For each experiment, we first select hyper-parameters and train models on an independent training set, using cross-validation [24]. We compared two different cross-validation methods for hyperparameter selection. During this phase, no use of the cost matrix is made. The second phase jointly evaluates the algorithms, the hyperparameter selection method, as well as the effect of the cost matrix used for taking decisions, on an independent test set, generated from a new simulation run. We evaluate the methods on both CE and WCE, as well as in DR and FA.

4.1. Simulation environment

In order to evaluate our approach we simulated a Mobile Ad hoc Network (MANET) and we conducted a series of experiments. We have assumed that the network has no preexisting infrastructure and that the employed ad hoc routing protocol is the Ad hoc On Demand Distance Vector (AODV [29]). We implemented the simulator within the GloMoSim [13] library. Our simulation models a network of 50 hosts placed randomly within an $850 \times 850\ \mathrm{m}^2$ area. Each node has a radio propagation range of 250 m and the channel capacity was 2 Mbps. The nodes in the simulation move according to the 'random way point' model. At the start of the simulation, each node remains stationary for a period equal to the *pause time*, then randomly selects and moves towards a destination with a speed uniformly lying between zero and the maximum speed. On reaching this destination it pauses again and repeats the above procedure till the end of the simulation. The minimum and maximum speed is set to 0 and 20 m/s, respectively, and pause times at 0, 200, 400, 700 s. The simulation time of the experiments was 700 s, thus a pause time of 0 s corresponds to the continuous motion of the node and a pause time of 700 s corresponds to the time that the node is stationary.

Each node is simulated to generate Constant Bit Rate (CBR) network traffic at 2 Mbps. The size of the packets sent by each node varies from 128 to 1024 bytes. We have studied the performance of the classification algorithms for various sampling intervals (5, 10, 15, 30 s) in order to study how quickly these algorithms can perform intrusion detection. The sampling interval dictates both the interval for which the statistical features are calculated, and the period between each classification decision. We expect that longer intervals may provide more information, but with the cost of slower detection. We have also evaluated the performance of the classification algorithms for 5, 15 and 25 malicious nodes. In each case the number of all nodes in the network is set to 50.

In our experiments, we have simulated four different types of attacks:

- Flooding **attack:** We have simulated a Flooding attack [38] for multiple paths in the network layer, where each malicious node sends forged RREQ (Route REQuest) packets randomly to all nodes of the network every 100 ms.
- Forging **attack**: We have simulated a Forging attack [27] for RERR (RouteERRor) packets, where each malicious node modifies and broadcasts (to a selected victim) a RERR packet every 100 ms leading to repeated link failures.
- Packet Dropping **attack**: We have simulated a selective Packet Dropping [8] attack, where each malicious node drops all RERR packets leading legitimate nodes to forward packets in broken links.
- Black Hole **attack**: In a Black Hole attack [33], a malicious node advertises spurious routing information, thus receiving packets without forwarding them but dropping them. In the Black Hole attack we have simulated the scenario where each time a malicious Black Hole node receives a RREQ packet it sends a RREP (RouteREPly) packet to the destination without checking if the node has a path towards the selected destination. Thus, the Black Hole node is always the first node that responds to a RREQ packet and it drops the received RREQ packets. Furthermore, the malicious Black Hole node drops all RREP and data packets it receives if the packets are destined for other nodes.

An important decision is the selection of feature vectors that will be used in the classification. The selected features should be able to succinctly represent network activity, while differentiating between "normal" and "abnormal" activity. We have selected the following features from the network layer:

- RREQ *Sent:* indicates the number of RREQ packets that each node sends.
- RREQ *Received:* indicates the number of RREQ packets that each node receives.
- RREP *Sent:* indicates the number of RREP packets that each node sends.
- RREP *Received:* indicates the number of RREP packets that each node receives.
- RERR *Sent:* indicates the number of RERR (Route Error) packets that each node receives.

- RERR *Received:* indicates the number of RERR packets that each node sends.
- Data *Sent:* indicates the number of Data packets that each node sends.
- Data *Received:* indicates the number of Data packets that each node receives.
- *Number of Neighbours:* indicates the number of one-hop neighbours that each node has.
- PCR (*Percentage of the Change in Route entries*): indicates the percentage of the changed routed entries in the routing table of each node. PCR is given by $(|S_2 - S_1| + |S_1 - S_2|)/|S_1|)$, where $(S_2 - S_1)$ indicates the newly increased routing entries and $(S_1 - S_2)$ indicates the deleted routing entries during the time interval $(t_2 - t_1)$.
- PCH (*Percentage of the Change in number of Hop*): indicates the percentage of the changes of the sum of hops of all routing entries for each node. PCH [35] is given by $(H_2 - H_1)/H_1$, where $(H_2 - H_1)$ indicates the changes of the sum of hops of all routing entries during the time interval $(t_2 - t_1)$.

For each sampling interval time (5, 10, 15, 30 s) we have created one training dataset, where each training instance contains summary statistics of network activity for the specified interval using all the above features and in addition, the type of attack performed during this interval. This enables us to use supervised learning techniques for classification. Each training dataset was created by running different simulations with duration 700 s for different network mobility (pause time equal to 0, 200, 400, 700 s) and varying numbers of malicious nodes (5, 15, 25). The derived datasets from each of these simulations were merged and one training dataset was produced for each sampling interval. A similar procedure was followed in order to produce the testing datasets.

4.2. Cross-validation and hyperparameter selection

Cross-validation [24] is an approach that has extensively been used as an unbiased hyper-parameter selection method for machine learning techniques. The main idea is that the training dataset D is split in k-folds (parts), such that $D = \bigcup_{i=1}^{k} D_i$ with $D_i \cap D_j = \emptyset$ for all $i \neq j$. For a given choice of model and hyper-parameters, the following procedure is performed for k iterations: At the i-th iteration, the subset D_i is used for validating the classifier performance, and the classifier is trained on the remaining data, i.e. on $D \backslash D_i$. The performance itself is measured on D_i. Consequently, we obtain a set of k measurements, which are then averaged. The main question we pose in this paper is how to select the folds.

The usual procedure is to perform a random partition, i.e. to randomly assign folds to each data record, as seen at the top of Fig. 1. This is a usually a good choice for i.i.d data [10]. However, in intrusion detection the testing dataset can be significantly different from the training dataset. For that reason, we would like to have a procedure which is more robust to this. For that reason, we examined whether *sequential* cross-validation, which is illustrated at the

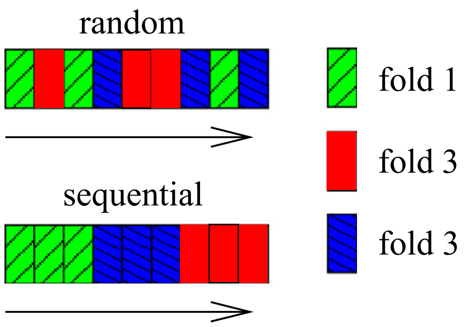


Fig. 1. Cross-validation illustration on a dataset with $t = 9$ records. We perform 3-fold cross-validation. Each fold is represented by a different colour. In random cross-validation, folds are assigned randomly to each record. In the sequential variant, folds are assigned sequentially.

bottom of Fig. 1. This allocates folds sequentially, and may protect against the selection of too optimistic hyper-parameters.

In both cases, for our experiments we used $k = 10$, thus obtaining *10* measurements CE_i of the classification error. For each hyper-parameter choice θ in the set of candidates Θ, we recorded the average cross-validation classification error $\mathrm{CE}(\theta) = k^{-1}\sum_{i=1}^{k}\mathrm{CE}_i(\theta)$ and selected the one with the lowest average error, i.e. $\theta^* = \arg\min_\theta \mathrm{CE}(\theta)$. We then re-trained the classifier on the full dataset D, using θ^*. The trained classifier was then evaluated on the independent test set using either a uniform or a weighted cost matrix.

For the MLP classifier we have tuned three parameters: the *learning rate* (η), the *number of hidden units (nh)* and the *number of iterations (T)* used in the stochastic gradient descent optimisation. For the selection of the most appropriate parameters, we have followed the following procedure.

Keeping *nh* equal to 0 we selected the appropriate η among values that range between 0.0001 and 0.1 with step 0.1 and the appropriate T selecting among 10, 100, 500 and 1000. Having selected the appropriate η and the appropriate T, we examined various values in order to select the appropriate *nh*. We selected the best among 10, 20, 40, 60, 80, 100, 120, 140, 160, 320. Additionally, we used the MLP model with no hidden units as a Linear model.

For the GMM we also tuned three parameters, i.e. the *threshold* (τ), the *number of iterations (T)* and the *number of Gaussian Mixtures (ng)*. Keeping *ng* fixed to 20, we selected the appropriate τ from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and the $T \in \{25, 100, 500, 1000\}$ For the selection of the appropriate *ng*, after selecting τ,T we selected *ng* from $\{10, 20, 40, 60, 80, 100, 120, 140, 160, 320\}$. Additionally, we used the GMM model with one Mixture component as a Naïve Bayes model.

For the SVM we tuned two parameters, i.e. the standard deviation (σ) for the Gaussian kernel and the regularisation parameter c which represents the trade-off between the size of the margin and the number of misclassified examples. For the selection of the appropriate combination of σ and c, we selected σ from $\{1, 10, 100, 1000\}$ and c from $\{1, 10, 100, 1000\}$.

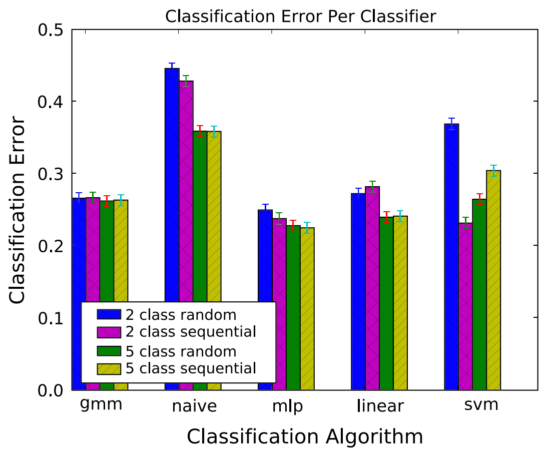


Fig. 2. Classification error (CE) per classifier when tuning is performed either with *random* or *sequential* cross-validation

4.3. Experimental results

In the experiments, we compare the performance of classifiers tuned with either sequential or random cross-validation, on an independently generated testing dataset. Each classifier is combined with either a uniform or weighted cost matrix for making classification decisions. The performance is evaluated both in terms of the classification error (CE) and the weighted classification error (WCE). The expected result is that using the uniform cost matrix for classification will result in a lower CE than using a weighted cost matrix, while the converse will hold for WCE.

Fig. 2 depicts how the *classification error* (CE) varies for each classifier when the tuning has been performed either *randomly* or *sequentially* for binary and multi-class classification. For these experiments we have used the training and testing datasets for different sampling intervals (5, 10, 15, 30 s) and took the average for all sampling intervals for each classifier. It is clear that the Naïve Bayes classifier has the highest CE, while the lowest CE is achieved by the MLP. In the two-class case, we can see that the sequential method significantly increases the robustness of Naïve Bayes, MLP and SVM. However, it has no significant effect in the five-class case for any classifier apart from the SVM. In general, the SVM appears to be extremely sensitive to the choice of cross-validation method. This is evident in Fig. 3 which depicts the cumulative differences in CE between *random* and *sequential* schemes for the five classifiers under consideration for different sampling intervals (dt). We may conclude from these results that, when cross-validation.

Fig. 4a depicts the performance of four classifiers in terms CE for the two and five-class problems when a *uniform* or a *weighted* cost matrix is used. In the uniform case, the cost matrix is such that all misclassifications carry the same cost. In the second (*weighted*) case, failure to detect an attack has a cost 10 times greater than that of false alarms (see Table 1). The *weighted* matrix significantly improves classification performance in the two-class case for all classifiers. In the five-class case however, we observe an opposite but much smaller effect. Neither of these two effects are surprising since, in the five-class case we are effectively placing little importance in misidentifying attacks. This becomes clearer once we look at the *weighted classification error* (WCE).

In fact Fig. 4b depicts the performance of the same four classifiers for the same problems of binary and multi-class classification but this time in terms of the *weighted classification error* (WCE). In both cases (binary and multi-class classification) we see the expected result that the WCE is lower when the weights of the classes are taken into consideration in the classification algorithm.

The differences between classification using *weighted* and *uniform* cost matrices are shown in more detail in Fig. 5. This shows CE and WCE for both the two-class and

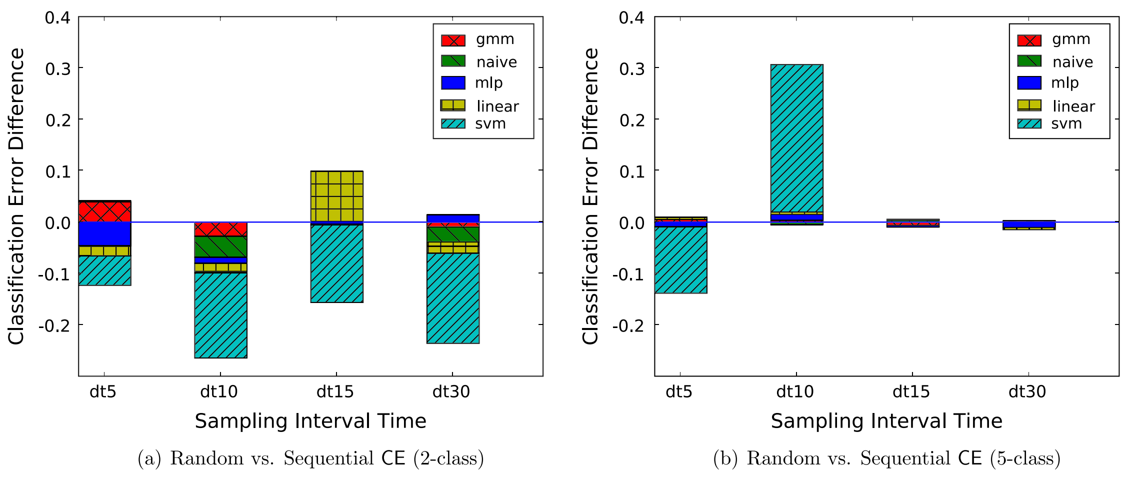
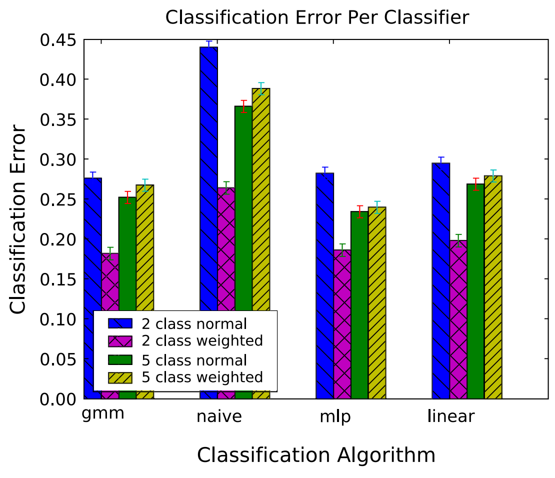
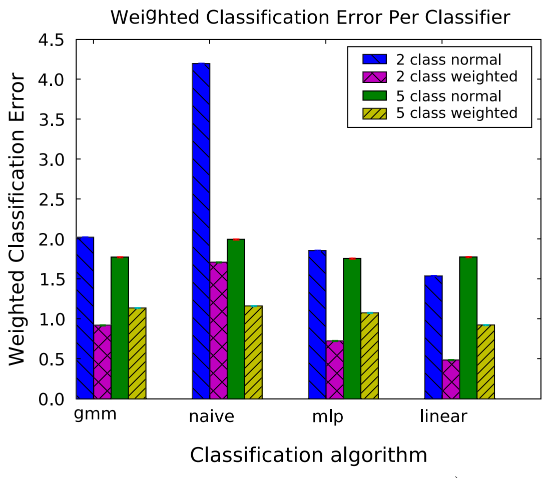


Fig. 3. Cumulative differences in (CE) between *random* and *sequential* cross-validation tuning. For every method shown, positive values indicate that *random* tuning performs better than *sequential* and vice versa. Results are given for sampling interval times $dt \in \{5, 10, 15, 20\}$ and four different classifiers. It is clearly visible from this figure that, for the two-class problems, the *sequential* tuning performs better. For the five-class problems, it is clear that the SVM classifier's performance is very sensitive.

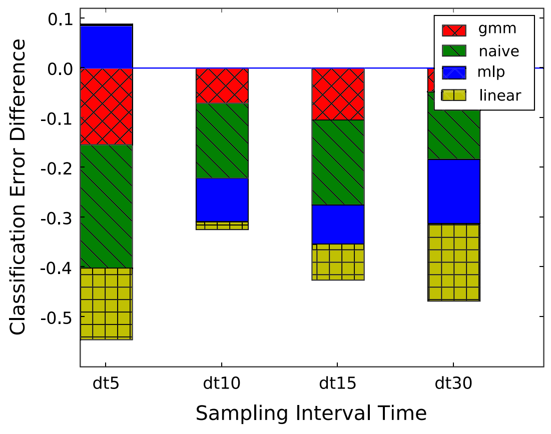


(a) Uniform matrix CE(CE)

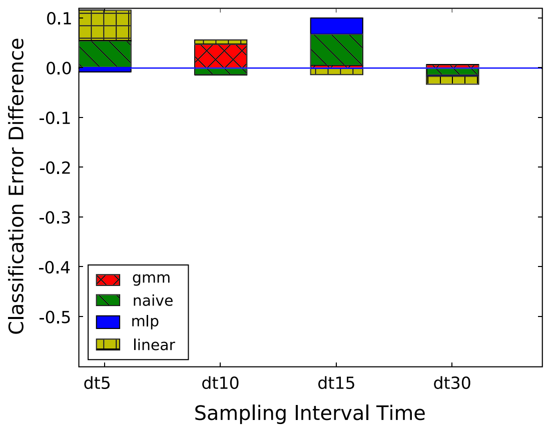


(b) Weighted matrix CE(WCE)

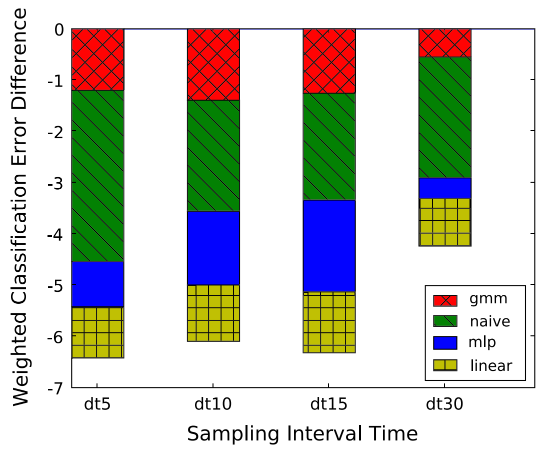
Fig. 4. Performance of normal and weighted classification when we measure either CE or WCE for 2 and 5 class problems.



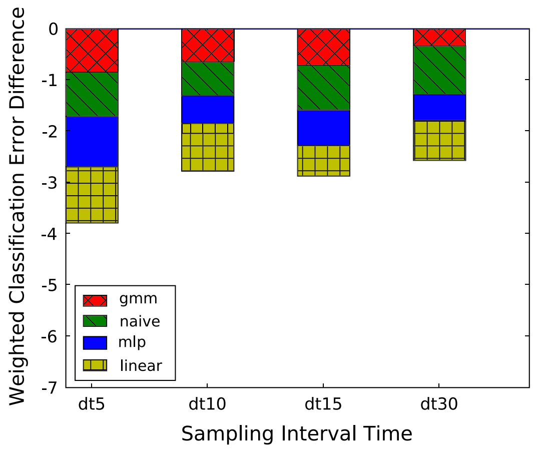
(a) CE difference (2-class)



(b) CE difference (5-class)

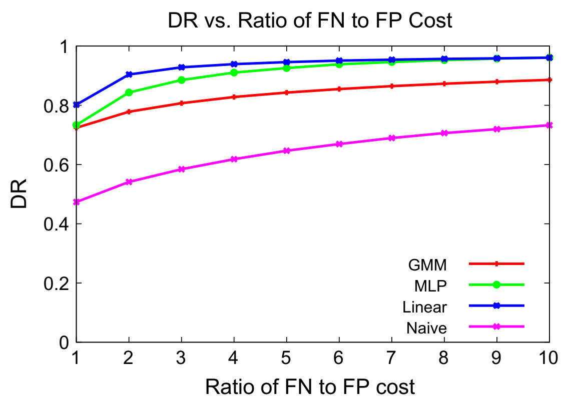


(c) WCE difference (2-class)

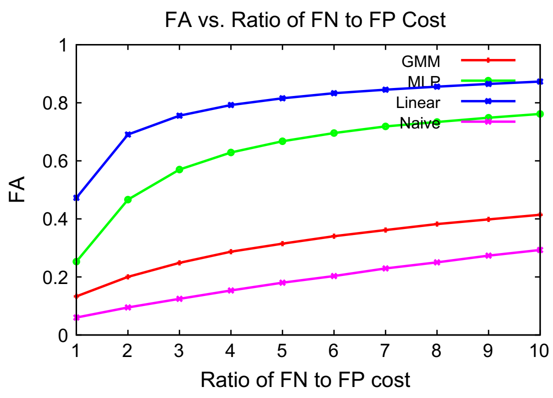


(d) WCE difference, (5-class)

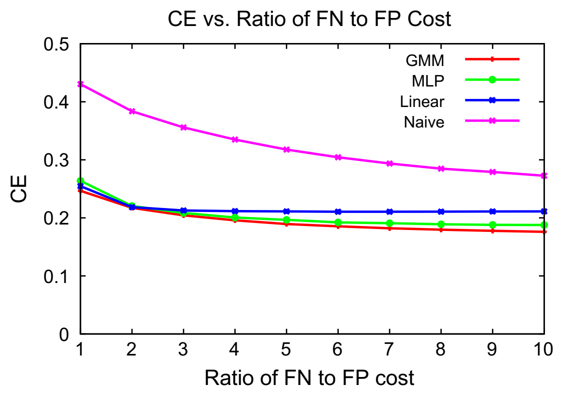
Fig. 5. Cumulative difference between *uniform* and *weighted* classifiers. Negative values indicated that the *weighted* classifier is better. The first row (a and b) shows the relative *classification error* (CE), while the second row (c and d) shows the relative *weighted classification error* (WCE). Results are shown for four different sampling interval times {5, 10, 15, 30}. It can be seen that using a *weighted* classifier results in clearly better performance for WCE. When the CE is measured, the *weighted* classifiers are slightly worse than the *uniform* classifiers for the 5-class problem, and significantly better for the 2-class problem.



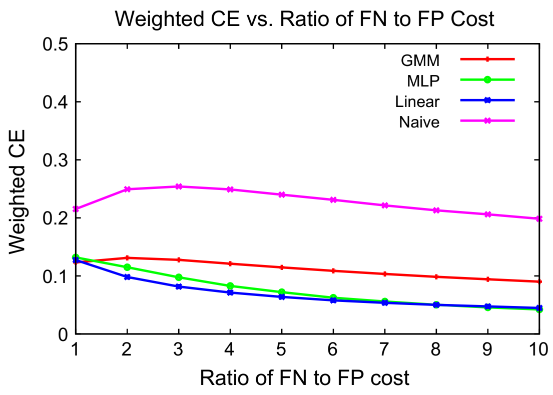
(a) DR vs. Ratio of FN to FP Cost



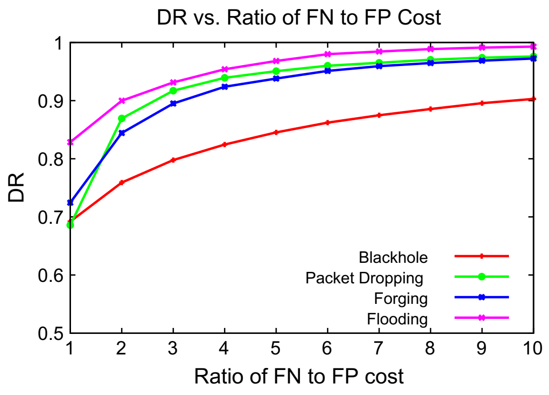
(b) FA vs. Ratio of FN to FP Cost



(c) CEvs. Ratio of FN to FP Cost



(d) WCEvs. Ratio of FN to FP Cost



(e) DR vs. Ratio of FN to FP Cost for all Types of Attack

Fig. 6. Classification error difference and weighted classification error difference for various conditions.

five-class case, for all classifiers, as the sampling interval time varies. It can be seen that in all cases, WCE is reduced when a *weighted* classifier is used, as expected. In addition, there is an unexpected improvement in CE for the two-class case when a *weighted* classifier is used.

Finally, we wanted to see what was the effect of varying the weighted cost matrix C_W with respect to the *Detection Rate* (DR), the *False Alarm Rate* (FA), the *classification error* (CE) and the *weighted classification error* (WCE). For these experiments, we have used the training and testing datasets that correspond to sampling interval *dt*15 s and performed binary classification. Overall, Fig. 6 shows that the classifiers behave 'nicely', that is, the detection rate smoothly increases with the ratio. Thus, the class probabilities they output are a good measure of uncertainty. From Fig. 6a, it is obvious that the highest *Detection Rate* (DR) is achieved for the MLP and the Linear classifiers while the lowest *Detection Rate* (DR) for the Naïve Bayes classifier.

Correspondingly, the highest *False Alarm Rate* (FA) and with high increase, as we increase the cost (ratio FN to FP) is observed for the Linear and MLP classifier. Thus, it is obvious that the classifiers that present the highest *Detection Rate* (DR) also present high *False Alarm Rate* FA. The same picture holds for FA, with the differences being more pronounced, as can be seen in Fig. 6b. It is interesting to see that, for example, the further the matrix is from uniform, the less the classification error is (Fig. 6c).

The same holds for both CE and WCE From Fig. 6c it is obvious that GMM is the most robust classifier, since it achieves the lowest *classification error* (CE) uniformly for all classification costs. However, the Linear classifier on the other hand, as depicted in Fig. 6d is more robust on terms of *weighted classification error* (*weighted* CE) even though its classification performance is suboptimal. A good compromise appears to be the MLP classifier. Finally, we investigate how the cost affects the detection rate (DR) for different types of attacks. From Fig. 6e it is obvious that the effect is stronger for the Forging and Packet Dropping attacks, while the easiest attack to detect is the Flooding attack and the hardest attack to detect is the Black Hole.

5. Conclusions

This paper examined how to properly use classification methods in intrusion detection for MANETs. More precisely, in this paper we have performed a thorough analysis of five well-known classifiers. We investigated: (i) Firstly, how simple classification versus cost-sensitive classification affects performance, both in terms of *classification error* (CE) and in terms of *weighted* error (expected cost) (WCE). (ii) Additionally, we wanted to investigate how hyper-parameter tuning affects performance when new unknown attacks are included in the test dataset. In order to achieve that we compared standard *random* cross-validation with *sequential* cross-validation, so that the validation set in most folds would include previously unknown attacks. In *random* cross-validation, all attack types are included in all folds. (iii) We perform a fair comparison of all classifiers, since we use the same datasets and tune their hyper-parameters based on the same procedure. The datasets cover a broad range of conditions including various attack types, various levels of network mobility and malicious activity and finally different data collection intervals for the intrusion detection system.

Broadly speaking, the results show that surprisingly, the use of a cost matrix not only minimises expected cost (as we anticipated), but it also reduces the classification error for most classifiers (see Fig. 5). Consequently, intrusion detection is improved both with respect to classification error and with respect to cost. The sequential cross-validation method had a limited amount of success. In the two-class case, it usually improved performance, while it had little effect in the five-class case. This could be attributed to the fact that the two are essentially different problems. For the first problem, it is sufficient to learn to distinguish normal from malicious traffic.

With respect to particular models, the experimental results indicate that the Naïve Bayes classifier has the poorest performance while the best overall performance is achieved with the MLP classifier. For the four out of five classifiers the cross validation method has no significant effect in the 5-class problem, while the SVM classifier is very sensitive to this choice. For the two-class problems, a significant improvement is obtained with *sequential* cross-validation. Finally, we found that *cost-sensitive* classification, is more effective than error-minimising classification in terms of minimising the expected cost for all classifiers, which is an expected result. In addition, and perhaps somewhat surprisingly, it reduces the *classification error* in the binary case significantly for all classifiers as well. Consequently, we believe that both *weighted* classification matrices and *sequential* cross-validation tuning schemes should be part of the repertoire of intrusion detection practitioners who utilise statistical classifiers. A good direction of future work would be to validate our results with real-world data or with simulators whose parameters are derived on real-world data such as mLab [22,23].

Acknowledgements

This work was partially supported by the Marie Curie IEF Project “PPIDR: Privacy-Preserving Intrusion Detection and Response in Wireless Communications”, Grant No. 252323 and the Marie Curie IEF Project “ESDEMUU: Efficient sequential decision making under uncertainty”, Grant No. 237816.

Appendix A. Model details

This section describes the four models employed in some more detail.

MLP A specific instance of an MLP can be viewed simply as a function $g : \mathscr{X} \to \mathscr{Y}$, where g can be further defined as a composition of other functions $z_i : \mathscr{X} \to \mathscr{Z}$. In most cases of interest, this decomposition can be written as $g(x) = Kw'z(x)$ with $x \in X$, w being a parameter vector, while K is a particular kernel and the function $z(x) = [z_1(x), z_2(x), \ldots]$ is referred to as the *hidden layer*. For each of those, we have $z_i(x) = K_i(v_i'x)$ where each v_i is a parameter vector, $V = [v_1, v_2, \ldots]$ is the parameter matrix of the hidden layer and finally K_i is an arbitrary kernel. For this particular application we wish to use an MLP m, as a model for the conditional class probability given the observations, i.e.

$$\mathbb{P}(Y = y | X = x, M = m), \quad y = g(x), \tag{6}$$

for which reason we are using a sigmoid kernel for K. In the experiments we shall be employing a hyperbolic tangent as the kernel for the hidden layer, when there is one.

Linear This is essentially a special case of an MLP. In particular, when there is no hidden layer, we have $z_i = x_i$. This is the Linear model, the second model into consideration.

GMM The GMM, the third model under consideration, models conditional observation density for each class, i.e. $\mathbb{P}(X = x | Y = y, M = m)$. This can be achieved simply by using a separate set of mixtures U_y for modelling the observation density of each class y. Then, for a given class y the density at each point x is calculated by marginalizing over the mixture components $u \in U_y$, for the class, dropping the dependency on m for simplicity:

$$\mathbb{P}(X = x|Y = y) = \Sigma_u \mathbb{P}(X = x|U = u)\mathbb{P}(U = u|Y = y). \quad (7)$$

Note that the likelihood function $\mathbb{P}(X = x|U = u)$ will have a Gaussian form, with parameters the covariance matrix Σ_u and the mean vector μ_u. The term $\mathbb{P}(U = u|Y = y)$ will be represented by another parameter, the component weight.[4] Finally, we must separately estimate $\mathbb{P}(Y = y)$ from the data, thus obtaining the conditional probability given the observations:

$$\mathbb{P}(Y = y|X = x) = \frac{1}{Z}\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y), \quad (8)$$

where $Z = \sum_{y \in Y} \mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)$ does not depend on y and where we have again dropped the dependency on m.

Naïve Bayes The fourth model under consideration is the Naïve Bayes model. This is a special case of the Gaussian Mixture Model (GMM) when there is only one Gaussian mixture.

SVM The last model we evaluated is the Support Vector Machine (SVM) [3] model, which uses Lagrangian methods to minimise a regularized function of the empirical *classification error*. The SVM algorithm finds a linear hyperplane separation with a maximal margin in this hyperspace. The points that are lying on the margin are called support vectors. The main parameter of the algorithm is c, which represents the trade-off between the size of the margin and the number of violated constraints, and the kernel $K(x_i,x_j)$. In this work we will utilise SVM s with a gaussian kernel of the form $K(x_i, x_j) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\|x_i - x_j\|^2/\sigma^2)$.

The conditional class probabilities from either Eq. (7) or (8), depending on the model, can then be plugged into Eq. (1), for calculating the decision function (2).

[4] Since we use separate mixture components for each class, $\mathbb{P}(U = u|Y = y) = 0$, when $u \notin U_y$, which also allows us to drop the dependency on y in the likelihood function.

References

[1] F. Abdel-Fattah, Z. Md. Dahalin, S. Jusoh, Dynamic intrusion detection method for mobile ad hoc networks using CPDOD algorithm, International Journal of Computer Applications, vol. 2, Published by the Foundation of Computer Science, 2010. pp. 22–29.

[2] S. Bose, S. Bharathimurugan, A. Kannan, Multi-layer intergraded anomaly intrusion detection for mobile ad hoc networks, in: Proceedings of the IEEE International Conference on Signal Processing, Communications and Networking (ICSCN 2007), February 2007, pp. 360–365.

[3] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167.

[4] J.B.D. Cabrera, C. Gutiérrez, R.K. Mehra, Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks, In Information Fusion 9 (2008) 96–119.

[5] W.W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, Lake Taho, CA, USA, 1995, pp. 115–123.

[6] A. Vasiliou, A.A. Economides, MANETs for environmental monitoring, in: IEEE International Telecommunications Symposium, 2006, pp. 813–818.

[7] H. Deng, Q. Zeng, D.P. Agrawal, SVM-based intrusion detection system for wireless ad hoc networks, in: Proceedings of the 58th IEEE Vehicular Technology Conference (VTC'03), vol. 3, Orlando, FL, USA, 6–9 October 2003, pp. 2147–2151.

[8] D. Djenouri, O. Mahmoudi, M. Bouamama, D.L. Llewellyn-Jones, M. Merabti, On securing manet routing protocol against control packet dropping, in: Proceedings of the IEEE International Conference on Pervasive Services (ICPS '07), Istanbul, Turkey, 15-20 July 2007, pp. 100–108.

[9] P. Domingos, Metacost: a general method for making classifiers cost sensitive, in: Proceedings of the Fifth ACM SIGKDD Intl Conf. on Knowledge Discovery and Data Mining, San Diego, CA, USA, 1999, pp. 155–164.

[10] B. Efron, R.J. Tibshirani, An introduction to the bootstrap, Monographs on Statistics & Applied Probability, vol. 57, Chapman & Hall, 1993.

[11] W. Fan, W. Lee, S.J. Stolfo, M. Miller, A multiple model cost-sensitive approach for intrusion detection, in: Proceedings of the 11th European Conference on Machine Learning (ECML '00), Lecture Notes in Computer Science, Barcelona, Catalonia, Spain, vol. 1810, 2000, pp. 142–153.

[12] S. Ghodratnama, M R. Moosavi, M. Taheri, M. Zolghadri Jahromi, A cost sensitive learning algorithm for intrusion detection, in: Proceedings of the 18th Iranian Conference on Electrical Engineering (ICEE) 2010, May 2010, pp. 559–565.

[13] Glomosim: Global Information Systems Simulation Library, Version 2.0, 2000. <http://pcl.cs.ucla.edu/projects/glomosim>.

[14] Y. Huang, W. Fan, W. Lee, P. Yu, Cross-feature analysis for detecting ad-hoc routing anomalies, in: Proceedings of the 23rd International Conference on Distributed Computing Systems, Rhode Island, USA, 2003, p. 478.

[15] Y. Huang, W. Lee, A cooperative Intrusion detection system for ad hoc networks, in: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03), Fairfax, VA, USA, 2003, pp. 135–147.

[16] H.C. Jang, Y.N. Lien, T.C. Tsai, Rescue information system for earthquake disasters based on manet emergency communication platform, in: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly, ACM, 2009, pp. 623–627.

[17] W. Lee, W. Fan, M. Miller, S. Stolfo, E. Zadok, Toward cost-sensitive modeling for intrusion detection and response, Journal of Computer Security 10 (1–2) (2002) 5–22.

[18] W. Lee, S.J. Stolfo, K.W. Mok, A data mining framework for building intrusion detection models, in: Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 1999, pp. 120–132.

[19] Y. Liu, Y. Li, H. Man, Mac layer anomaly detection in ad hoc networks, in: Proceedings of the 6th Annual IEEE SMC Information Assurance Workshop (IAW '05), West Point, NY, USA, 15–17 June 2005, pp. 402–409.

[20] J. Long, J.P. Yin, E. Zhu, W.T. Zhao, A novel active cost-sensitive learning method for intrusion detection, in: 2008 International Conference on Machine Learning and Cybernetics, 2008, pp. 1099–1104.

[21] M.H. DeGroot, Optimal Statistical Decisions, John Wiley & Sons, 1970 (Republished in 2004).

[22] A. Karygiannis, E. Antonakakis, mLab: A mobile ad hoc network test bed, in: 1st Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, IEEE International Conference in Pervasive Services 2005 – ICPS'05) July 14, Santorini, Greece, 2005.

[23] A. Karygiannis, E. Antonakakis, A. Apostolopoulos, Host-based network monitoring tools for MANETS, in: Proceedings of the 3rd ACM International Workshop on Performance Evaluation of Wireless Ad hoc, Sensor and Ubiquitous Networks (PE-WASUN '06), ACM, 2006, pp. 153–157.

[24] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 2, 1995, pp. 1137–1145.

[25] A. Mitrokotsa, M. Tsagkaris, C. Douligeris, Intrusion detection in mobile ad hoc networks using classification algorithms, in: Proceedings of the 7th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2008), Palma de Mallorca, Spain, 23–27 June, Springer, 2008, pp. 133–144.

[26] A. Mitrokotsa, C. Dimitrakakis, C. Douligeris, Intrusion detection using cost-sensitive classification, in: Proceedings of the 3rd European Conference on Computer Network Defense (EC2ND 2007). LNEE, Heraklion, Crete, Greece, October 2007, Springer-Verlag, 2007, pp. 35–46.

[27] P. Ning, K. Sun, How to misuse AODV: a case study of insider attacks against mobile ad-hoc routing protocols, in: Proceedings of the 2003 IEEE Workshop on Information Assurance (IAW '03), West Point, NY, USA, 18–20 June 2003, pp. 60–67.

[28] A. Patwardhan, J. Parker, A. Joshi, M. Iorga, T. Karygiannis, Secure routing and intrusion detection in ad hoc networks, in: Proceedings of the 3rd International Conference on Pervasive Computing and Communications (PerCom 2005), 8–12 March 2005, pp. 191–199.

[29] C.E. Perkins, E. Royer, S. Das, Ad hoc on-demand distance vector (AODV) RFC 3561, 2003.

[30] P. Pietraszek, Using adaptive alert classification to reduce false positives in intrusion detection, in: Proceedings of Recent Advances in Intrusion Detection 7th International Symposium (RAID'04), Lecture Notes in Computer Science, Sophia, Antipolis, France, vol. 3224, Springer-Verlag, 2004, pp. 102–124.

[31] B.C. Seet, G. Liu, B.S. Lee, C.H. Foh, K.J. Wong, K.K. Lee, A-STAR: a mobile ad hoc routing strategy for metropolis vehicular communications, in: NETWORKING 2004. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications, Springer, 2004, pp. 989–999.

[32] W. Shim, G. Kim, S. Kim, A distributed sinkhole detection method using cluster analysis, Expert Systems with Applications 37 (12) (2010) 8486–8491.

[33] M.A.I. Shurman, S.M. Yoo, S. Park, Black hole attack in mobile ad hoc networks, in: Proceedings of the 42nd ACM Southeastern Conference (ACMSE '04), Huntsville, AL, USA, 2–3 April 2004, pp. 96–97.

[34] C. Strasburg, N. Stakhanova, S. Basu, J.S. Wong, A framework for cost sensitive assessment of intrusion response selection, in: Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference, 2009.

[35] B. Sun, Intrusion Detection in Mobile Ad Hoc Networks. PhD thesis, Computer Science, Texas A& M University, 2004.

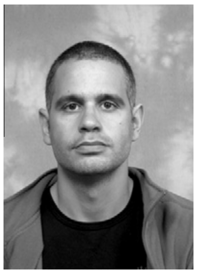
[36] K. Ting, Inducing cost-sensitive trees via instance weighting, in: Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery, Lecture Notes in Artificial Intelligence, vol. 1510, Springer-Verlag, 1998, pp. 137–147.

[37] G. Vigna, S. Gwalani, K. Srinivasan, E.M. Belding-Royer, R.A. Kemmerer, An intrusion detection tool for AODV-based ad hoc wireless networks, in: Proceedings of the Annual Computer Security Applications Conference (ACSAC) 2004, 6–10 December 2004. ACSAC'04, 2004, pp. 16–27.

[38] P.Yi, Y. Jiang, Y. Zhong, S. Zhang, Distributed intrusion detection for mobile ad hoc networks, in: Proceedings of the 2005 Symposium on Applications and the Internet Workshop (SAINTW '05), 2005, pp. 94–97.

[39] Y. Zhang, W. Lee, Y. Huang, Intrusion detection techniques for mobile wireless networks, Wireless Networks 9 (5) (2003) 545–556.

Katerina (or Aikaterini) Mitrokotsa is a senior researcher (Marie Curie fellow) at the LASEC group headed by Prof. Vaudenay in EPFL. Formerly, she held positions as a post-doctoral researcher in TU Delft and as a visitor assistant professor in the Department of Computer Science at the Free University (Vrije Universiteit) in Amsterdam. In 2007, she received a Ph.D in Computer Science from the University of Piraeus in Greece. She has been active both in European and National research projects while she has been awarded the Rubicon Research Grant by the Netherlands Organization for Scientific Research (NWO) and a Marie Curie Intra European Fellowship. Her research interests include intrusion detection systems, RFID security & privacy, security in wireless ad hoc networks, sequential decision making and classification algorithms.



Christos Dimitrakakis is a senior researcher (Marie Curie Fellow) at the artificial intelligence laboratory at EPFL, Switzerland. His research interests are reinforcement learning, decision theory, applied statistics, stochastic optimisation, multi-agent systems and telecommunications. He was a postdoctoral researcher at the University of Leoben, Austria, the University of Amsterdam the Netherlands and FIAS, at Goethe-University Frankfurt, in Germany. He holds a Bachelor degree from the University of Manchester Electronic Systems Engineering and a Masters degree from the University of Essex. After a stint at Atmel corporation he obtained his Ph.D. from EPFL in 2006, while working at the IDIAP Research Institute in the small, but charming, town of Martigny.