JYX JYVÄSKYLÄN YLIOPISTO UNIVERSITY OF JYVÄSKYLÄ

This is a self-archived version of an original article. This version may differ from the original in pagination and typographic details.

Author(s): Mazhelis, Oleksiy

Title: One-Class Classifiers : A Review and Analysis of Suitability in the Context of Mobile-Masquerader Detection

Year: 2006

Version: Published version

Copyright: © Author, 2006

Rights: CC BY 4.0

Rights url: https://creativecommons.org/licenses/by/4.0/

Please cite the original version:

Mazhelis, O. (2006). One-Class Classifiers : A Review and Analysis of Suitability in the Context of Mobile-Masquerader Detection. South African Computer Journal, 6(36), 29-48. https://doi.org/10.46298/arima.1877

One-Class Classifiers: A Review and Analysis of Suitability in the Context of Mobile-Masquerader Detection

O Mazhelis

Department of Computer Science and Information Systems
University of Jyväskylä
P.O. Box35, FIN-40351, Jyväskylä, Finland

ABSTRACT

One-class classifiers employing for training only the data from one class are justified when the data from other classes is difficult to obtain. In particular, their use is justified in mobile-masquerader detection, where user characteristics are classified as belonging to the legitimate user class or to the impostor class, and where collecting the data originated from impostors is problematic. This paper systematically reviews various one-class classification methods, and analyses their suitability in the context of mobile-masquerader detection. For each classification method, its sensitivity to the errors in the training set, computational requirements, and other characteristics are considered. After that, for each category of features used in masquerader detection, suitable classifiers are identified.

KEYWORDS: One-class Classifiers, Mobile Terminal Security, Masquerader Detection

1 INTRODUCTION

The problem of classification can be defined as a problem of assigning an object represented by a vector of feature values to a category of objects – class. Using a set of objects from the training set, a classifier learns to assign the category/class labels to previously unseen objects from the test set. One-class classification can be seen as a special type of two-class classification problem, where data only from one class is available for training the classifier (referred to as *one-class classifier*). One-class classifiers are applied when the data from other classes is extremely hard or impossible to collect. One such application is mobile-masquerader detection, which can be defined as the detection of an attempt to impersonate the legitimate user of a mobile terminal in order to obtain an unauthorized access to sensitive data or services authorized for that user. The risk of such impersonation is high, since the terminals are carried and, due to their small size, they are often lost [1, 2]. Furthermore, since smartphones and PDAs are often used to store personal and business names and addresses, to receive and view emails, to store corporate information, etc. [3], an impersonation of the user of such a terminal may result in an abuse of the critical personal or corporate information.

The problem of detecting masqueraders may be approached as the classification problem where the user behavioural or environmental characteristics are classified as belonging to the legitimate user or to an impostor [4]. While the data originated from the legitimate user of the device may be relatively easily collected, the data originated from the behaviour of impostors might be very difficult, if at all possible to achieve, due to privacy and coverage issues [5]. Therefore, the use of one-class classifiers is justified.

This paper is aimed at the analysis of various types of one-class classifiers and the examination of their applicability to the problem of mobile-masquerader detection. The classifier's applicability is analysed according to a simple framework taking into account both the type of the features the classifier deals with, and the characteristics of the classification method, such as robustness, computational and storage requirements, and the number of parameters to be estimated or set. The same analysis framework may, however, be adopted when considering the applicability of one-class classifiers for other application domains, where the use of one-class classifiers is justified: machine fault diagnosis, credit card fraud detection, insurance fraud detection, etc.

The paper is organized as follows. In the next Section, the problem of one-class classification is formally stated, a taxonomy of one-class classification methods is presented, and the analysis framework is described. In Section 3, multiple one-class classifiers are considered according to the introduced analysis framework. Section 4 considers the behavioural and environmental features to be used in mobile-masquerader detection, and identifies one-class classifiers potentially suitable for processing these features. Finally, conclusions to the paper are provided in Section 5.

Email: O Mazhelis mazhelis@it.jyu.fi

2 ONE-CLASS CLASSIFICATION

In this Section, the classification problem is formally stated, and the process of learning classifiers' models is discussed. The Section also proposes taxonomy of one-class classifiers, and specifies the criteria, according to which the one-class classifiers are reviewed in this paper.

2.1 Classification problem

Let an object Z be represented by a vector $\mathbf{x} \equiv (x_1, \ldots, x_{n_f})$ of the values of n_f features from the feature space $\mathcal{X}$, to which we will refer to as to the classifier's observation vector. The classification problem can be defined as a problem of assigning the object Z by to a class C_i, where $C_i,\ i = \{1, \ldots, N_C\}$ denotes the label of class i. The training dataset $\mathcal{DS}_T$ is the set of observation vectors along with the corresponding class labels: $\mathcal{DS}_T = \{((x_1, \ldots, x_{n_f})_j, y_j) | j = 1, \ldots, |\mathcal{DS}_T|\}$, where y_j is the class label. In turn, the test dataset of observations to be classified denoted as $\mathcal{DS}_C$ consists of the vectors of feature values without class labels: $\mathcal{DS}_C = \{((x_1, \ldots, x_{n_f})_j) | j = 1, \ldots, |\mathcal{DS}_C|\}$.

Using a training data-set, the classifier learns the set of parameters Θ constituting the *model* of the classifier. After that, given an unlabeled observation vector $\mathbf{x}$, the classifier produces an output $u(\mathbf{x}, \Theta)$. Possible values of the output can be $u(\mathbf{x}, \Theta) \in \{C_1, \ldots, C_{N_C}\}$, i.e. the classifier can assign to the object the label of one of N_C classes.

Alternatively, for each class C_i, the classifier may implement a real-valued discriminant function $u_{C_i}(\mathbf{x}, \Theta)$ [6], such that the greater values of the function correspond to the higher probability of class membership $P(Z \in C_i|\mathbf{x}) = P(C_i|\mathbf{x})$. In this case, the class with the highest value of discriminant function is selected:

$$\gamma(\mathbf{x}, \Theta) = \underset{i=1,\ldots,N_C}{\operatorname{agrmax}}\ u_{C_i}(\mathbf{x}, \Theta), \tag{1}$$

where γ is a mapping function. If the classifier outputs approximated probabilities $P(C_i|\mathbf{x})$, this function implements Bayes decision rule, assigning the object to the class with the highest posterior probability. This rule is known to provide the optimal classification accuracy when different classification errors have equal costs [7].

For two-class classification problem, a single discriminant function in a form

$$u(\mathbf{x}, \Theta) = P(C_1|\mathbf{x}) - P(C_2|\mathbf{x}) \tag{2}$$

is sufficient to implement the classification as [6]:

$$\gamma(\mathbf{x}, \Theta) = \begin{cases} C_1, & \text{if } u(\mathbf{x}, \Theta) \geq 0, \\ C_2, & \text{if } u(\mathbf{x}, \Theta) < 0. \end{cases} \tag{3}$$

While in the expressions above the discriminant was calculated as a function of posterior probabilities, some classification methods calculate the value of discriminant without explicit estimation of these posterior probabilities.

The probability density function (PDF) $p(C_i|\mathbf{x})$ or the parameters Θ of discriminant functions are estimated empirically using training set, e.g. by minimising an error function reflecting the misclassification error over the training set. It is assumed that the values of parameters minimising the error function over the training dataset, will also minimise the misclassification error over the complete set of allowed feature values. This ability of a classifier to generalise beyond the training dataset is referred to as generalisability.

In one-class classification, the training dataset contains only the observation vectors belonging to a class C_1, while the testing dataset includes the observation vectors from both classes C_1 and C_2. Two types of classification errors can be encountered by one-class classifiers. The type I error $\mathcal{E}_I$ occurs if the object of the class C_1 is recognised as not belonging to this class. The type II error $\mathcal{E}_{II}$ is encountered when the object from the class C_2 is considered as belonging to the class C_1. The type I errors are also referred to as false negatives (in security they are also known as false rejection errors), and the type II errors are referred to as false positives (in security they are also known as false acceptance errors). Note, that only type I error can be estimated using training data-set.

In the context of one-class classification, the parameters of PDF or the parameters of discriminant function can be evaluated only for the class C_1. In order to make the classification possible, an assumption about the distribution of the data in the second class (C_2) can be made; e.g., uniform distribution of $p(\mathbf{x}|C_2)$ may be assumed [8, 9]. After that, the calculation of posterior probabilities $p(C_i|\mathbf{x})$ is possible, and the classification is performed using the discriminant function (2). However, in practise, the value of PDF or discriminant function is often compared against a threshold t:

$$\gamma(\mathbf{x}, \Theta) = \begin{cases} C_1, & \text{if } u(\mathbf{x}, \Theta) \geq t, \\ C_2, & \text{if } u(\mathbf{x}, \Theta) < t. \end{cases} \tag{4}$$

The value of t is usually selected afterwards, when the parameters of PDF or discriminant function are estimated; it is selected such that the value of the type I error would be limited by a predefined level. Some methods, however, require the value of t to be specified in advance, and the selection of values of other parameters depends on this threshold. An example of such method is support vector data description [10].

2.2 Learning of classifiers

Before a classifier can be used to assign labels to objects, it should be trained, i.e. the parameters of the classifier's model should be determined. This is done during learning phase (also called as training phase), using training data-set. During this phase, either the probability density functions $P(C_i|\mathbf{x})$ should be estimated, or the parameters Θ of discriminant functions are to be determined.

2.2.1 Probability density estimation

When the probability density function should be estimated, parametric or nonparametric density estimation methods can be employed.

Parametric methods assume that the data are generated according to a distribution of a specific form whose parameters are to be estimated using training dataset. The task of estimating the distribution parameters is often approached as the task of maximising a likelihood function. For example, if the assumed distribution is Gaussian, i.e. $\Theta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, the parameters of the distribution can be estimated through maximising the likelihood in the form

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i \in \mathcal{DS}_T} p(\mathbf{x}_i | \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{5}$$

For the above example of Gaussian distribution, the analytical solution maximising the likelihood is known. Often, however, the task of likelihood maximisation is analytically intractable. In such cases, the parameter estimation can sometimes be performed using Expectation-Maximisation (EM) algorithm [11]. This algorithm iteratively applies two procedures referred to as expectation and maximisation steps, respectively. During the expectation step (E-step), the distribution of hidden parameters $\mathbf{z}$ (e.g. the variable indicating which component of a mixture of Gaussians generated the observation vector) is approximated, and the expectation E of the log-likelihood of parameters Θ with respect to these hidden parameters is calculated. During the maximisation step (M-step), the parameters Θ are reassigned the values maximising the expectation of log-likelihood. These E- and M-steps are iteratively repeated until convergence.

Non-parametric density estimation methods, contrary to the parametric ones, do not make specific assumptions about the underlying distribution. Rather, the form of the distribution is induced directly from the data (training dataset). Examples of these methods are Parzen density estimator, K-nearest neighbours algorithm, histograms, etc. These methods will be described in Section 3.

2.2.2 Estimation of the parameters of discriminant function

Parameters Θ of a discriminant function may be estimated by minimizing an error function of these parameters (such as sum-of-squares error function or cross-entropy error function) over the data from the training set. The error function value is evaluated using empirical data, and it reflects the degree of classifier's misclassification error over the training set. For example, the cross-entropy error function is calculated as [6]:

$$\mathcal{E}_{CE} = -\sum_{\mathbf{x}_i \in \mathcal{DS}_T} \sum_{k=1}^{2} y_{ik} \ln u_k(\mathbf{x}_i, \Theta). \tag{6}$$

where $y_{ik} = 1$, if $\mathbf{x}_i \in C_k$, and $y_{ik} = 0$, otherwise.

However, in order to apply these error functions, the observation vectors for class C_2 need to be synthesized, e.g. generated according to a specific distribution assumed for class C_2 [12]. As a result, these functions are rarely used in practice for one-class classification.

2.2.3 Generalisability

The generalisability of a classifier is its ability to generalise beyond the training dataset. The problem of generalisability can be better understood by considering the decomposition of the misclassification error into bias and variance terms. Rather than estimating the error for a particular training dataset $\mathcal{DS}_T$, this decomposition is defined for the expectation of the error over all possible training datasets. For the sum-of-square error function, and for regression rather than classification problem, this expectation can be rewritten as [6]:

$$\begin{aligned} & E_{\mathcal{DS}_T}[(u(\mathbf{x}_i, \Theta) - y_i)^2] = \\ & = \{E_{\mathcal{DS}_T}[u(\mathbf{x}_i, \Theta)] - y_i\}^2 + \\ & + E_{\mathcal{DS}_T}[\{u(\mathbf{x}_i, \Theta)] - E_{\mathcal{DS}_T}[u(\mathbf{x}_i, \Theta)\}^2] \\ & = (\text{bias})^2 + \text{variance}. \end{aligned} \tag{7}$$

where $y(\mathbf{x})$ is the regression function being approximated.

The bias component of the above equation reflects the error due to low model flexibility that is insufficient to accurately approximate the function being learnt. For example, a significant bias is expected if a linear model is used to approximate a quadratic function. In turn, the variance component reflects the variability of the learnt model across different training sets. Thus, it reflects how sensitive the classifier's model is to the choice of training set. A high variance value indicates that the model is too flexible, and that it learns, in addition to the true function $y(\mathbf{x})$, the characteristics of the training set $\mathcal{DS}_T$. In this case, the model is said to overfit the data, indicating that the model complexity is higher than the complexity of the function being approximated. For example, the approximation of a linear function by a quadratic model is likely to result in overfitting.

In order to generalise well beyond the training data-set, the classifier's model should have a low value of the variance. This can be accomplished e.g. by incorporating an additional regularisation term, punishing the models with high complexity, in the definition of an error function being minimised. Alternatively, several classifiers can be learnt, and by combining their individual classifications the variance component of the error may be reduced.

Many one-class classification methods are based on the estimation of a boundary around the training data. In order to avoid overfitting, boundaries with a lesser degree of flexibility are to be applied. For example, as shown in [13], the ellipsoid K-means is preferred to the convex polytope, which, though more flexible and hence able to produce a boundary with a tighter fit to the training data, yet does not provide

a good generality. Conversely, the ellipsoid K-means is able to achieve a tradeoff between the tightness of the fit and the generality, and consequently provides better classification accuracy on a test dataset.

It was also found that methods requiring many parameters to be tuned are more susceptible to overfitting; therefore, the methods with a small number of parameters or parameter-free methods should be used in order to alleviate the generalisability problem [14].

2.3 Taxonomy of one-class classifiers

For the purposes of the paper, a taxonomy of one-class classification methods is proposed in this subsection. The methods are divided into categories according to the following criteria:

1. The internal model used by a classifier. Following the work of [9], three types of one-class classifiers can be distinguished, including density methods, boundary methods, and reconstruction methods:

- Density methods, as the name implies, are based on the estimation of the probability density function (PDF) of the feature values $p(\mathbf{x}|C_1)$ in the complete feature space. In the absence of knowledge about the second class, the PDF for that class may be assumed uniform, i.e. $p(\mathbf{x}|C_2) =$ const. A specific form of the distribution of feature values is often unknown; it is approximated, e.g., by a mixture of Gaussians. The data in training set are assumed to be representative of the true data distribution. In classification, the PDF value corresponding to the current observation vector is compared against a threshold t.
- In reconstruction methods, contrary to the density methods, assumptions about underlying data structure are made. Namely, a model of data-generation process is assumed, and the parameters of this model are estimated during the learning phase. For classification, the reconstruction error $\mathcal{E}_{\text{reconstr}}$ reflecting the fit of current observation vector to the model is evaluated. The closer is the fit, the more likely the data were generated by this model. The discriminant function can then be implemented as $1/\mathcal{E}_{\text{reconstr}}$.
- Boundary methods do not estimate the density of the data, but rather estimate the boundary thereof. These methods calculate the distance between the observation vector being classified and the boundary built around the observation vectors in the training data-set. The distance calculation takes into account both i) the distance between the observation vectors being analysed and the observation vectors in the training data-set, and ii) the distances between the observation vectors in the training data-set. Neither the density of the data nor the data generation process is specifically modelled. Contrary to density and reconstruction methods traditionally used for multi-class classification, boundary methods are specifically targeted at the one-class classification.

2. The type of data. According to the type of features, the classifiers can be dichotomised into those based on symbolic or numeric features [15]:

- The classifiers dealing with symbolic data (also known as qualitative, discrete, or categorical) can be exemplified by the classifier based on Markov models and the classifier based on association rules. These classifiers can be applied to analyse the numeric data after these data have been transformed into a number of categories (e.g. using histograms or clustering).
- An example of the classification method dealing with numeric (also called as real-valued, continuous, or quantitative) data is K-nearest neighbour classifier. This classifier employs the measure of distance between observation vectors, and it may be difficult to define the distance measure for symbolic data. Often, however, symbolic features can be transformed into numerical by introducing boolean indicator variables, and consequently can be analysed by the classification method designed for numeric data.

3. The ability of classifiers to take into account temporal relations among features. Based on the importance of the temporal relations between features of the observation vectors, the classifiers can be divided into the ones ignoring temporal regularities and the ones whose internal model takes such regularities into account:

- An example of the method ignoring temporal relations is K-nearest neighbour classifier, for which the temporal order of the features is not important. The classifiers dealing with non-temporal data can sometimes be adapted to take into account temporal regularities. For example, a multi-layer perceptron (MLP) neural network can be used to model sequential patters by adding a feedback link between an output(s) and input(s) of the network.
- The methods based on temporal relations model how the values of features change along the time axis. An example of the methods capable of modelling temporal relations is the classifier based on Markov models. This classifier estimates the probabilities of the new states of the system on the basis of several previous states; thus, the temporal order of the system states is important for this classifier.

The produced taxonomy and selected one-class classifiers located in it are presented in Figure 1.

The division of methods according to the internal model of classifiers is aimed at making the analysis process more systematic. The methods based on different models produce classifications using different approaches: the density methods estimate the density of a new observation vector; the boundary methods calculate the distance from the vector to a learnt boundary; in turn, a reconstruction error is calculated in reconstruction methods. That is why it was decided to group the methods being considered in Section 3, according to their internal model.

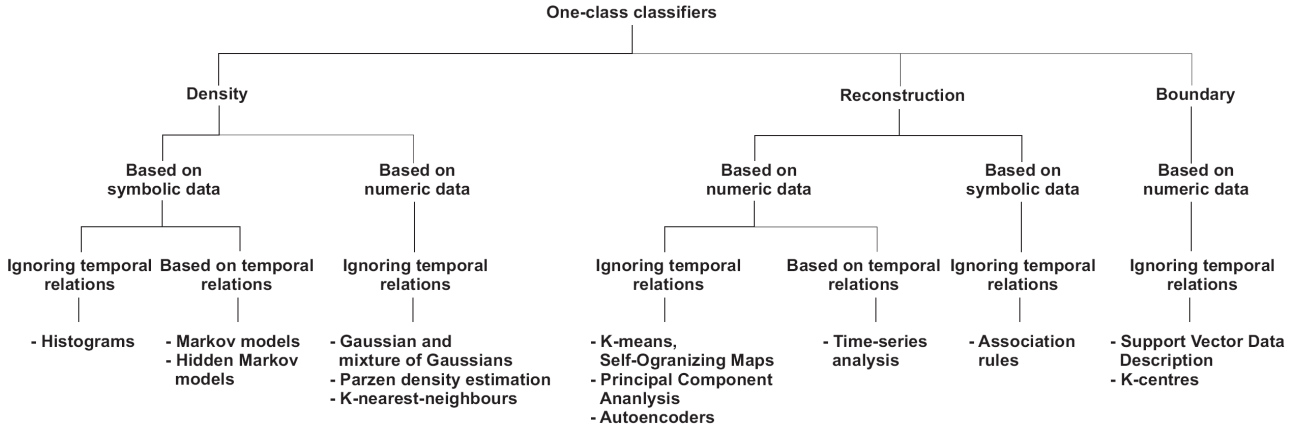

Figure 1: Taxonomy of one-class classifiers

From a practical perspective, the internal models of classifiers may be not the most important characteristic of a classification method. For example, in deciding upon the classifier to be employed in masquerader detection, the need to address temporal aspects in the data may be more important than the internal classifier's model. The type of features (numeric vs. symbolic) also may appear more important than the internal structure. Therefore, these characteristics are included into the taxonomy.

In the bottom part of the figure, the one-class classification methods are exemplified. These methods will be considered in Section 3 according to the framework of analysis presented in next subsection. The selection of the methods for this review was aimed at covering the well-known one-class classification methods, but the selection also took into account the availability of published examples of their use in the security domain. Meanwhile, in addition to the methods included in the review, a number of other one-class classification methods exist [16, 17].

The proposed taxonomy divides the space of classification methods into $3 \times 2 \times 2 = 12$ categories. However, as could be seen from the figure, several brunches in the taxonomy are missing, since no classification methods belonging to these categories are included in the review. Some of the missing branches would be non-empty if the modifications of the classification methods present in other branches would have been considered. For example, a modification of the hidden Markov model is able to analyse the numeric observation vectors [18] and should be located in the "density"–"numeric data"–"based on temporal relations" branch. For several categories, no classification methods that belong to these categories have been identified. For example, no boundary methods dealing with symbolic data or taking into account temporal relations among features have been found in literature.

2.4 Framework for analysis

In the following section, several one-class classification methods will be reviewed. For each method, the internal model of the classifier, along with the employed learning and classification processes, will be summarised. After that, the characteristics of these methods will be analysed according to the following criteria (some of them can be found in [9] and [15]):

- Robustness. In learning the classifiers, the assumption is usually made that the training dataset is a representative of the data distribution for the class C_1. However, it may further be assumed that the data in the training dataset are contaminated by a noise error component (for numeric data) or contain mislabelling errors (for symbolic data). The class labels in the training dataset are symbolic; therefore, the noise corresponds to the feature values only. The mislabelling errors may occur in the values of features as well as in the values of class labels. (The latter case corresponds to the situations when the training dataset is contaminated with the data originated from class C_2; we will refer to such observation vectors with invalid class label as to outliers.) The ability of a classifier to learn the true characteristics of the data in the presence of noise/errors, i.e. the method's robustness, is important for one-class classifiers, especially when the behavioural and environmental characteristics of users are being classified. Due to a great variability exhibited in user behaviour and environment, the values of the corresponding features are not likely to be error-free.
- Computational and storage requirements. While both the computational abilities and available storage capacity of computing facilities is increasing constantly, they are still the limiting factors prohibiting the use of some of the methods in certain applications. This is especially relevant for the mobile devices that are usually inferior to the desktop computers in computational power, battery power, and the size of available memory.
- Number of parameters to be estimated or set. The number of free parameters that should be either learnt using training set or directly set (e.g. by a user) varies among classification methods. In the context of personal mobile devices, where no security administrator is usually present, the number of parameters set by a human being

should be minimised.

- Applications in the domain of security. Reported applications of various one-class classification methods in the domains close to mobile-masquerader detection (intrusion detection [19] or fraud detection [20]) may serve as empirical evidences of applicability of these methods to the masquerader detection problem. Such evidences, however, should be considered with care, since these methods were applied mainly on desktop computers or servers rather than on mobile devices, and because the context, in which the classifiers were applied (e.g. the analysis of network packets or the analysis of system calls) may be different from the context of analysing the behaviour or environment of a mobile-device user.

3 METHODS OF ONE-CLASS CLASSIFICATION

In the following subsections, the density, the reconstruction, and the boundary methods of one-class classification will be considered separately.

3.1 Density methods

The density methods are based on the estimation of the probability density function. Several representative density methods including histograms, Markov models, Gaussian and mixture of Gaussians models, Parzen density estimation, and K-nearest-neighbours estimation used in various application areas are considered below.

3.1.1 Histograms

The histogram analysis is one of the most intuitive and widely used methods of density estimation [17]. It can be employed for the analysis of both symbolic and numeric data. In the latter case, the histogram is produced by dividing the feature space into a number of bins ("buckets"), and calculating the number of observation vectors that fall in each bin. In fact, the data in different groups are treated as having distinct symbolic values. The probability density is then estimated for each bin as the fraction of the observation vectors in this bin [6].

The histograms are relatively robust to the noise in training data as well as to mislabelling errors. However, the accuracy of estimation depends on the way the feature space is divided into bins. Too large bins result in over-smoothed density, while too small bins produce very spiky density estimation [6]. Furthermore, due to the curse of dimensionality [7], a large size of training dataset may be needed in order to estimate the density accurately.

Different techniques of partitioning data into buckets have been proposed in order to improve the estimation accuracy. Examples of these techniques are equi-width, equi-depth, and V-optimal partitioning [21]. In V-optimal partitioning, for example, a weighted variance of the values in each bucket is minimised. [22] proposed so-called self-tuning histogram whose bins can be adjusted incrementally as new observations become available.

The parameters that need to be provided or learnt depend on the type of the histogram. Usually, at least one parameter (e.g. the number of bins N_{bins}) needs to be supplied, and the number of parameters to be estimated is usually equal or greater than the number of bins. The learning of histogram is computationally inexpensive. The histograms can be constructed incrementally, discarding the observation vectors that have been considered. In this case, memory space is not needed for storing all the elements of the training set. Similarly, little computational efforts are needed for estimating density using a constructed histogram, and the histogram itself requires little storage space.

In anomaly intrusion detection, the histograms were extensively used in the design of the statistical component of IDES and NIDES [23, 24, 25, 26, 27]. Yamanishi et al. [28] also employed histograms to represent probability density for categorical variables.

3.1.2 Markov models

A Markov chain is a model of discrete-time stochastic process, i.e. the changes of an observation variable are assumed to occur at discrete points in time. The observation variable can take a finite number of values; these values designate the state of the modelled system at time τ: $x_\tau \in \{s_1, \ldots, s_{N_s}\}$.[1] Thus, this model may be suitable for modelling temporal regularities present in symbolic features.

A stationary Markov chain [29] assumes that the probability distribution at time τ depends on the state at time $\tau - 1$, and does not depend on the previous states $\tau - 2, \ldots, 1$. It is further assumed that the probability distribution does not change with time.

Assuming a set $S = \{s_1, \ldots, s_{N_s}\}$ of possible states, the Markov chain model can be represented by a transition probability matrix $\mathbf{A} = \{a_{ij}\}$, $i, j = 1, \ldots, s_{N_s}$, where $a_{ij} = P(s_j^{\tau+1} | s_j^\tau)$, and by a vector of initial probability distributions $\mathbf{\Pi} = \{\pi_i\}$, $i = 1, \ldots, s_{N_s}$, where $\pi_i = P(s_i^1)$ is the probability that initial state of the observation variable is s_i. The initial and transition probabilities can be estimated empirically as a fraction of corresponding states or transitions between states.

Given the model, the probability of a sequence of the observation variable states $s_1, \ldots, s_\tau$ at times $1, \ldots, \tau$ is estimated as

$$p_{MC}(s_1, \ldots, s_\tau) = \pi_{x_1} \prod_{k=2}^{\tau} a_{k-1,k}. \tag{8}$$

When the states of the system cannot be observed directly, a hidden Markov model (HMM) [18] is used

[1] For simplicity, a single observation variable x is used in this subsection instead of observation vector $\mathbf{x}$. The vector $\mathbf{x}$ composed of n_f features, where each feature has N_s possible values, can be transformed to a single variable having $n_f \times N_s$ possible values.

instead of the above Markov chain. In HMM, the system being in state s_τ at time τ, is assumed to emit some observation x_τ whose possible values belong to the set of visible symbols $\mathbf{v} = \{v_1, \ldots, v_{N_v}\}$. These symbols are assumed to be emitted according to probability distribution $P(v_k^\tau | s_j^\tau) = b_{jk} \in \mathbf{B}$. The transition matrix for HMM is defined as $\mathbf{A} = \{a_{ij}\},\ i, j = 1, \ldots, s_{N_s}$, where $a_{ij} = P(s_j^{\tau+1} | s_j^\tau)$.

The parameters $\mathbf{\Pi}$, $\mathbf{A}$, and $\mathbf{B}$ can be found using e.g. the Baum-Welch algorithm, implementing EM procedure [18]. Given the parameters of the model, the probability of a sequence of observations $P(v^1, \ldots, v^\tau)$ is determined using forward algorithm [7, 18].

The Markov models are relatively insensitive to a small number of mislabelling errors in the training dataset, since these errors may have little influence on the estimation of the parameters of the models. Such insensitivity enabled the successful use of the HMM for addressing the problem of speech recognition [18, 30], where a training data may be contaminated with noise.

The number of parameters being estimated is determined by the number of states N_s and by the number of visible symbols (for HMM), and is equal to $n_{\text{param}MC} = N_s^2$ for first-order Markov chain, and is equal to $n_{\text{param}HMM} = N_s^2 + N_s(N_v - 1)$ for the HMM. The user should specify the number of hidden states.

For the conventional Markov chain model, the imposed computational overhead is negligible for both learning and execution. The needed storage space is determined by the number of states and hence is relatively small.

Contrary, the classification with the HMM and especially the estimation of the HMM's parameters is computationally expensive as indicated e.g. by the results of [31] and [32]. This is due to the fact that one iteration of the training procedure requires $O(|\mathcal{DS}_T| N_s^2)$ steps, and a number of iterations are needed before the Baum-Welch algorithm converges. The space requirements during training are also high since the intermediate values need to be stored, and they require $|\mathcal{DS}_T|(2N_s + 1)$ floating point values [31].

In the security domain, Ye [33] investigated the application of conventional Markov chain model to the problem of detecting anomalies in the audit logs produced by Basic Security Module (BSM) of Solaris operation system. A number of studies in the intrusion detection employed the HMMs as one-class classifiers, e.g. [34, 31, 35, 32].

3.1.3 Gaussian and mixture of Gaussians

These methods assume that the data is distributed according to the normal (Gaussian) distribution, or according to a mixture of several Gaussian distributions [6]. The Gaussian distribution is defined as:

$$p_G(\mathbf{x}, \boldsymbol{\mu}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \times \times \exp\Big\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \Big\}, \tag{9}$$

where $\boldsymbol{\mu}$ is the mean vector and $\mathbf{\Sigma}$ is the covariance matrix. In turn, the mixture of Gaussians model extends the above Gaussian models and represents a linear combination of n_{MG} Gaussian distributions as:

$$p_{MG}(\mathbf{x}) = \frac{1}{n_{MG}} \sum_{i=1}^{n_{MG}} p_G(\mathbf{x}, \boldsymbol{\mu}_i, \mathbf{\Sigma}_i) P(i), \tag{10}$$

where the mixing parameter $P(i)$ reflects the prior probability that an observation vector is generated from i-th component of the mixture.

The number of parameters for Gaussian model is equal $n_{\text{param}G} = n_f + \frac{1}{2} n_f (n_f - 1)$ and for mixture of Gaussians is $n_{\text{param}MG} = n_{MG}(n_{\text{param}G} + 1)$. The parameters of the Gaussian model can be found by maximising the likelihood $\mathcal{L}(\boldsymbol{\mu}, \mathbf{\Sigma})$ over the training dataset. This likelihood is maximised when the parameter values are evaluated according to Equations (11) and (12):

$$\hat{\boldsymbol{\mu}} = \frac{1}{|\mathcal{DS}_T|} \sum_{\mathbf{x}_i \in \mathcal{DS}_T} \mathbf{x}_i \tag{11}$$

$$\hat{\mathbf{\Sigma}} = \frac{1}{|\mathcal{DS}_T|} \sum_{\mathbf{x}_i \in \mathcal{DS}_T} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T \tag{12}$$

The learning process in this case is computationally inexpensive. For the mixture of Gaussians, the analytical expression for the parameter values maximising the likelihood is not known. However, these parameters can be found efficiently by employing the EM algorithm. The learning process using EM algorithm is more computationally demanding, as a number of interactions should be done before the algorithm converges. The classification process is relatively simple; the only computationally expensive operation is the inversion of the covariance matrix.

The methods based on Gaussian models are sensitive to the noise present in the training data, as this noise may introduce a significant bias to the estimated covariance matrix [9]. The accuracy, with which the density is estimated, and, hence, the accuracy of classification depends on whether the data follows the assumed distribution [17]. Besides, these methods are relatively sensitive to the outliers [17]. Lauer [36] developed a method that tolerates a small number of errors in the training set; however, this method requires the proportion of the outliers in the training set to be known in advance.

The storage space required for classification is negligible as only the parameters of the models need to be stored. The storage requirements for learning the models, however, are much higher since all the data from the training dataset are used. In order to minimise the space requirements, the Gaussian model parameters can be evaluated incrementally. In the case of incremental learning, only few observation vectors are needed at each learning step in order to update the parameter values. Modifications of the EM algorithm supporting incremental learning [37] can be used to reduce the storage requirements of the learning of the Gaussian model.

The bias and the variance of the Gaussian and mixture of Gaussians models depend on whether the data follows the assumed distributions. In general, the mixture of Gaussians model is more flexible, and therefore, is expected to have a lower bias error but a higher variance error value.

In the security domain, the Gaussian model was used to detect anomalies in Solaris OS audit events [38]. As a distance function (inverse to the discriminant function), the Hotelling T^2 statistics was employed representing a statistical distance from observation vector $\mathbf{x}$ to the mean of the multivariate Gaussian distribution. The use of a mixture of Gaussians model was reported successful in detecting anomalies in the parameters of user requests to a CORBA-server [39]. Mixture of Gaussians was also employed by [28] in order to model the distribution of continuous variables for anomaly detection.

3.1.4 Parzen density estimation

Parzen density estimation does not make any specific assumptions about the shape of the data distribution. The density is estimated directly from the training data and is a function of the number of observation vectors situated in a region of a specified volume [7]:

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{V}\varphi(\frac{\mathbf{x}-\mathbf{x}_i}{h}), \tag{13}$$

where $N = |\mathcal{DS}_T|$ is the size of the training data-set, and $V = h^{n_f}$ is the volume of the region in a form of the n_f-dimensional hypercube with the length of an edge h. The value of h plays the role of a smoothing parameter and it should be provided in advance. The kernel function $\varphi(\mathbf{v})$ is a window function that should satisfy $\varphi(\mathbf{v}) > 0$ and $\int \varphi(\mathbf{v})d\mathbf{v} = 1$. The Parzen window is defined as:

$$\varphi(\mathbf{v}) = \begin{cases} 1, & \text{if } |v_j| < 1/2,\ j = 1,\ldots,n_f, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Another commonly used kernel (window) function is a multivariate Gaussian, for which:

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi h^2)^{n_f/2}}\exp(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2h^2}). \tag{15}$$

Advantage of the method is its ability to approximate arbitrary distribution, whose parametric form is unknown. The learning phase is trivial: since no parameters need to be estimated, the learning phase consists in storing the values of the observation vectors.

The method however requires the smoothing h parameter to be specified carefully. Too large values of the parameter result in an over-smoothed estimated density. On the other hand, when too small h value is provided, the estimated density contains noise, i.e. it reflects the peculiarities of the training dataset rather than the characteristics of the distribution being estimated [6].

Another drawback of the method is the need to store all the observation vectors. This makes the estimation of probability slower. Besides, if the number of observation vectors to be stored is large, the storage space consumed may become prohibitive. This problem can be partly solved by reducing the number of kernel functions and adapting their widths according to the data [6].

The Parzen density estimation is relatively robust to the outliers in training data since they influence only density estimation in the regions of close proximity [9]. The sensitivity to the noise in data depends on how well the smoothing parameter h is selected – as mentioned above, too low values of h make the estimation noise-sensitive.

In the domain of anomaly intrusion detection, this method of density estimation was employed by [40] in order to estimate the PDF of features extracted from TCP/IP packets. As reported by the authors, the detection accuracy obtained in the experiments using KDD Cup 1999 dataset was comparable to the accuracy of best competitors.

3.1.5 K-nearest-neighbours

K-nearest-neighbours method is similar to the kernel-based estimation discussed above. The probability density is also calculated based on the number of observation vectors in a region of a certain volume. However, in K-nearest-neighbours, the number of observations K is fixed in advance, and the volume of the area is allowed to grow so that K nearest observation vectors would be included in it [7]. (Contrary, the kernel-based estimation assumes the constant volume of the regions and lets the number of observation vectors vary.) Taking the number K as an input smoothing parameter, the method estimates the density as:

$$p(\mathbf{x}) = \frac{K}{N\,V_K}, \tag{16}$$

where V_K is the volume of the smallest area (hypersphere) with the centre in $\mathbf{x}$ surrounding K observation vectors nearest to $\mathbf{x}$.

The advantage of K-nearest-neighbours, similarly to the kernel-based estimation, is its ability to estimate arbitrary distributions. Furthermore, by using varying volume size, K-nearest-neighbours overcomes the shortcoming of the kernel-based estimation, which tends to over-smooth the estimate in the areas of high density, and tends to produce too noisy estimate in the areas of low density [6].

The drawback of this method is the need to keep the observation vectors, in the same way as in the kernel-based density estimation. Furthermore, the produced estimate is not true probability density as its integral over the $\mathbf{x}$ space diverges [6]. This limitation is however compensated by the ability to adjust the area volume to the density of the data.

No parameters need to be learnt by this method. The number of neighbours K, however, needs to be provided, and too great or too low K values result

in over-smoothed or noisy estimated density, respectively. One approach to the problem of selecting the optimal value of K is to assign K the value that minimises error on a separate dataset [7].

Contrary to the kernel density estimator, K-nearest-neighbours method is sensitive to the outliers in the training data. The method's robustness to noise depends on the selection of the smoothing parameter.

In the domain of intrusion detection or fraud detection, no reported research employing K-nearest-neighbours for density estimation has been found.

3.2 Reconstruction methods

This section is devoted to the reconstruction methods wherein assumptions about underlying data structure are made. The well-known methods belonging to this category include K-means, Self-Organizing Maps, Principal Component Analysis, autoencoders, and the classification based on time series analysis, among other methods. In the security domain, also the classification based on association rules has been successfully employed; this classification method also belongs to the category of reconstruction methods. In the following subsections, the description of these methods is provided.

3.2.1 K-means, Self-Organizing Maps

The K-means clustering and self-organizing maps (SOM) are examples of clustering methods assuming that the data is clustered and can be described by a set of prototype (codebook) vectors $\boldsymbol{\mu}_k,\ k = 1, \ldots, K$ [9]. The number K of prototype vectors should be selected beforehand. During classification, the reconstruction error is calculated as

$$\mathcal{E}_{\text{reconstr}} = \min_k ||\mathbf{x} - \boldsymbol{\mu}_k||^2 \tag{17}$$

and compared against a threshold.

The placement of prototype vectors is derived from the training dataset. In K-means clustering, the prototype vectors are selected to minimise [41]

$$\mathcal{E}_{KM} = \sum_{i=1}^{N} \left(\min_k ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2 \right), \tag{18}$$

where $N = |\mathcal{DS}_T|$. Both batch and on-line algorithms can be employed for finding the solution minimising $\mathcal{E}_{KM}$ [6].

In batch algorithm, at each step, the observation vectors are grouped into k disjoint sets $\mathcal{S}_k$ according to the nearest prototype vectors. Then, the prototype vectors are recalculated as:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i \in \mathcal{S}_k} \mathbf{x}_i, \tag{19}$$

where $N_k = |\mathcal{S}_k|$. The procedure is repeated until convergence.

In on-line algorithm, the simple competitive learning [42] is employed, i.e. each subsequent observation vector $\mathbf{x}_i$ is used to update the position of its nearest prototype $\boldsymbol{\mu}_k$:

$$\boldsymbol{\mu}_k(\tau + 1) = \boldsymbol{\mu}_k(\tau) + \eta(\tau)(\mathbf{x}_i - \boldsymbol{\mu}_k), \tag{20}$$

where $\eta(\tau)$ is the learning rate $0 < \eta(\tau) < 1$.

The SOM [42] as well employs competitive learning to define the positions of prototype vectors. Meanwhile, rather than updating only the nearest prototype, the prototypes in neighbourhood of the nearest neighbour are also updated; furthermore, more distant prototypes get smaller update. This neighbourhood is determined by a predefined topology, e.g. by a two-dimensional grid. The prototypes in neighbourhood of $\mathbf{x}_i$ are updated according to:

$$\begin{aligned} \boldsymbol{\mu}_k(\tau + 1) &= \boldsymbol{\mu}_k(\tau) + \\ &+ \eta(\tau) f_{\text{wind}}(|\mathbf{x}_i - \boldsymbol{\mu}_k|)(\mathbf{x}_i - \boldsymbol{\mu}_k). \end{aligned} \tag{21}$$

The function $f_{\text{wind}}(|\mathbf{x}_i - \boldsymbol{\mu}_k|)$ is a window function; it is equal to 1 when $\mathbf{x}_i = \boldsymbol{\mu}_k$, and declines as the value of $|\mathbf{x}_i - \boldsymbol{\mu}_k|$ grows [7].

The above clustering methods are sensitive to remote outliers, since they may bias heavily the placement of the prototype vectors. Meanwhile, the noise present in training data may be compensated by a high number of observation vectors within each cluster; in this case, the noise has little effect on the prototype vector estimation.

Both methods require the number K of clusters to be provided. The K-means clustering is sensitive to the correctness of K: if K differs from the real number of clusters in the data, then wrong clusters may be produced by the method [41]. In turn, the SOM is sensitive to the topological assumptions determining the neighbourhoods [6].

Both clustering methods are computationally light [42]. A small memory space is needed in order to store the prototype vectors. When on-line algorithms are used in clustering, the memory requirements are also very conservative.

In anomaly intrusion detection, the K-means clustering was employed by [43] for clustering networked computers into "activity groups", and by [44] for reducing (compressing) the raw data to a manageable size. The SOM were used e.g. for clustering calling data in detecting telecommunications frauds by [45].

3.2.2 Principal Component Analysis

The principal component analysis (PCA) is aimed at explaining the internal variance and covariance structure of n_f-dimensional data in terms of the set of variables (principal components), which are linear combinations of the original variables. The principal components x'_i represent the projection of the original variables to the eigenvectors $\mathbf{e}_i = (e_{i1}, \ldots, e_{in_f}),\ i = 1, \ldots, n_f$ of the covariance matrix $\boldsymbol{\Sigma}$ ordered according to the decreasing eigenvalues λ_i of the covariance matrix [7]:

$$\begin{aligned} x'_i &= \boldsymbol{e}_i^T(\boldsymbol{x} - \boldsymbol{\mu}) \\ &= e_{i1}(x_1 - \mu_1) + \ldots + e_{in_f}(x_1 - \mu_{n_f}). \end{aligned} \tag{22}$$

The greater the eigenvalue of the component, the higher is its variance. By employing only q eigenvectors with highest eigenvalues, the set of variables can be transformed to the set with lower dimensionality $n'_f = q \leq n_f$ preserving most of the variance of the data.

In order to implement one-class classification, the reconstruction error may be calculated as the Mahalanobis distance from the observation vector to its mean in the transformed space [9, 46]:

$$\mathcal{E}_{\text{reconstr}} = \sum_{i=1}^{n_f} \frac{x'_i}{\lambda_i} \tag{23}$$

Empirical distribution of this error can be used to select the threshold for classification.

Alternatively, only q first principal components can be used for error calculation thus reducing the dimensionality of the data [46]:

$$\mathcal{E}_{\text{reconstr}} = \sum_{i=1}^{q} \frac{x'_i}{\lambda_i}. \tag{24}$$

Shyu et al. [46] applied this approach to analyse TCP packets for anomaly intrusion detection. In their experiments, they managed to reduce the dimensionality of the data from 34 original features to 5 major components.

The use of the PCA as one-class classifier is justified when the dimensionality of the data analysed by the classifier is high. Using the PCA, the computational complexity of classification may be decreased [47]. The only parameter that may need to be specified in advance is the number q of principal components. The PCA is, however, sensitive to the noise and outliers in training data, since they may distort the estimation of variances and covariances. The use of the PCA may be also difficult due to the course of dimensionality as the number of parameters to be estimated is relatively high ($n_{\text{param}PCA} = n_{\text{param}G} + n_f$).

3.2.3 Autoencoders

Neural networks are networks of interconnected processing units arranged in one or several layers that can be used to implement a complex functional mapping between input and output variables. Linear or nonlinear transformations can be performed by processing units at different network layers. The parameters of these units (e.g. weights) are adjusted using training data so that an error function would be minimised over the training set.

The autoencoders, also referred to as autoassociators, are special type of neural networks, which are trained to reproduce their input features at their output [48]. It is assumed that the autoencoder learns the internal structure of the data. In classification, only the observation vectors whose structure is similar to the structure learnt by the network, are reproduced by the autoencoder accurately.

The reconstruction error is calculated as the distance between the network output f_{auto} and network input $\mathbf{x}$:

$$\mathcal{E}_{\text{reconstr}} = ||f_{\text{auto}}(\mathbf{x}) - \mathbf{x}||^2. \tag{25}$$

The advantage of the autoencoders is their flexibility allowing a variety of functional mappings to be represented. This flexibility is used to automatically learn and verify the structure of the data. The autoencoder with single hidden layer and q linear transformation units implements projection onto q-dimensional space spanned by first q principal components, and therefore can be used, similarly to PCA, for data dimensionality reduction.

Being neural networks, autoencoders are sensitive to outliers [6] as they may largely contribute to the error function being minimised. In order to mitigate the negative influence of these errors, the Minkowski-R error with $R < 2$ may be employed. In order to avoid oferfitting to noisy training data, additional regularisation component is included in the error function, or the training is stopped when the error on a separate validation dataset starts to grow [6].

The shortcoming of autoencoders is the need to select a number of parameters that should be specified by the user [9]. These include the number of hidden layers N_{hl} and the number of hidden units N_{hu} at each layer, the type of transformation function, the learning rate, and the stopping rule. Furthermore, a number of weights (usually equal to the number of hidden and input units) need to be estimated using training set. Consequently a large amount of data should be available for this estimation to be accurate.

During training phase, the complete training dataset is used for estimating the weights of the autoencoder. Therefore, the memory space required for training may be not negligible. The computational complexity of training is also high, since the learning process iterates over the training dataset several times until the stopping rule is satisfied; Hinton [49] estimates the learning complexity for neural networks to be approximately $O(N_{\text{w}}^3)$, where N_{w} is the number of weights in the network. The computational complexity and storage requirements of classification, however, are conservative.

In the domain of intrusion detection, autoencoders have been successfully employed to detect anomalies in TCP/IP traffic [50].

3.2.4 Classification based on association rules

Association rules [51] are aimed at modelling the correlation between different symbolic features. Although the methods of mining association rules exist that deal with numeric features, these methods are not considered in this paper.

An association rule is an implication in a form $r : r^A \rightarrow r^C$, where r^A and r^C are boolean predicates, referred to as antecedent and consequence of the rule, respectively. An observation vector $\mathbf{x}$ satisfies the rule r, if $r^A(\mathbf{x}) = true$ and $r^C(\mathbf{x}) = true$.

The rules are characterised by their support and confidence values. The support $\sup(r^A)$ of antecedent A is the fraction of vectors in the set $\mathcal{DS}_T$, satisfying

the antecedent condition. In turn, the support of the rule r is the fraction of the vectors satisfying the rule:

$$\sup(r^A \to r^C) = \frac{1}{|\mathcal{DS}_T|} \times \times |\{\mathbf{x} \in \mathcal{DS}_T : r^A(\mathbf{x}) = true, r^C(\mathbf{x}) = true\}|. \quad (26)$$

The confidence of the rule is defined as

$$\mathrm{conf}(r^A \to r^C) = \frac{\sup(r^A \to r^C)}{\sup(r^A)}. \quad (27)$$

For example, the rule $r_m : x_i = 0 \to x_j = 0$, [conf=0.35, sup=0.05] indicates that if feature i has the value $x_i = 0$, then the feature j has the value $x_j = 0$ in 35% of cases. In 5% of the observation vectors, the features i and j are $x_i = x_j = 0$.

In order to mine association rules, the Apriori algorithm [52] can be employed. This algorithm consists of two parts. First, all the frequent itemsets, i.e. observation vectors with support above predefined minimum support values, are found. After that, the found frequent itemsets are used for generating association rules with confidence above or equal to predefined minimum confidence value. In order to determine the relevant (or "most interesting") rules, the rules known to be redundant or irrelevant are filtered, and the remaining rules are prioritised according to a suitable interestingness measure [53, 54, 55].

In classification, the confidence and the support of the rule are used. The support of the rule can be employed to implement the discriminant function in a form

$$u_{AR}(\mathbf{x}, \mathcal{RS}) = \max_{m \in \mathcal{RS}} \sup(r_m) : r_m^A(\mathbf{x}) = \text{true},\ r_m^C(\mathbf{x}) = \text{true}, \quad (28)$$

where $\mathcal{RS}$ is the set of rules. Thus, the term $1/u_{AR}(\mathbf{x}, \mathcal{RS})$ can be seen as the value of the reconstruction error $\mathcal{E}_{\text{reconstr}}$.

Similarly, the confidence of the rule may be used to produce the reconstruction error as e.g.

$$\mathcal{E}_{\text{reconstr}}(\mathbf{x}, \mathcal{RS}) = \max_{m \in \mathcal{RS}} \mathrm{conf}(r_m) : r_m^A(\mathbf{x}) = \text{true},\ r_m^C(\mathbf{x}) = \text{false}. \quad (29)$$

The confidence value in the expression above is an approximation of probability $P(r_m^C(\mathbf{x}) = \text{false} | r_m^A(\mathbf{x}) = \text{true})$. Thus, this function reflects the degree of anomaly due to the fact that a rule is not satisfied.

The robustness of the association rules to the mislabelling errors is regulated by the values of minimum confidence and minimum support that are used during rule mining; when higher values of these parameters are used, the sensitivity to the errors is reduced.

The minimum confidence and minimum support values are to be specified by the user. Besides, the type of interestingness measure should be defined. The number of parameters to be estimated during rule mining depends on the number of rules: $n_{\text{param}AR} = 2|\mathcal{RS}|$.

The rule mining process is both computationally expensive and extremely memory consuming since the general problem of mining these rules is NP-complete [56].

Association rules were employed in several approaches to intrusion detection, where they were used to model the regularities of normal network traffic [57, 58], and to address the issue of alarm correlation [59].

3.2.5 Time series analysis

A time series can be defined as an ordered finite set of numerical values of a variable of interest along the time axis [60]. In this subsection, for simplicity, we shall consider the one-dimensional variable x. An observation vector is formed by the consecutive values of $x_k,\ k = 0, 1, \ldots, n_f - 1$ that produce univariate time series. It is assumed that consequent x_k are measured at equal time intervals.

Time series analysis models the correlations between elements in time series. A number of models representing the value of x at time k have been proposed [61], e.g. autoregressive models, moving average model, and autoregressive-moving-average model. Here, we consider the autoregressive (AR) model representing the value of x at time k as a function of immediate past values of x along with random error. The order of the AR model reflects how many lagged past values are included. The p-order AR model, for example, is defined as:

$$x_k = \sum_{i=1}^{p} \theta_i x_{k-i} + e_k, \quad (30)$$

where θ_i are the parameters of the model referred to as autoregressive coefficients, and e_k denotes the error (residual), which is assumed independent and normally distributed. The values of autoregressive coefficients are estimated by minimizing the sum of residuals $\sum_{\mathcal{DS}_T} e_k$ over the training set.

Having learnt the parameters of the model, the reconstruction error for a new observation x_{k+1} is calculated as

$$\mathcal{E}_{\text{reconstr}} = x_{k+1} - \sum_{i=1}^{p} \theta_i x_{k+1-i} \quad (31)$$

The calculated values of reconstruction error can be used to estimate empirically the threshold t for classification. In some models, however, the distribution of x_{k+1} can be assumed; in these cases, the statistical upper and lower control limits can be determined directly and used as thresholds.

By incorporating the residual error component, the models used in time series analysis are able to tolerate some noise in the training data. The mislabelled data, however, may distort heavily the estimated parameters of the models. The accuracy of the model (in terms of the reconstruction error) greatly depends on whether the selected model structure reflects the real temporal structure of time series.

Besides fixing the number of lagged values, no other parameters should be specified for the models. The number of parameters being estimated depends on the specific model; e.g., for the p-order AR model, $n_{\text{param}ARM} = p$.

For some models (e.g. for the AR model) the parameters can be estimated on-line; the storage requirements and the computational complexity of learning for such models are negligible. If the parameters are estimated using batch algorithms, the storage requirements may become prohibitive, since all the training data needs to be stored. The classification phase involves simple computations, and, hence, is inexpensive both from the point of view of processor load and memory consumption.

The time series analysis has been employed in anomaly intrusion detection by [62, 63] to reveal temporal regularities in audit trails collected by the BSM of Solaris OS.

3.3 Boundary methods

This section considers the boundary methods, in which a boundary is built around the training data, and the classification is based on the calculated distance between an observation vector and the boundary. The methods considered in this subsection include a simple representative boundary method (K-centres), and the methods derived from the well-known Support Vector Classifier (Support Vector Data Description and ν Support Vector Classification).

3.3.1 K-centres

K-centres is a simple boundary method, covering the training dataset with K smallest hyperspheres of equal radii [64]. The centres of hyperspheres are placed on some of the observation vectors from the training dataset so that the following function would be minimised [9]:

$$\mathcal{E}_{KC} = \max_{i=1}^{N} \left(\min_{k=1}^{K} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2 \right). \tag{32}$$

The solution minimising this error function can be found using the batch algorithm described above for K-means clustering.

During classification, the distance between new observation vector and found centres is calculated:

$$d_{KC} = \min_{k=1}^{K} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2. \tag{33}$$

The method is highly sensitive to noise and outliers, because the radius of the hyperspheres is determined by the maximum distance to the covered vectors. As a result, the centres of hyperspheres may be distorted by the noisy or mislabelled data [9].

The method requires the number K of the centres to be provided. The number of parameters to be determined using training set is equal to K. Similarly to K-means clustering, the classification with K-centres is computationally simple. It may however require a significant memory space for training due to the need to store all the observation vectors from the training dataset.

Lane and Brodley [65] employed K-centres as a base classifier and compared it with greedy clustering technique for data reduction in anomaly intrusion detection.

3.3.2 Support Vector Data Description and ν Support Vector Classification

Support vector data description (SVDD) is aimed at defining the hypersphere with a minimum volume (i.e. minimum radius) encompassing/covering the entire training dataset [66, 9, 10]. The SVDD is a special case of support vector classifier [67].

During the training of the SVDD classifier, the parameters α_i are estimated by minimising

$$L = \mathcal{E}_{SVDD} = \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j), \tag{34}$$

subject to the constraints $\sum_i \alpha_i = 1$ and $0 \le \alpha_i \le C$, where C determines the number of vectors that will not be covered by the description.

The minimisation of the above equation is solved as a quadratic programming problem. In order to provide more flexible descriptions, the inner products $(\mathbf{x}_i \cdot \mathbf{x}_j)$ can be substituted with a kernel function $K(\mathbf{x}_i \cdot \mathbf{x}_j)$ [67]. The kernel function, e.g. polynomial or Gaussian, transforms the vectors to a higher dimensional feature space where a more accurate description can be produced.

In classification, the distance from new observation vector $\mathbf{x}$ to the centre of the hypersphere is calculated and compared against its radius:

$$\gamma(\mathbf{x}) = \begin{cases} C_1, & \text{if } ||\mathbf{x} - \mathbf{a}||^2 \le R^2, \\ C_2, & \text{otherwise.} \end{cases} \tag{35}$$

Here, $\mathbf{a}$ designates the centre of the hypersphere, and it is estimated as $\sum_i \alpha_i \mathbf{x}_i$. The radius R is calculated as

$$R = (\mathbf{x}_k \cdot \mathbf{x}_k) - 2 \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_k) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j), \tag{36}$$

where $\mathbf{x}_k$ are the vectors, which have $\alpha_k < C$.

Scholkopf et al. [68] propose a similar solution to the data description problem called as ν support vector classification (ν-SVC). Instead of hypersphere, a hyperplane is used to separate the data in training dataset from the origin with a maximum margin. The parameters α_i are found as a solution to the minimisation problem:

$$\min_{\alpha_{i,j}} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$
$$\text{subject to} \quad 0 \le \alpha_i \le \frac{1}{N\nu}, \ \sum_i \alpha_i = 1, \tag{37}$$

where ν plays the role of regularization term, similar to C above. The classification is performed using the

distance between a new observation $\mathbf{x}$ and the origin:

$$\gamma(\mathbf{x}) = \begin{cases} C_1, & \text{if } \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho \geq 0, \\ C_2, & \text{otherwise.} \end{cases} \quad (38)$$

ρ is calculated as $\sum_j \alpha_j K(\mathbf{x}_j, \mathbf{x}_i)$, where $\mathbf{x}_i$ is an observation vector, for which α_i is not at the lower or the upper bound.

These methods are relatively resistant to the noise and mislabelling errors. By adjusting the regularization parameters (C or ρ), the noisy and mislabelled vectors can be excluded from the produced description [9].

The regularization parameter is the only one that needs to be provided by the user. Additional parameters may, however, be needed for the kernel functions employed. The number of parameters that are to be estimated by the quadratic optimization procedure is equal to the size of the training set N. This may preclude the applicability of the method when the size of the training set is large [10].

The classification using these methods is computationally simple and does not require significant memory. Meanwhile, the training procedure is computationally expensive, since standard algorithms for solving quadratic programming problem have the time complexity of order $O(N^3)$ [69]. This complexity, however, may be reduced, if the simplicity of the employed constraints is taken into account in the design of the optimisation routine [69]. A large storage space may be needed at the training phase, since all the observation vectors from the training dataset are used in the optimisation.

In the domain of anomaly intrusion detection, the ν-SVC was employed by [70] for analysis of the user commands, by [71] for analysis of the access to Windows Registry, and by [72] and [73] for network traffic analysis.

3.4 Summary

The characteristics of the reviewed classification methods including sensitivity to the errors in training data, the computational and memory requirements, and the number of parameters, are summarised in Table 1.

The method's sensitivity to errors as well as computational and memory requirements are expressed in the table using an ordinal scale with four fuzzy values including (from better to worse): very low (VL), rather low (RL), rather high (RH), and very high (VH). Whenever both an incremental and batch algorithm for training a classifier are available, the computational and storage requirements for this classifier are described assuming that the incremental algorithm is employed for training.

In enumerating the parameters set by the user, only numerical parameters were taken into account, while other problem-related parameters specifying the structure of the employed model (e.g. the number of hidden layers in an autoencoder or the number of observable symbols in the HMM) were ignored.

4 SELECTING CLASSIFIERS FOR MOBILE-MASQUERADER DETECTION

In the process of selecting appropriate one-class classifier(s) for a particular application domain, two aspects should be taken into account, among other things:

- The type of the features which the classifier takes as input;
- The internal characteristics of the classification method used.

The peculiarities of the application domain often restrict the type and the characters of available features; furthermore, the application domain may impose specific requirements to the computational demands, robustness, and other properties of the classification methods to be employed. Therefore, the systematic analysis and juxtaposition of the properties of one-class classifiers, described in Section 3 and summarised in the Table 1 above, may be used to facilitate identifying the subset of candidate classifiers, which are suitable for the domain in question. Below, the selection of appropriate one-class classifiers for the domain of mobile-masquerader detection is considered.

The selection of classifiers to be employed in mobile-masquerader detection should take into account the constraints imposed by the limited resources of mobile terminals including limited computational power and storage space as well as limited battery power. These constraints reduce the number of classifiers that may be employed.

In particular, the HMM classifier, SVDD, ν-SVC, autoencoders, and the classifier based on the association rules are computationally expensive and therefore, may be prohibitive for resource-constrained terminals. The Parzen density estimation and K-nearest neighbour estimation requires all the training data to be present at the time of classification; this may be problematic if an extensive training dataset needs to be stored.

Besides the peculiarities of the terminals, the suitability of a classification method is determined by the characteristics and features being analysed. Some of the classification methods involve a bias in a form of specific assumptions about the distribution of the data or about the data-generation process. The use of these methods may therefore be sub-optimal in situations, where these assumptions do not necessarily hold. Specifically, the Gaussian and the mixture of Gaussians methods assume that the data follow normal distribution; similarly, the SOM assumes the underlying topology in data.

4.1 Applicability of classification methods to mobile-masquerader detection

Thus, in order to define suitable classification methods, it is necessary to consider the characteristics and features (measures) to be analysed. Table 2 provides the list of characteristics and features that are hypothesised to be useful in mobile-masquerader detec-

Table 1: Characteristics of one-class classification methods (ordinal scale is used with four fuzzy values: VL – very low, RL – rather low, RH – rather high, and VH – very high)

Classification method	Sensitivity to		Computational requirements of		Storage requirements		Number of parameters	
	errors in feature values	outliers	learning	classification	learning	classification	set by the users	learnt
Density methods								
Histograms	RL	RL	VL	VL	RL	RL	1	N_{bins}
Conventional Markov models	RL	RL	VL	VL	RL	RL	0	N_s^2
HMM	RL	RL	VH	RH	VH	RL	1	$N_s^2 + N_s \times (N_v - 1)$
Gaussian, mixture of Gaussians	RH	RH	RH	RL	RH	VL	1	$O(n_{MG} \times n_f^2)$
Parzen density estimation	RL	RL	VL	RH	RH	RH	1	0
K-nearest neighbours	RL	RH	VL	RL	RH	RH	1	0
Reconstruction methods								
K-means, SOM	RL	RH	VL	VL	VL	VL	1	K
PCA	RH	RH	RL	VL	RL	VL	1	$O(n_f^2)$
Autoencoders	RL	RH	VH	VL	RH	RL	1	N_{w}
Assoc. rules	RL	RL	VH	RL	VH	RL	2	$2\|\mathcal{RS}\|$
Time series analysis	RL	RH	VL	VL	VL	VL	1	p
Boundary methods								
K-centres	VH	VH	RH	VL	RH	VL	1	K
SVDD, ν-SVM	RL	RL	VH	VL	RH	VL	1	N

tion [4]. Below, the classifiers' suitability for processing the proposed features is considered.

For each characteristic and assigned features, the above table also indicates the type of feature values (numeric or symbolic) and whether the temporal order of measurements is important or not. Depending on the feature type and on the importance of temporal relations, the following four categories of characteristics can be distinguished:

- **Numeric features, temporal relations ignored.** These features include the interval between consecutive program or service evocations; the temporal length and interval between actions; the speed of move; the time spent at a workplace; the time of reading a textual document; the frequency of errors; the time spent for communications; the time when the communication facilities are enabled; and keystroke duration and latency times.

  A number of classification methods can be applied to these features (cf. Figure 1), including Gaussian and mixture of Gaussians, K-nearest neighbours, Parzen estimator, K-means, SOM, PCA, autoencoders, K-centres, SVDD and ν-SVC.

  The K-nearest neighbours, Parzen estimator, K-means, SVDD, and ν-SVC can be assigned to the features for which the training set is of limited size. The Gaussian and the mixture of Gaussians can be employed as well; however, their suitability depends on whether the distribution of feature values follows the assumed distribution. For the features with more extensive training sets available (e.g. keystroke duration and latency times) the autoencoders may be a good choice. Furthermore, the keystroke duration and latency times produce the feature vector of relatively high dimensionality; therefore, they can be processed using the PCA or the SOM reducing the dimensionality of the feature vector. Meanwhile, due to high computational complexity, the use of autoencoders, as well as SVDD and ν-SVC may consume too heavily the resources of a mobile terminal.
- **Numeric features, temporal relations are important.** The features describing changes in behaviour and environment, the features describing statistical characteristics of voice, and the features reflecting the characteristics of strokes produced with stylus belong to this category.

  Among the classification methods shown in Figure 1, only time series analysis is capable of processing this type of features. Besides, a modification of HMM where the output probabilities in each state are modelled by an auto-regressive process [74] can be applied to these features.
- **Symbolic features, temporal relations ignored.** This category encompasses the following features: the type of programs/services being evoked; the use of shortcut vs. the use of menu; the phone numbers and the email addresses of people contacted with; the places where stops are made; and the frequency of different words used.

  The histograms and the classifier based on association rules can be employed for processing these features. However, since the mining of association rules is computationally- and memory-demanding, the application of association rules in the context of mobile terminals may be problematic. Therefore, histograms are the main candidate for analysing this type of features.
- **Symbolic features, temporal relations are important.** This category is comprised of two features: the sequence of consecutive actions and the sequence of traversed cells.

Table 2: Tentative characteristics and features to be employed in mobile-masquerader detection, according to [4]

Characteristic	Feature	Type	Temporal order
Device's facilities usage	Temporal interval between two consecutive evocations of a program or service of a same type.	Real	No
Device's facilities usage	Type of program or service evoked	Symbolic	No
Sequences of actions followed	Sequences of n actions	Symbolic	Yes
Temporal lengths of actions	Temporal lengths of actions	Real	No
Temporal intervals between actions in a sequence	Temporal intervals between subsequent actions	Real	No
Use of shortcuts vs. use of menu	For each menu command with shortcut, the chosen option	Symbolic	No
People contacted with, conditioned on type of communication, time, etc.	Phone number, e-mail address, or other address information of the contacted people	Symbolic	No
Routes taken	Sequence of cells traversed between two consecutive prolonged stops	Symbolic	Yes
Speed of move conditioned on route and time	Speed of move conditioned on route and time	Real	No
Places visited, conditioned on time of day, day of week, etc.	Locations where prolonged stops were made	Symbolic	No
Length of work day	Time that the terminal is in the place affiliated with the user's workplace(s)	Real	No
Changes in behaviour and environment	Changes in behavioural and environmental characteristics	Real	Yes
Time of reading a unit of textual information	Time during which a document is open for reading	Real	No
Time between an incoming event and response	Temporal interval between an incoming message (e.g. e-mail) is read and the response is written	Real	No
Words or phrases used more often	Frequency of different words used in handwriting (with stylus) or typing	Symbolic	No
Accuracy in typing, in menu item selection, etc.	The ratio of errors to the overall number of actions, i.e. the frequency of mistyped keystrokes, errors in menu item selection, etc.	Real	No
Time devoted to communication	Time during a day spent for communication (using terminal) including different types of communication (calls, e-mails, etc.)	Real	No
Time, when the user is online	Time, during which the communication facilities of the terminal are not deliberately restricted	Real	No
Statistical characteristics of voice	Cepstrum coefficients of the signal power	Real	Yes
Temporal characteristics of keystrokes	Key duration time, inter-key latency time	Real	No
Pressure, direction, acceleration, and length of strokes when stylus is used	Pressure, direction, acceleration, and length of strokes	Real	Yes
Set of installed software, current screen resolution, volume level	Changes of device configuration	Symbolic	No

According to Figure 1, these features may be analysed with conventional or hidden Markov models. Since for both features, the states (the user actions and the traversed cells) are directly observable, the use of conventional Markov model is justified.

Above, various features to be used in masquerader detection were mapped to the one-class classification methods reviewed in this paper. For the first category of features, several potentially suitable classifiers were identified while for other categories only one classifier was found. Since the review of classifiers provided in this paper, is not exhaustive, there may be also other one-class classifiers suitable for processing these features. In the next subsection, for three of four categories above, the designs of classifiers to process features in these categories are exemplified.

4.2 Building classifiers for mobile-masquerader detection

Recently, a dataset describing the behaviour and environment of several mobile users was collected at the University of Helsinki. The dataset was gathered in the course of two field studies aimed at testing social awareness service named ContextContacts [75]. Two groups of respectively four and five users living in the greater Helsinki area participated in the studies for approximately three months, and their behaviour and environment were monitored using the ContextPhone software platform [76] running in the background on Nokia Series 60 smart-phones. The data collected with this software reflects changes of GSM Cell IDs wherein the terminal is registered, the use of applications, active profile, the use of charger, idle and active time, Bluetooth environment, and communication events (SMS and calls). An anonymized version of this dataset can be found at `http://www.cs.helsinki.fi/group/context/data/`.

In the above dataset, some of the features proposed in Table 2 are available; these include:

- *Type of program or service evoked.* Active applications evoked by the user are registered in the dataset.
- *Sequence of cells traversed.* The dataset records the identifiers of the cells (Cell IDs) wherein the mobile terminal is registered.
- *Speed of move.* Though the speed of movements is not available in the dataset, the timestamps of the Cell ID records can be used to estimate the time the terminal spends in a cell, which, in turn, can be used to roughly estimate the terminal's speed in terms of "cell per second".

- *Locations where prolonged stops were made.* The information about the Cell IDs and the time spent in cells can be used to identify those locations (in terms of Cell IDs) where the terminal stays for a relatively long period of time.
- *Temporal interval between two consecutive evocations of a program or service of a same type.* In the dataset, the evocations of two services (calls and SMS) are recorded and time-stamped; using this information, the intervals between evocations of these services can be evaluated.
- *Temporal lengths of actions.* In the dataset, the durations of calls are recorded.
- *Address information of the contacted people.* Phone numbers contacted via calls or SMS are available in the dataset. Besides, the identifiers (MAC-addresses) of neighbouring Bluetooth-devices are logged.

These features belong to three categories introduced above: numeric features, temporal relations ignored (temporal lengths of actions, temporal interval between two consecutive evocations of a program or service of a same type, speed of move); symbolic features, temporal relations ignored (type of program or service evoked, locations where prolonged stops were made, address information of the people contacted); and symbolic features, temporal relations are important (sequence of cells traversed). In the remainder of this Section, for each of these three categories, a design of a classifier to process the features belonging to this category is described; further details of these classifier can be found in [77]. For all classifiers, their observation vectors are initialised with feature values by using a sliding window $[\tau_1, \tau_2]$ of the length $l_\tau = \tau_2 - \tau_1$ (determining the time interval, within which the feature values are collected) and the increment for the window δ_τ.

Numeric features, temporal relations ignored. For the features in the first category, several classifiers were identified in the above analysis as potentially suitable; among them, K-nearest neighbours, Parzen estimator, K-means, and K-centres were preferred for the terminals with restricted computational capabilities. The K-means classifier requires a careful selection of the parameter K, which may be problematic for a mobile-terminal user. The K-centres is sensitive to the noise and outliers in the training dataset. In turn, the Parzen estimator tends to produce over-smoothed or too noisy density estimation. Therefore, the K-nearest neighbours classifier is chosen as the classifier for the features in this category. The design of this classifier is described below for the example of the "temporal lengths of actions" feature.

The classifier analyses the mean call duration time $\overline{\tau}^{\mathrm{dur}}$ within the window $[\tau_1, \tau_2]$. The value of $\overline{\tau}^{\mathrm{dur}}$ is calculated as $\overline{\tau}^{\mathrm{dur}} = \frac{1}{n_d} \sum_{j=i-n_{\mathrm{dur}}+1}^{i} \tau_j^{\mathrm{dur}}$, where τ_i^{dur} is the last call duration registered within the window, and n_{dur} is the average number of calls finished within a window. Using the accumulated empirical distribution function (EDF) of the $\overline{\tau}^{\mathrm{dur}}$ values, the probability density $p(\overline{\tau}^{\mathrm{dur}})$ of the current mean inter-arrival time is evaluated using k-nearest-neighbours method. Given the current inter-arrival time values, the classifier outputs the classification $u_i = p(\overline{\tau}^{\mathrm{dur}})$.

Symbolic features, temporal relations ignored. For the features in the second category, histograms were deemed appropriate. Below, an application of this classifier to one of the features in this category (type of program or service evoked) is described.

The classifier assigned to this feature estimates the probability of an application j being evoked out of m applications as $\hat{P}(\mathrm{app}_j|U) = (a_{\mathrm{app}_j} + 1)/(\sum_m a_{\mathrm{app}_m} + 1)$, where a_{app_j} is the number of times the user evokes the application. Assuming the independence of consequent application evocations, the probability of application evocations within a time window $[\tau_1, \tau_2]$ is approximated as $\hat{P}(\mathrm{app}_{i-n_{\mathrm{app}}+1}, \ldots, \mathrm{app}_i|U) = \prod_{j=i-n_{\mathrm{app}}+1}^{i} \hat{P}(\mathrm{app}_j|U)$, where app_i is the last application evoked within the time window, and n_{app} is the average number of applications evoked within the window. Given the current active applications to be classified, the classifier outputs the classification $u_i = \hat{P}(\mathrm{app}_{i-n_{\mathrm{app}}+1}, \ldots, \mathrm{app}_i|U)$.

Symbolic features, temporal relations are important. The conventional Markov model was hypothesized to be suitable for the features in this category. Below, the design of the classifier based on this model is specified for the "sequence of cells traversed" feature.

The model of the assigned classifier includes a matrix, where each element $a_{\mathrm{cell}_i\ \mathrm{cell}_j}$ is a counter that stores the number of times the terminal's Cell ID changed from cell i to cell j. The matrix values are used in approximating the probability $\hat{P}(\mathrm{cell}_j|\mathrm{cell}_i, U)$ of a handover:

$$\hat{P}(\mathrm{cell}_j|\mathrm{cell}_i, U) = \frac{a_{\mathrm{cell}_i\ \mathrm{cell}_j} + 1}{\sum_m a_{\mathrm{cell}_i\ \mathrm{cell}_m} + n_{\mathrm{ct}\ i}}, \qquad (39)$$

where cell_m are the cells to which traversals from cell_i were registered, and $n_{\mathrm{ct}\ i}$ is the number of such cells. Given the parameters l_τ and δ_τ of sliding window, the average number of handovers n_{ho} within a window is estimated. Assuming the independence of consequent handovers, the probability of a sequence of cell changes within a time window $[\tau_1, \tau_2]$ is approximated as $\hat{P}(\mathrm{cell}_{i-n_{\mathrm{ho}}}, \ldots, \mathrm{cell}_i|U) = \prod_{j=i-n_{\mathrm{ho}}}^{i-1} \hat{P}(\mathrm{cell}_{j+1}|\mathrm{cell}_j, U)$, where cell_i is the last cell registered within the time window. In the classification phase, given the current route to be classified, the classifier outputs the classification $u_i = \hat{P}(\mathrm{cell}_{i-n_{\mathrm{ho}}}, \ldots, \mathrm{cell}_i|U)$.

Using the dataset described above, initial experiments with these classifiers were conducted. The hold-out cross-validation [78] was used in the experiments, in order to evaluate the accuracies of classifiers. The model of each classifier was learnt using the training data-set $\mathcal{DS}_T$, and was subsequently used to classify the instances of a test data-set $\mathcal{DS}_C$. Since the data

originated from masqueraders were not available in the dataset, the other users' data were employed as the masquerader's data. The results of these experiments are reported in [77]; according to these results, a relatively good accuracy can be achieved with some of these classifiers, e.g. with the classifiers processing address information of contacted people. Meanwhile, the accuracy of some of the classifiers (in particular, the classifiers processing temporal intervals between consecutive evocations of a program or service of a same type) was very low, indicating that the corresponding features are poor differentiators between users.

According to no-free-lunch theorem [79], no single classification method would provide superior classification accuracy independently of the context, in which the method is applied. Therefore, further empirical studies are needed in order to determine the accuracy, and hence, suitability of different methods in a given context [7]. For example, many of the above methods are sensitive to the noise and mislabelling errors in the training data, and the empirical testing can be employed to evaluate how significant is the influence of such errors for the classification problem in hand.

Besides, in the context of mobile-masquerader detection, these empirical studies should determine whether a classification method fits to the constraints of the mobile terminals. For instance, the above K-nearest neighbours classifier requires the observation vectors to be kept, thereby consuming the memory of a mobile terminal. However, for the available features analyzed by this classifier, the use of memory is rather conservative. For example, the data required for the classifier analyzing durations of calls consumes in compressed format from 7 to 29 kilobytes; furthermore, a significant proportion of this data is redundant and can be excluded in real-world applications. Thus, the amount of data required for the K-nearest neighbours classifier appears to be tolerable for contemporary smart-phones, making the use of this classifier in the context of mobile-masquerader detection feasible.

5 CONCLUSIONS

A noticeable research during last years has been devoted to the problem of one-class classification. The peculiarity of this type of classification is the availability of the observation vectors of only one class during learning. These methods are particularly useful in the application areas, where the observation vectors belonging to other classes are difficult or impossible to obtain. One such application area is mobile-masquerader detection.

A number of one-class classification methods have been proposed in literature, varying from modifications of conventional N-class classification methods to the methods designed specially for one-class classification problem. In this paper, some of these methods have been reviewed, and their suitability to the problem of mobile-masquerader detection has been analysed.

Both the review of classifiers and the analysis of their suitability to a specific application domain were based on the taxonomy of one-class classification methods that was introduced in the paper. This taxonomy categorises the one-class classification methods according to the internal model, the type of the features, and the ability to take into account the temporal relations between the features.

According to the internal model, the classification methods were divided into density, reconstruction, and boundary methods. This division was employed in the review, since the methods within a same category have some similarities in the learning and classification phases. Within each category, the methods were reviewed according to a developed analysis framework. Namely, for each method, the review included the summary of the learning and the classification process implemented by the classifier, its sensitivity to the noise and mislabelling errors in training data, the number of parameters to be set by the user or estimated using training data, and the resources consumed during the learning and classification phases.

The categorisation of classification methods according to the type of data and according to the ability to take into account the temporal regularities was employed in order to evaluate the applicability of the methods being reviewed to the problem of mobile-masquerader detection. For this, the features that are hypothesised to be useful in masquerader detection were divided into four categories, according to the type of features (symbolic or numeric) and according to the importance of temporal ordering of measurements. Then, for each category of features, potentially suitable classifiers were identified, and the design of some of the classifiers has been described in detail. Further empirical research is, however, needed in order to compare different methods quantitatively in the context of mobile-masquerader detection.

ACKNOWLEDGMENTS

This work was partly supported by the COMAS Graduate School of the University of Jyväskylä. The author would like to acknowledge the Context project for the implementation of the ContextPhone platform and making available the dataset on mobile context and communication, and would like to thank Seppo Puuronen as well as anonymous reviewers for valuable comments and suggestions.

REFERENCES

[1] Pointsec Mobile Technologies. "Half of All Corporate PDAs Unprotected Despite Employer Risk". Pointsec News Letter 2, Available from `http://www.pointsec.com/news/mediakit/` (read 09.02.2006), June 2004.

[2] Pointsec Mobile Technologies. "Confidential Data Gets Taken for a Ride". Pointsec News Letter 1, Available from `http://www.pointsec.com/news/mediakit/` (read 09.02.2006), March 2005.

[3] Pointsec Mobile Technologies. "IT Professionals Turn Blind Eye to Mobile Security as Survey Reveals Sloppy Handheld Habits". Pointsec news releases, Available from `http://www.pointsec.com/news/release.cfm?PressId=108` (read 09.02.2006), November 18 2005.

[4] O. Mazhelis and S. Puuronen. "Characteristics and measures for mobile-masquerader detection". In P. Dowland, S. Furnell, B. Thuraisingham and X. S. Wang (editors), *Proc. IFIP TC-11 WG 11.1 & WG 11.5 Joint Working Conference on Security Management, Integrity, and Internal Control in Information Systems*, pp. 303–318. Springer Science+Business Media, 2005.

[5] T. Lane. *Machine Learning Techniques for the Computer Security Domain of Anomaly Detection*. Ph.D. thesis, Purdue University, W. Lafayette, IN, 2000.

[6] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.

[7] R. O. Duda, P. E. Hart and D. G. Stork. *Pattern Classification*. John Wily & Sons, Inc., New York, second edn., November 2000. ISBN 0-471-05669-3.

[8] C. M. Bishop. "Novelty detection and neural network validation". In *IEE Proceedings – Vision, Image and Signal processing, Special Issue on Applications of Neural Networks*, vol. 141(4), pp. 217–222. 1994. URL `http://citeseer.ist.psu.edu/bishop94novelty.html`.

[9] D. Tax. *One-class classification*. Ph.D. thesis, Delft University of Technology, 2001.

[10] D. M. J. Tax and R. P. W. Duin. "Support Vector Data Description". *Machine Learning*, vol. 54, pp. 45–66, 2004.

[11] A. Dempster, N. Laird and D. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[12] W. Fan, M. Miller, S. J. Stolfo, W. Lee and P. K. Chan. "Using Artificial Anomalies to Detect Unknown and Known Network Intrusions". In *Proceedings of the First IEEE International Conference on Data Mining*. 2001. URL `http://www.cc.gatech.edu/ wenke/papers/artificial_anomalies.ps`.

[13] G. L. Peterson, R. F. Mills, B. T. McBride and W. C. Allred. "A Comparison of Generalizability for Anomaly Detection". In D. Margineantu, S. Bay, P. Chan and T. Lane (editors), *International Workshop on Data Mining Methods for Anomaly Detectio*. 2005.

[14] E. Keogh, S. Lonardi and C. A. Ratanamahatana. "Towards parameter-free data mining". In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 206–215. ACM Press, New York, NY, USA, 2004. ISBN 1-58113-888-9. doi: http://doi.acm.org/10.1145/1014052.1014077.

[15] M. Hilario and A. Kalousis. "Characterizing Learning Models and Algorithms for Classification". Deliverables 2.1a, 2.2a, CUI - University of Geneva, March 1999.

[16] M. Markou and S. Singh. "Novelty detection: a review – part 1: neural network based approaches". *Signal Processing*, vol. 83, pp. 2499–2521, 2003.

[17] M. Markou and S. Singh. "Novelty detection: a review – part 1: statistical approaches". *Signal Processing*, vol. 83, pp. 2481–2497, 2003.

[18] L. Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[19] S. Kumar. *Classification and Detection of Computer Intrusions*. Ph.D. thesis, Purdue University, West Lafayette, USA, 1995.

[20] J. Hollmen. *User Profiling and Classification for Fraud Detection in Mobile Communications Networks*. PhD thesis, Helsinki University of Technology, 2000.

[21] V. Poosala, P. J. Haas, Y. E. Ioannidis and E. J. Shekita. "Improved histograms for selectivity estimation of range predicates". In J. Widom (editor), *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pp. 294–305. ACM Press, New York, NY, USA, 1996. ISBN 0-89791-794-4. doi: http://doi.acm.org/10.1145/233269.233342.

[22] A. Aboulnaga and S. Chaudhuri. "Self-tuning histograms: building histograms without looking at data". In S. B. Davidson and C. Faloutsos (editors), *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pp. 181–192. ACM Press, New York, NY, USA, 1999. ISBN 1-58113-084-8. doi: http://doi.acm.org/10.1145/304182.304198.

[23] H. Javits and A. Valdes. "The SRI IDES Statistical Anomaly Detector". In *IEEE Symposium of Research in Computer Security and Privacy*. IEEE Computer Society Press, May 1991. ISBN 0-81862-168-0. URL `http://www.sdl.sri.com/papers/stats91/`.

[24] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, H. S. Javitz, A. Valdes and P. G. Neumann. "A Real-Time Intrusion Detection Expert System". Final technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992. URL `http://www.sdl.sri.com/projects/nides/reports/9sri.pdf`.

[25] T. Lunt. "Detecting Intruders in Computer Systems". In *Conference on Auditing and Computer Technology*. 1993.

[26] H. S. Javits and A. Valdes. "The NIDES Statistical Component: Description and Justification". Technical Report A010, Computer Science Laboratory, SRI International, Menlo Park, California, March 1993. URL `http://www.sdl.sri.com/papers/statreport/`.

[27] D. Anderson, T. Frivold and A. Valdes. "Next-generation Intrusion Detection Expert System (NIDES): A Summary". Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, California, May 1995.

[28] K. Yamanishi, J. ichi Takeuchi, G. Williams and P. Milne. "On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms". *Data Mining and Knowledge Discovery*, vol. 8, no. 3, pp. 275–300, May 2004.

[29] D. L. Isaacson and R. W. Madsen. *Markov chains: theory and applications*. John Wiley & Sons, New York, 1976. ISBN 0-471-42862-0.

[30] J. Campbell, J.P. "Speaker recognition: a tutorial". *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1437–1462, Sep 1997.

[31] C. Warrender, S. Forrest and B. Pearlmutter. "Detecting intrusions using system calls: Alternative data models". In *IEEE Symposium on security and Privacy*, pp. 133–145. IEEE Computer Society Press, 1999.

[32] D.-Y. Yeung and Y. Ding. "Host-based intrusion detection using dynamic and static behavioral models". *Pattern Recognition*, vol. 36, no. 1, pp. 229–243, 2003.

[33] N. Ye. "A Markov Chain Model of Temporal Behavior for Anomaly Detection". In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, pp. 171–174. 2000.

[34] S.-B. Cho and H.-J. Park. "Efficient anomaly detection by modeling privilege flows using hidden Markov model". *Computers & Security*, vol. 22, no. 1, pp. 45–55, 2003.

[35] T. Lane and C. E. Brodley. "An Empirical Study of Two Approaches to Sequence Learning for Anomaly Detection". *Machine Learning*, vol. 51, no. 1, pp. 73–107, April 2003.

[36] M. Lauer. "A Mixture Approach to Novelty Detection Using Training Data with Outliers". In L. D. Raedt and P. Flach (editors), *Proceedings of the 12th European Conference on Machine Learning*, vol. 2167 of *Lecture Notes in Computer Science*, pp. 300–311. Springer-Verlag, Berlin Heidelberg, 2001. ISSN 0302-9743.

[37] R. M. Neal and G. E. Hinton. *Learning in graphical models*, chap. A view of the EM algorithm that justifies incremental, sparse, and other variants, pp. 355–368. MIT Press, 1999. ISBN 0-262-60032-3.

[38] N. Ye and Q. Chen. "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems". *Quality and Reliability Engineering International*, vol. 17, no. 2, pp. 105–112, 2001.

[39] R. Puttini, Z. Marrakchi and L. Me. "A Bayesian Classification Model for Real-Time Intrusion Detection". In *Proc. of 22th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MAXENT'2002)*, vol. 659(1) of *AIP Conference Proceedings*, pp. 150–162. IOP Institute of Physics Publishing Ltd, 2003.

[40] D. Yeung and C. Chow. "Parzen-window network intrusion detectors". In *Proceedings of the Sixteenth International Conference on Pattern Recognition (ICPR)*, vol. 4, pp. 385–388. 2002.

[41] T. Hastie, R. Tibshirani and J. H. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001. ISBN 0387952845.

[42] T. Kohonen. "The self-organizing map". *Proc. of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[43] D. Marchette. "A statistical method for profiling network traffic". In *Proceedings of the Workshop on Intrusion Detection and Network Monitoring*, pp. 119–128. USENIX Association, 1999.

[44] S. Zanero and S. M. Savaresi. "Unsupervised learning techniques for an intrusion detection system". In G. Bella and P. Ryan (editors), *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 412–419. ACM Press, New York, NY, USA, 2004. ISBN 1-58113-812-1. doi: http://doi.acm.org/10.1145/967900.967988.

[45] J. Hollmen, V. Tresp and O. Simula. "A self-organizing map algorithm for clustering probabilistic models". In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN'99)*, vol. 2 of *IEE Conference Proceedings*, pp. 946–951. The IEE, 1999. ISBN 0 85296 721 7.

[46] M.-L. Shyu, S.-C. 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 (ICDM'03)*, pp. 172–179. 2003. Http://www.cs.fiu.edu/ chens/PDF/ICDM03_WS.pdf.

[47] D. De Ridder, E. Pekalska and R. Duin. "The economics of classification: error vs. complexity". In *Proceedings of the 16th IAPR International Conference on Pattern Recognition (ICPR 2002)*, vol. II, pp. 244–247. IAPR, IEEE Computer Society Press, Los Alamitos, CA, 2002. Http://inpc55.et.tudelft.nl/ dick/publications.html.

[48] N. Japkowicz. *Concept-Learning in the Absence of Counter-Examples: An Autoassociation-Based Approach to Classification*. Ph.D. thesis, Technical Report DCS-TR-390, Rutgers University, October 1999.

[49] G. E. Hinton. "Connectionist learning procedures". *Artif. Intell.*, vol. 40, no. 1-3, pp. 185–234, 1989. ISSN 0004-3702.

[50] B. ling Zhang. "Internet Intrusion Detection by Autoassociative Neural Network". In *MMU International Symposium on Information and Communications Technologies 2005*. 2005. URL `http://m2usic.mmu.edu.my/main/proceeding05bysession.html`.

[51] R. Agrawal, T. Imielinski and A. N. Swami. "Mining Association Rules between Sets of Items in Large Databases". In P. Buneman and S. Jajodia (editors), *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207–216. ACM Press, New York, NY, USA, May 1993. ISBN 0-89791-592-5. doi: http://doi.acm.org/10.1145/170035.170072. URL `http://citeseer.ist.psu.edu/agrawal93mining.html`.

[52] R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules". In J. B. Bocca, M. Jarke and C. Zaniolo (editors), *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, September 1994. ISBN 1-55860-153-8. URL `http://citeseer.ist.psu.edu/agrawal94fast.html`.

[53] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen and A. I. Verkamo. "Finding interesting rules from large sets of discovered association rules". In N. R. Adam, B. K. Bhargava and Y. Yesha (editors), *Third International Conference on Information and Knowledge Management (CIKM'94)*, pp. 401–407. ACM Press, New York, NY, USA, 1994. ISBN 0-89791-674-3. URL `http://citeseer.ist.psu.edu/klemettinen94finding.html`.

[54] R. Bayardo Jr. and R. Agrawal. "Mining the most interesting rules". In U. Fayyad, S. Chaudhuri and D. Madigan (editors), *Proceedings of*

the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 145–154. ACM Press, New York, NY, USA, September 1999. ISBN 1-58113-143-7. doi: http://doi.acm.org/10.1145/312129.312219.

[55] P.-N. Tan, V. Kumar and J. Srivastava. "Selecting the right interestingness measure for association patterns". In O. R. Zaïane, R. Goebel, D. Hand, D. Keim and R. Ng (editors), *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 32–41. ACM Press, New York, NY, USA, 2002. ISBN 1-58113-567-X. doi: http://doi.acm.org/10.1145/775047.775053.

[56] F. Angiulli, G. Ianni and L. Palopoli. "On the complexity of mining association rules". In *Ninth National Symposium on Advanced Database Systems (Sistemi Evoluti per Basi di Dati)*. 2001.

[57] D. Barbara, J. Couto, S. Jajodia and N. Wu. "ADAM: a testbed for exploring the use of data mining in intrusion detection". *SIGMOD Rec.*, vol. 30, no. 4, pp. 15–24, 2001. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/604264.604268.

[58] W. Lee and S. Stolfo. "A framework for constructing features and models for intrusion detection systems". *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 227–261, 2000. ISSN 1094-9224. doi: http://doi.acm.org/10.1145/382912.382914.

[59] S. Manganaris, M. Christensen, D. Zerkle and K. Hermiz. "A data mining analysis of RTID alarms". *Computer Networks*, vol. 34, no. 4, pp. 571–577, October 2000.

[60] X. Qin and W. Lee. "Statistical Causality Analysis of INFOSEC Alert Data". In G. Vigna, C. Kruegel and E. Jonsson (editors), *Proceedings of The 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, vol. 2820 of *Lecture Notes in Computer Science*, pp. 73–93. Springer-Verlag Heidelberg, Heidelberg,Germany, 2003. ISBN 3-540-40878-9. doi:10.1007/b13476.

[61] R. H. Shumway and D. S. Stoffer. *Time series analysis and its applications*. Springer texts in statistics. New York: Springer, 2000. ISBN 0-387-98950-1.

[62] N. Ye, C. Borror and Y. Zhang. "EWMA techniques for computer intrusion detection through anomalous changes in event intensity". *Quality and Reliability Engineering International*, vol. 18, no. 6, pp. 443–451, 2002.

[63] N. Ye and Q. Chen. "Computer intrusion detection through EWMA for auto-correlated and uncorrelated data". *IEEE Transactions on Reliability*, vol. 52, no. 1, pp. 73–82, 2003.

[64] A. Ypma and R. Duin. "Support objects for domain approximation". In *Proceedings of Int. Conf. on Artificial Neural Networks ICANN'98*. 1998.

[65] T. Lane and C. E. Brodley. "Temporal Sequence Learning and Data Reduction for Anomaly Detection". *ACM Transactions on Information and System Security*, vol. 2, no. 3, pp. 295–331, 1999.

[66] D. Tax and R. Duin. "Data domain description using support vectors". In M. Verleysen (editor), *Proc. of 7th European Symposium on Artificial Neural Networks (ESANN'1999)*, pp. 251–256. D-Facto, 27 rue du Laekenveld, B-1080 Brussels, Belgium, 1999. ISBN 2-600049-9-X.

[67] V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[68] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor and J. Platt. "Support vector method for novelty detection". *Advances in Neural Information Processing Systems*, vol. 12, 2000.

[69] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola and R. C. Williamson. "Estimating the Support of a High-Dimensional Distribution". *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[70] K. Wang and S. J. Stolfo. "One Class Training for Masquerade Detection". In *ICDM Workshop on Data Mining for Computer Security (DMSEC)*. Available from `http://www.cs.fit.edu/~pkc/dmsec03/dmsec03notes.pdf` (read 20.10.2005), November 2003.

[71] K. A. Heller, K. M. Svore, A. D. Keromytis and S. J. Stolfo. "One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses". In *In Proceedings of the ICDM Workshop on Data Mining for Computer Security (DMSEC 2003)*. 2003.

[72] B. V. Nguyen. "An Application of Support Vector Machines to Anomaly Detection". Report cs681, Department of Mathematics, Ohio University, Athens, Ohio, USA, Fall 2002.

[73] A. Lazarevic, L. Ertoz, A. Ozgur, J. Srivastava and V. Kumar. "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection". In *Proceedings of Third SIAM Conference on Data Mining*. 2003.

[74] N. Tisby. "On the application of mixture AR hidden Markov models to text independent speaker recognition". *IEEE Transactions on Signal Processing*, vol. 39, no. 3, pp. 563–570, Mar 1991.

[75] A. Oulasvirta, M. Raento and S. Tiitta. "ContextContacts: Re-Designing SmartPhone's Contact Book to Support Mobile Awareness and Collaboration". In *Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices and Services, MOBILEHCI'05*, pp. 167–174. ACM, 2005.

[76] M. Raento, A. Oulasvirta, R. Petit and H. Toivonen. "ContextPhone, a prototyping platform for context-aware mobile applications". *IEEE Pervasive Computing*, vol. 4, no. 2, apr–jun 2005. ISSN 1536-1268.

[77] O. Mazhelis, S. Puuronen and M. Raento. "Evaluating classifiers for mobile-masquerader detection". In *Proceedings of Security and Privacy in Dynamic Environments (SEC2006), 21st IFIP TC-11 International Information Security Conference (to appear)*. Springer Science and Business Media, 2006.

[78] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2000.

[79] D. H. Wolpert. *The Mathemtatics of Generalization*, chap. The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework. Addison Wesley, 1994.