# Hybrid Self-Organizing Feature Map (SOM) For Anomaly Detection in Cloud Infrastructures Using Granular Clustering Based Upon Value-Difference Metrics

Ioannis M. Stephanakis[1], Ioannis P. Chochliouros[2], Evangelos Sfakianakis[3], Syed Noorulhassan Shirazi[4] and David Hutchison[5]

[1] Hellenic Telecommunication Organization S.A. (OTE),
99 Kifissias Avenue, GR-151 24, Athens, Greece
+30 210 611579, stephan@ote.gr
[2] Research Programs Section, OTE
Pelika & Spartis Str., GR-151 22 Maroussi, Greece
ichochliouros@oteresearch.gr
[3] Research Programs Section, OTE
Pelika & Spartis Str., GR-151 22 Maroussi, Greece
esfak@oteresearch.gr
[4] School of Computing & Communications
Lancaster University, LA1 4WA, UK
n.shirazi@lancaster.ac.uk
[5] School of Computing & Communications
Lancaster University, LA1 4WA, UK
d.hutchison@lancaster.ac.uk

## 1 Introduction

Cloud computing delivers computing services from large, highly virtualized network environments to many independent users, using shared applications and pooled resources. One may distinguish between *Software-as-a-Service* (SaaS), in which case software is offered on-demand through the Internet by the provider and it is parameterized remotely (like for example on-line word processors, spreadsheets, Google Docs and others), *Platform-as-a-Service* (PaaS), in which case customers are allowed to create new applications that are remotely managed and parameterized and the platform offers tools for development and computer interface restructuring (like for example Force, Google App Engine and Microsoft Azure [29]) and *Infrastructure-as-a-Service* (IaaS), in which case virtual machines, computers and operating systems may be controlled and parameterized remotely. Cloud computing can also be classified based on their deployment models; *public cloud*, where everyone may register and use the services, *private cloud* - that is accessible through a private network - and *partner cloud* - that offers services to specific partners/users. A hybrid cloud is a combination of private/internal and external cloud resources that enables outsourcing of noncritical functions whilst keeping the remainder internal. Modern virtualized cloud environments promote the aspects of elasticity and resource transparency that are enabled by service and *Virtual Machine* (VM) migration. The majority of cloud management software ensures that a VM retains its network identity and connectivity by initiating a "live" or "cold" migration strategy. "Live" migration allows to move the entire running VM (i.e. its active memory and execution state) from one physical node of the cloud to another without significant downtime as an effective newline resource-management strategy empowering workload balancing. On the other hand, a VM should be powered off before migration during the so called "cold" migration. Migration is a key feature of cloud environments that introduces novel security and resilience challenges [23]. There are several standards that refer to secure cloud computing. They require that anomaly activity should be detected in a timely manner and potential impact of events should be analyzed. A baseline of network operations and expected data flows for users and systems should be established and managed whereas attack targets and methods have to be evaluated regarding their potential impact. Event data should be aggregated and correlated from multiple sources and sensors. Incident alert thresholds should be established. An anomaly detector applied to network traffic visible at the cloud infrastructure level could be misled by the effect of migration on that traffic in two ways. First, legitimate migration could be misidentified as an anomaly. Second, migration could occur simultaneously with a genuine challenge, and thus mask its detection [38]. Furthermore, the elasticity also generates a huge volume of monitoring data (big-data problem), which can be considered as overhead for underlying detection mechanisms. Due to these issues anomaly detection in the cloud is a challenging area of active research and several software tools have been developed to this end [1]. These systems aim at identifying anomalous events with respect to normal system behavior. Such systems assume a model of normal behavior and issue alerts whenever operational characteristics deviate from the prescribed "normal" behavior making a suitable assumption that such changes are frequently caused by malicious or disruptive events. Anomaly detection techniques for cloud environments are still evolving due to the fact that the topic presents several challenging aspects as discussed earlier.

An hybrid NN-expert system in artificial intelligence is a computer system that emulates the decision-making ability of a human expert. Subsystems include the inference engine and the knowledge base. The knowledge base represents facts and rules. The inference engine applies the rules to the known facts to deduce new facts. Inference engines can also include explanation and debugging abilities. Such hybrid systems include Neural Networks [14], blackboard systems, belief (Bayesian) networks, case-based reasoning and rule-based systems and can be implemented in a variety of ways. Anomalies, on the other hand, are classified as point anomalies – if an individual data instance can be considered as anomalous with respect to the rest of the data – contextual anomalies – if an information occurrence is anomalous in a precise context – and collective anomalies – if collections of data instances are anomalous with respect to the entire data set. **Table 1** summarizes various anomaly scenarios and their types related to cloud implementations. Generally there are many anomaly detection techniques: Principal Component Analysis (PCA) [31], clustering-based methods [31,49], Naive Bayesian approaches [50] and Expectation-Maximization Gaussian Mixture Models [9]. Self-Organizing Maps (SOM) have been used for detection

as well (see, for example, [36]). Ordered sequences, i.e. continuous and discontinuous pattern matching, constitute an alternative proposition [2]. A survey of anomaly detection approaches is given in [26,32]. Several approaches for anomaly detection have been tested within the framework of current EU projects [19]. Most of the above mentioned techniques require clustering high dimensional data addressing, thus, specific challenges inherent in cloud operation. This is really a challenging task since all sampling points tend to become outliers as dimensionality increases and clustering algorithms falter. Full space clustering becomes computationally expensive and inefficient since it is easy to miss clusters. A number of approaches to subspace clustering have been proposed in the past two decades. A review of the methods from the data-mining community can be found in [30]. One may also see [46] for subspace clustering techniques. They include matrix factorization-based algorithms [42], algebraic-geometric generalized PCA [47,27], Gaussian Mixture Models [21] and mixtures of probabilistic PCA [45], locally linear manifold clustering [16] and sparse subspace clustering (SSC) [13] as well as local density approaches as data clouds [3] and dimension induced clustering [15]. The SECCRIT Consortium has developed an architectural framework for deploying critical infrastructure services in the cloud, which provides a basis for the development of our Cloud Resilience Management Framework. Several architectures for anomaly detection based upon the cloud have been investigated. We can apply $D_2R_2$ to the SECCRIT architectural framework to provide a resilience view (**Fig. 1**), see for example [39]. At the physical layer, the cloud infrastructure operator has access to physical nodes and the network, which can be monitored to inform the detection process. The operator can also reconfigure these devices, in response to detected challenges using policies. In a cloud infrastructure, $D_2R_2$ may exist as monitoring and reconfiguration points on physical hosts and networks as well as on some virtual components. Resilience managers and detectors need not exist on any physical equipment used directly to provide virtual resources to the above layer. At the tenant infrastructure layer, the tenant has access to VMs, and possibly virtual taps on VNs, which can inform detection. In response to challenges, the tenant may reconfigure the hosted machines, and some functionality of the virtual networks might also be exposed. Thus, tenant-infrastructure $D_2R_2$ is spread across components visible to this layer. Within the inner $D_2R_2$ loop, some interaction between these layers may exist in the form of events and reconfigurability exposed by the lower layer.

The contribution of this paper consists mainly of presenting a novel approach of producing *Self-Organizing Feature Maps* (SOFMs) of sets of ordered structures. The structures within a set may contain binary as well as vector components and may be considered as parameterizations of distinctive subspace clusters of a distributed high dimensional input space. The input space entails measurements from the entire cloud that pertain to normal operation. Subspace measurements refer to a set of cloud servers and local traffic monitoring. **Section 2.1** deals with the challenges for anomaly detection in cloud environments and the modular architecture of a state-of-the-art platform for such a system. Literature references for several approaches are cited. **Section 3** outlines the basic notions of subspace clustering and presents the well-known algorithm of Expectation Maximization (EM) for Gaussian Mixture Model and non-parametric representations of clusters based upon *Reduced* (*aggregate*) *ordered* sets. Each ordered set can be regarded as a granule of information with internal structure featuring scalar, vectorial as well as categorical attributes [34]. **Section 4** elaborates upon the representation of subspace clusters by the nodes of a SOFM, defines the structure of multiset inputs for binary and vectorial measurements and describes distance measures between ordered mulisets based on the *Cross-Order Distance matrix*. The details of the proposed algorithm and its training are given in **Section 5**. Numerical simulations for structured measurements pertaining to anomaly detection in cloud environment are illustrated in **Section 6**. This work is concluded by discussing results of anomaly detection which are obtained by the application of the proposed method and provides suggestions for further research.

## 2 The Structure Of A Cloud-Based Anomaly Detection NN Hybrid System

### 2.1 Knowledge base and the neighborhood model– Causes of Anomalies

An anomaly detection system is formally considered as an information system [11], which can be written as a quadruple $IS=(U, A, V, f)$, where: $U$ is a non-empty finite set of objects, called a universe, $A$ is a non-

empty finite set of features, V is a union of feature domains such that $V = \cup_{a \in A} V_a$ where $V_a$ denotes the value domain of feature $a$, $f: U \times A \rightarrow V$ is an information function such that $f(x, \alpha) \in V_\alpha$ for every $\alpha \in A$ and $x \in U$. $f(x, \alpha)$ can be defined upon scalar, vectorial or binary-word attributes. One may split set A of features into two subsets $C \subset A$ and $D = A - C$ conditional set of features and decision (or class) features respectively. The condition features represent measured features of the objects, while the decision features are a posteriori outcome of classification. The value difference metric (*VDM*) was introduced by [41] to provide an appropriate distance function $D(x,y)$ on nominal attributes. A simplified version (without the weighting schemes) of the VDM is defined as follows: $VDM = \sum_{f \in F} d_f \left(x_f, y_f\right)$ *w*here $F$ is the set of all features in the problem domain, and $x$ and $y$ are any two objects between which distance is calculated. For any feature $f \in F$, $d_f(x_f, y_f)$ is defined as the distance between the probability density of object $x$ on feature $f$ at $x_f$ , $P(x_f)$, and the probability density of object $y$ on feature $f$ at $y_f$, $P(y_f)$. A system for anomaly detection recognizes two possible decision features $\{$*normal traffic*, *anomaly*$\}$. We try to determine a set of objects $\{x_1, x_2, x_3 \ldots \ldots\} \in U$ - along with their corresponding subspaces $\{B(x_1), B(x_2), B(x_3) \ldots \ldots\} \subseteq C$ - that provide a representation of class *normal traffic* as a superposition of local clusters defined upon various feature metrics. Anomalies are due to malicious activities or application-level malfunctioning.

*Intrusion Detection System* (IDS) is a software application or device that implements an expert system and monitors the activities of a network for policy violations or malicious activities. It generates reports to the management system. A number of systems may try to prevent an intrusion attempt but this is neither required nor expected from a conventional monitoring system. The main focus of *Intrusion Detection and Prevention Systems* (IDPS) is to identify possible incidents, log information about them and report attempts. Organizations use IDPS for other purposes as well, like - for example - identifying problems with security policies, deterring individuals and documenting existing threats from infringing security policies. IDPS have become an essential addition to the security infrastructure of nearly every organization. Various techniques can be used to detect intrusions.

Public and private clouds can be affected both by malicious attacks and infrastructure failures (like for example power outages). Such events may have an impact upon cloud operations. The authors in [8] attempt to develop an understanding of the challenges faced by customers of an *Infrastructure-as-a-Service* (IaaS) cloud, along with their experience in resolving related problems. Their work is based on actual user problems and everyday practices as reported to the open support forum of a large IaaS cloud provider. They found that - exempt from problems related to application-level malfunctioning - the observed problems are closely related to the introduction of virtualization, i.e. connectivity issues, virtual-image management, performance, poor isolation between users, hardware degradation and others. These findings are supported by suplamental literature documenting virtualization-specific attacks by attackers gaining control over installed VMs, (like for example, DKSM [4] and “bluepill” [37]).

Cloud providers usually install anomaly detection among other detection mechanisms [40] in order to tackle these challenges However, the increasing size and complexity of applications – along with the large scale of data centers in which they operate - make anomaly detection extremely challenging. Each computer server hosts hundreds of VMs, and each VM hosts hundreds of application processes resulting in very large monitoring metrics which may obscure detection. Determining applicable metrics in order to achieve efficient detection is another challenge. A metric of high dimensionality may yield poor detection results; it is complex as well as computationally expensive. Dynamic invocation of VMs, VM migration, frequent installations and removals of applications result in an ever-changing workload pattern. These variations in workload make it extremely difficult to detect and identify anomalies. Extracting knowledge from data streams in real-time or almost real-time is essential in order to avoid failures. Executions of remediation and recovery strategies have to be prompt. Inherent properties of cloud-computing make anomaly detection complex and challenging.

### 2.2 State-of-the-art Anomaly Detection in The Cloud

Anomaly detection in the context of virtualized data centers is a rather new research problem. An anomaly-based technique to detect intrusions at different layers of the cloud was proposed in [17]. However, it was not sufficiently demonstrated how to operationally apply such a technique. In [22], the authors propose a multi-level approach, which provides fast detection of anomalies discovered in the system logs of each guest OS. One of its disadvantages is the apparent lack of scalability, since it requires increasingly more resources under high system workload. Tree-Augmented Naive (TAN) Bayesian network is used in PREPARE in order to predict online anomalies and proactively take prevention actions [43].

Recent approaches tend to combine flexible scalable analytics and Monitoring-as-a-Service (MaaS) for next generation monitoring and anomaly detection systems. Such systems implement real-time anomaly detection as well as continuous and distributed pattern analysis [6,25]. Furthermore anomaly detection methods may be classified as parametric ones [44] and non-parametric ones [35]. Parametric approaches adopt simple statistical models for anomalous and background traffic in time domain. Model parameters are estimated in real time and there is no need for a long training phase or manual parameter training. Such examples include spectral methods [18,20] as well as multiple and sequential hypothesis testing (like for example sequential probability ratio tests SPRT tests combined with bivariate Parameter Detection Mechanism (bPDM) [48]. Non-parametric methods do not assume an underling model but rather depend upon the inherent structure of the data and composite indicators (see for example the CUSUM algorithm [7] as well as Shewhart charts based upon Mann-Whitney statistics and the Wilcoxon Signed-Rank Test). The literature on composite indicators offers several examples of aggregation techniques. The most used are additive techniques that range from summing up to aggregating weighted normalised indicators. Yet, additive aggregations imply requirements and properties, both of the indicators and of the associated weights, which are often not desirable and, at times, difficult to meet or burdensome to verify. To overcome these difficulties the literature proposes other, and less widespread, aggregation methods such as multiplicative (e.g. geometric) aggregations or non-compensatory aggregations, such as the multi-criteria analysis [28].

### 2.3 Virtualized Architecture of a Cloud Based Anomaly Detection System Based on Mining And Clustering Approaches

Anomaly Detection Systems in the cloud can be modeled as distributed information systems which are implemented as Network-Function Virtualizations (NFN) (which use the technologies of IT virtualization in order to virtualize entire classes of network node functions into building blocks). Such building blocks may connect, or chain together, in order to create communication services. Clustering of the sets of measurements pertaining to such information systems are implemented using the aforementioned techniques (SOFM, neural networks, EM-GMM). The attributes of the sets of measuremants comprise a long list of scalar, vector and binary-word features. One may use a set of local clusters to indicate normal operation. Local subspace distributions of measurements upon conditional attributes are used to represented clusters. One may use one representative set of measurements for a cluster or two representative sets of measurements or more. The subspaces which are defined upon conditional attributes may vary depending upon the representative measurement. Anomalies are detected as outliers of such an expert database.

Virtualized network functions (VNF) consist of one or more VM running different software and processes, on top of standard high-volume servers. A reference architecture used in a cloud based anomaly detection system divides activities in a cloud environment into four layers (see Fig. 1.a). Anomaly detection gets input from network and system activity, which is measurable at the physical (cloud-infrastructure provider) layer, which consists of physical networks and machines, and has an external view of system activity in VMs. Additionally, network activity can be measured in the tenant-infrastructure layer by monitoring traffic on virtual networks. Tenants running anomaly detectors on VMs

accessing these networks implement such a modular architecture. The proposed approach carries out distributed sampling at various cloud sites and assigns a structured set of local measurements to a specific server/client connected to the cloud. The block diagram of the proposed approach based on such an architecture is illustrated in Fig. 1.b. It consists of six (6) algorithmic steps. Structured sets of measurements throughout the cloud are ordered according to their similarity and their VDM distance from each other in a subsequent step of the algorithm. We distinguish between orders pertaining to normal and abnormal (anomalous) network traffic. An ordered set of measurements featuring dissimilar distribution over the VDM distance may be indicative of an anomaly. As a final step one may compare the ordered set with ordered sets of measurements pertaining to normal operation (which are used as reference sets). The ordered sets used as references are the nodes of a trained Self-Organizing-Feature-Map (SOM) for normal cloud operation.

## 3 Different Inference Engines For Subspace Clustering – Representing Clusters as Ordered Sets of Features

All subspace clustering methods can be used for determining clusters of measurements indicating normal operation by assuming that a subspace corresponds to selected features (as defined by the mesurements taken from the servers connected to the cloud). VDM distance are used. The proposed approach is intended as a non-parametric alternative of such algorithms as the Expectation Maximization algorithm for Gaussian Mixture Models (EM-GMMs) [12]. The basic idea of the EM algorithm is to estimate an updated model if the probability of the new model is greater than or equal to the previous estimate. The new model then becomes the initial model for the next iteration and process is repeated until some convergence threshold is reached. One may consider the problem of representing a collection of data points as a union of subspaces. Let $\{\mathbf{x}_j \in R^{Din}\}_{j=1}^{all\ samples}$ be a given set of points drawn from an unknown union of subspaces $S_i = \{\mathbf{x} \in R^{Din}: \mathbf{x} = \boldsymbol{\mu} + \mathbf{U}_i\mathbf{y}\}$, $i$=1,…$m$, where $\boldsymbol{\mu}_i \in R^{Din}$ is an arbitrary point in subspace $S_i$. Should $G(x; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ stand for the probability density function of a $D_{in}$-dimensional Gaussian with mean $\mu$ and covariance matrix $\boldsymbol{\Sigma}$, then $p(x) = \sum_{i=1}^{m} \pi_i\, G\big(x; \boldsymbol{\mu}_i, \mathbf{U}_i\mathbf{U}_i^{\mathrm{T}} + \sigma_i^2\mathbf{I}\big)$) and $\sum_{i=1}^{m} \pi_i = 1$ where parameter $\pi_i$, called the mixing proportion, represents the a priori probability of drawing a point from subspace $S_i$). The ML estimates of the parameters of this mixture model can be found using expectation maximization (EM) during normal traffic conditions. Anomaly detection is carried out by performing the expectation step during anomalous network operation. Gaussian distributions may overlap as illustrated in Fig. **2a**. This feature is useful in cases in which a specific state of the network is represented by a complex subspace in the domain of the measurements. Alternative approaches for representing distributions of data are the non-parametric ones. Such an approach used in the context of this research for comparison purposes is based upon the concept of data density [3]. It requires a small amount of data namely the mean of all data samples and a scalar product quantity calculated dynamically over time that indicates the spread of the data around the estimated center of a cluster, i.e. $d_\alpha = \frac{1}{1+\frac{1}{N_\alpha}\sum_{i=1}^{N_\alpha}\|x_\alpha - c_n^{closest}\|^2}$. Obviously index $d_\alpha$ ranges from zero to one. The concept of this approach is illustrated in Fig. **2.b** whereas training and anomaly detection are depicted in **Fig. 2.c** The following steps outline the aforementioned approach based on data density:

- Estimate cluster centers derived from measurements indicating normal operation.
- Set a goal (*threshold* 1) for the value of local data density. Start with one cluster and add one cluster at a time.
- Stop adding clusters should you exceed a predetermined threshold.
- Check a data distribution over the set of estimated cluster centers.
- Should local data density fall below a pre-specified threshold (*threshold* 2) detect anomaly (positive indication).

This approach is used for comparison purposes in **Section 6**.

Our proposed non-parametric approach is based upon ordered sets of features as well as specific norms in order to represent. There is no universally accepted method for ordering multivariate data. Widely known multivariate ordering methods include [5]:

- *Marginal ordering* (*M-ordering*) according to which feature vectors are ordered in each component independently. This scheme produces a set of ordered output vectors that is usually not the same as the set of input vectors.
- *Conditional ordering* (*C-ordering*) according to which vectors are ordered based on the marginal ordering of one of their components. This scheme disregards the vectorial nature of the multichannel data.
- *Partial ordering* (*P-ordering*) according to which vectors are partitioned into smaller groups that are then ordered. Despite its theoretical appeal, this scheme is computationally demanding. Since partial ordering is difficult to perform in more than two dimensions, it is not appropriate for three-component signals.
- *Reduced* (*aggregate*) *ordering* (*R-ordering*) according to which the feature vectors are first reduced to scalar representatives using a suitable distance or similarity measure. The ordering of these scalars is then taken as the ordering of the corresponding vectors. This is the most common ordering scheme in the literature.

The reduced ordering scheme is the most attractive and widely used in signal processing since it relies an overall ranking of the original set of input samples and the output is selected from the same set. The ordered sequence of scalar values $D(1) \leq D(2) \leq \ldots \leq D(i) \leq \ldots \leq D(N)$ for $i = 1,2 \ldots N$ implies the same ordering of the corresponding vectors $\mathbf{x}_i$ , i.e. $\{\mathbf{x}\ (1), \mathbf{x}\ (2) \ \ldots \ \mathbf{x}\ (i) \ \ldots \ \mathbf{x}\ (N)\}$. *R*-ordering non-linear processing is based on the ordering of aggregated distances, i.e. $D_i = \sum_{j=1}^{N} d(\mathbf{x}_i, \mathbf{x}_j)$ or aggregated similarities $D_i = \sum_{j=1}^{N} similarity(\mathbf{x}_i, \mathbf{x}_j)$ [10]). Let us assume that $x_k \in S_n$, where $S_n$ consist of $n$ repetitions of ordering experiments in normal or anomalous conditions, then it is assumed that $\lim_{S_n \to \infty} (\Pr\{order(x) = c\}) = 1$. This is condidered as a crisp ordering case. Nevertheless fuzzy outcomes are possible as well. One may define histograms upon such aggregate distances in order to distinguish between normal and anomalous traffic conditions. A bin by bin comparison of the probability distributions of the histograms over several value-difference metrics (*VDM*s) defines the neighbourhood-based object outlier factor of $x$ in $S$ as $NOOF(x_i) = \sum_{\substack{x_i, x_j \in S \\ i \neq j}} VDM_{all\ attributes}(\mathbf{x}_i, \mathbf{x}_j)$. Attributes that feature different histograms under normal and anomalous conditions should be selected. This implies that a selection of a set of value-difference metrics (*VDM*s) has to be made. One may arrange objects $x$ in a neighbourhood according to their neighbourhood-based object outlier factor, i.e. $NOOF(x(1)) \leq \ldots \leq NOOF(x(i)) \leq \ldots \leq NOOF(x(N))$. A representation of overlapping clusters by reduced/aggregate ordered sets of points in 2-D is illustrated in **Figs. 3**. Histograms of the number of ordered vectors over distance are presented in **Figs. 3b** to **3d** for the three distributions (for twenty ordered 2-D vectors). Lower order vectors tend to occupy the central part and most probable part of a local distribution.

## 4 A Model of Self-Organizing Feature Map (SOFMs) Based on Reduced/Aggregate Ordering of Subspace Features

### 4.1 Cloud Distributive Environment And Input Subspaces

Sampling of binary and vector features is carried out over all host and client servers connected to the cloud for a time window $[t_1, t_2]$ according to **Fig. 4**. Hence a ranking of all host and client servers connected to the cloud results after aggregate ordering of their feature vectors as explained in the previous paragraph. The spreading of feature vectors over a considerable distance range is indicative of an anomaly. Ordered sequences of feature vectors during normal cloud operation are clustered in nodes using a SOFM. Analyses using EM-GMM as well as local data densities are carried out for comparative purposes.

The proposed approach (**Fig. 1b**) consists of sampling the cloud network during operation for small time windows $[t_1\ \ t_2]$, $[t_3\ \ t_4]$, $[t_5\ \ t_6]$ …… and selecting samples of the form $x_s(\alpha_{packets}\ \ \alpha_{bytes}\ \ \alpha_{active\ flows}\ \ \alpha_{source/destination\ IP}\ \ \alpha_{source/destination\ port}\ \ \alpha_{packet\ size}; [t_1\ \ t_2])$ where $s \in \{U: s \text{ indicates a specific network condition}\}$. The samples that correspond to host and client servers connected to the cloud are then ordered in ascending distance order according to the reduced ordering scheme described in **Section 3**. One may use selected members of the ordered set, like for example the first *K* members in order to train a SOFM as described in the sequel. Each vector represents a structured record comprising binary (or octal or hex) information along with multivalued data. The proposed approach is directly applicable to data-base records. SOFM clusters the universe knowledge of the anomaly detection hybrid system.

### 4.2 Definition of the Cross-Order Distance Matrix Between Ordered Objects

The *Cross-Order Distance Matrix* is defined along with a distance or similarity measure and a method of selecting the elements of the *Cross-Order Distance Matrix* (or operating upon them) in order to estimate the distance between two ordered sets of feature vectors, denoted as $S$=**{x** (1), **x** (2) … **x** (*i*) … **x** (*N*)} where $s \in \{U: s \text{ indicates an anomaly}\}$ and $S$'=**{x'** (1), **x'**(2) … **x'** (*i*) … **x'** (*N*)} $s' \in \{U: s' \text{ indicates normal conditions}\}$. Each element of the matrix is a value difference metric (*VDM*) as defined in **Section 3**. Thus,

$$D_{SS'} = \left[dist(x_i, x_j)\ where\ i,j = 1 \dots \dots N; measure, method\right] = \\ = \begin{pmatrix} d(x^S(1), x^{S'}(1)) & d(x^S(1), x^{S'}(2)) & \cdots & d(x^S(1), x^{S'}(N)) \\ d(x^S(2), x^{S'}(1)) & d(x^S(2), x^{S'}(2)) & \cdots & d(x^S(2), x^{S'}(N)) \\ \vdots & \vdots & \ddots & \vdots \\ d(x^S(N), x^{S'}(1)) & d(x^S(N), x^{S'}(2)) & \cdots & d(x^S(N), x^{S'}(N)) \end{pmatrix} \quad (1)$$

A *method* (see **Table 2**) can be the sum of all elements of the *Cross-Order Distance Matrix*, its trace (defined as $trace\{\mathbf{D}_{SS'}\} = \sum_{k=1}^{N} d(x^S(k), x^{S'}(k))$, constant thresholding of all elements of the matrix (i.e. setting all elements below the threshold equal to zero and carry out summation over all non-zero elements), non-constant threholding using a rule such as

$$threshold = \min(d(x_1^S, x_N^S), d(x_1^{S'}, x_N^{S'})), \quad (2)$$

Let us consider sets of data that are indicative of the state of the cloud within time interval $[t_1\ t_2]$, i.e. $x_s(\alpha_{source/destination\ IP}, \alpha_{source/destination\ port} \cdots \cdots ; [t_1\ \ t_2])$ and refers to some host/server connected to the cloud. Each node of the SOFM in **Fig. 5** consists of an ordered set of samples $\{x^S(1), x^S(2), x^S(3), x^S(4)\dots\}$ of feature vectors which corresponds to servers/hosts connected to the cloud. Distributions of measurements within an interval $[t_1\ t_2]$ may refer to binary (octal or hex) words of data - like for example IP addresses or ports - or scalar data, like for example packet sizes. Comparisons are carried out between ordered sets pertaining to interval $[t'_1\ t'_2]$, i.e. $\{x^{S'}(1), x^{S'}(2), x^{S'}(3), x^{S'}(4) \dots\}$, and ordered sets pertaining to interval $[t_1\ t_2]$, i.e. $\{x^S(1), x^S(2), x^S(3), x^S(4) \dots\}$. The *Cross-Order Distance Matrix* between two such ordered sets is defined in order to quantify the similarity-distance between them.

Feature measurements during normal operation are clustered to the nodes of the SOFM according to some method applied upon the *Cross-Order Distance Matrix*. Measurements during an intrusion attack yield outliers of the trained SOFM and irregular histograms over VDM distances. The proposed approach may use all or selected rank samples during training.

Ordering of the sample set is a necessary preprocessing step. We use value difference metric (VDM) in order to find the distance between $x_{s'}$ and $x_s$, i.e.

$$d(x_{S'}, x_S; [t_1\ t_2]) = \{d(\alpha_{source/destination\ IP\ address}(s'), \alpha_{source/destination\ IP\ address}(s); [t_1\ t_2]) + \quad (3)$$

$$+ d(\alpha_{source/destination\ port}(s'), \alpha_{source/destination\ port}(s); [t_1\ t_2])$$
$$+ d(\alpha_{packet\ size}(s'), \alpha_{packet\ size}(s); [t_1\ t_2]\} =$$

$$= \left( \frac{\left|\alpha_{source/destination\ IP\ address}(s') - \alpha_{source/destination\ IP\ address}(s)\right|}{\left|\alpha_{source/destination\ IP\ address}(s')\right| + \left|\alpha_{source/destination\ IP\ address}(s)\right|} + \right.$$
$$+ \frac{\left|\alpha_{source/destination\ port}(s') - \alpha_{source/destination\ port}(s)\right|}{\left|\alpha_{source/destination\ port}(s')\right| + \left|\alpha_{source/destination\ port}(s)\right|} +$$
$$\left. + \frac{\left|\alpha_{packet\ size}(s') - \alpha_{packet\ size}(s)\right|}{\left|\alpha_{packet\ size}(s')\right| + \left|\alpha_{packet\ size}(s)\right|} \right)$$

One constructs the histogram over the lowest and the highest value of some object attribute in the training set in order to estimate the differences within interval $[t_1\ t_2]$ between servers and clients connected to the cloud for normal traffic conditions or anomalies. The differences between histograms are obtained using the Canberra distance over all histogram bins $\frac{1}{number\ of\ non\ zero\ bins} \sum_{l=1\ldots L} \frac{\left|h^{S'}_{packet\ size}(bin_l) - h^{S}_{packet\ size}(bin_l)\right|}{\left|h^{S'}_{packet\ size}(bin_l) + h^{S}_{packet\ size}(bin_l)\right|}$. For binary (or octal or hex) data within $[t_1\ t_2]$ one may use the Jaccard distance in **Eq. 4**, which measures the dissimilarity between sample sets. It is obtained by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union,

$$d_J(S, S') = 1 - J(S, S') = \frac{|S \cup S'| - |S \cap S'|}{|S \cup S'|}. \quad (4)$$

Distances that are obtained using some process upon the *Cross-Order Distance Matrix* should allow for discerning between ordered sets. Ordered sets of eight feature vectors are used, i.e sets consisting entirely of samples of measurements taken during normal operation and sets consisting of measurements taken during abnormal operation. Thresholded distance matrix allows for better results should one consider clustering anomalies using a SOFM. The distances between distributions A, B and C in **Fig. 3.a** for three different methods (i.e. "*sum of all elements*", "*trace-sum of diagonal zone elements*" and "*thresholded cross-order matrix*") are given in **Table 2a**. The values of distances are mean values of ten (10) instances of ordered sequences (featuring forty vectors/objects each). The variance internal is provided as well. A thresholded cross-order matrix appears to be the best choice whereas the sum of all elements fails to distinguish distribution B from A and B in some cases. Rough set theory [33] can be used as well in order to fuzzify the sums of elements within blocks of the *Order Distance Matrix.* The blocks may overlap or not. One can specify the order number as {*low, medium, high*}, i.e. the ordered members around the mean value, the middle ordered members and the higher ordered members (which are indicative of the outskirts of the information cluster granule). The rough set membership functions are defined upon the aggregate distance of ordered elements belonging to predefined subsets as $\mu^{\text{index low order}}$, $\mu^{\text{index medium order}}$ and $\mu^{\text{index high order}}$ for possible distributions. One may consider, for example, the aggregate distance of the low-order elements in a set as the sum of all possible distances between pairs of elements in a predefined low-order subset, the aggregate distance of the median-order elements in a set as the sum of all possible distances between pairs of elements in a predefined median-order subset and the aggregate distance of the high-order elements in a set as the sum of all possible distances between pairs of elements in the high-order subset. A binary relation defined upon a threshold can be used in order to determine the rough set membership functions for known distributions. Should the aggregate sum of similarities or distances between the subsets of ordered members fall within a lower and an upper threshold an $index_{\text{low}}$ (or $index_{\text{medium}}$ *or* $index_{\text{high}}$ respectively) will assume the value of 1. Thus the values of the elements of the *Order Distance Matrix* can be regarded as rough functions ranging from zero (0) to one (1). The distance between different distributions is defined accordingly as a function of an initial first order estimate $d_{XY}$ and higher order estimets based upon the logical terms $(1\text{-}\mu_X{}^a\ \mu_Y{}^b)$, where *X,Y* stand for the different distibutions {*A,B,C*} and *a, b* stand for specific subsets, i.e. {*low, medium, high*}. Several choices are

available for the specific metric function to be employed [24]. **Table 2b** illustrates the outcome for the distributions in **Fig. 3.a**. The proposed SOFM can be trained for such a matrix metric. An estimation of the rough membership functions has to be made at start. A proper parametrization implies that should one draw the same number of elements from the very same underling distribution and, subsequently, order them in subsets, the resulting elements of the fuzzified distance matrix will obtain the value of zero (0).

## 5 Anomaly Detection Using SOFMs with Multiset Inputs

### 5.1 Outline of the Proposed Approach

The proposed approach (**Fig. 1b**) consists of sampling the cloud network during abnormal conditions for small time windows $[t_1 \ \ t_2]$, $[t_3 \ \ t_4]$, $[t_5 \ \ t_6]$ …… and selecting samples of the form $x_s(\alpha_{packets} \ \ \alpha_{bytes} \ \ \alpha_{active\ flows} \ \ \alpha_{source/destination\ IP} \ \ \alpha_{source/destination\ port} \ \ \alpha_{packet\ size}; [t_1 \ \ t_2])$ for each server $s \in U$. The samples that correspond to host and client servers connected to the cloud are then ordered in ascending distance order according to the reduced ordering scheme described in **Section 3**. One may use selected members of the ordered set, like for example the first *K* members in order to train a SOFM as described in the sequel. Our proposed approach suggests training of local clusters using a SOFM. These clusters indicate normal operation. Anomalies are detected as local deviations from such clusters. An initial check is carried out for irregular histogram distributions (which is indicative of an anomaly) before estimating the distance between the input vector and the nodes of the trained SOFM.

### 5.2 Aggregating Multiset Inputs into Clusters – Training

There are three basic steps involved in the application of the SOFM algorithm after initialization; namely, sampling, similarity matching and updating. Reduced/aggregated ordering of sample structured vectors within time windows can be regarded as an intermediate step. The sum of the aggregated distances between fields of an ordered structure (i.e. VDM distances) and a set of *K* feature vectors corresponding to a node of the SOFM is evaluated for all *L* nodes of the map. The *Cross-Order Distance Matrix* - as defined in **Section 4** - is used to derive the sum of the aggregated distances. The result of the application of the selected method upon the *Cross-Order Distance Matrix* is used to determine the winning neuron. The aforementioned steps are described in detail as follows:

1. *Initialization of the partial sets*. Choose the initial values for the weight vectors $w_j(0)$. Assume that each weight vector $w_j$ that corresponds to a neuron consists of a set of *K* representative host and client servers samples for the time window, i.e. $\mathbf{w}_j = (\mathbf{w}_{j,1} \ \ \mathbf{w}_{j,2} \ \ \ldots\ldots \ \ \mathbf{w}_{j,K})$ where index *j* equals 1, 2,…, *L* (where *L* stands for the total number of neurons).
2. *Sampling*. Sample cloud and server conditions for time window $[t_1\ t_2]$, $\mathbf{v}(t) = \left(x_{U_1}(t), x_{U_2}(t), x_{U_3}(t) \ldots\ldots\right)$.
3. *Reduced/aggregated ordering* of the samples corresponding to the host/client servers connected to the cloud. Arrange vectors samples in a group $\mathbf{v}(t) = \left(x_1^{S(t)}, x_2^{S(t)}, x_3^{S(t)} \ldots\ldots\right)$ in such a way that $D_k = \sum_{j=1,j\neq k}^{N} d(x^S(k), x^S(j)) \leq D_l = \sum_{j=1,j\neq l}^{N} d(x^S(l), x^S(j))$ for $k<l$. VDM distance $d\left(x^S(l), x^S(j)\right)$ aggregates the partial metrics between attribute fields within the objects, i.e. $\sum_{all\ \alpha} d_\alpha\left(f_\alpha\left(x^S(l)\right), f_\alpha\left(x^S(j)\right)\right)$ where $\alpha \in \{packets, bytes, active\ flows, soure\ \&\ destination\ IP\ \&\ port\ addresses, packet\ size \ldots\}$.
4. *Similarity Matching*. Find the best-matching (winning) neuron $i(x^{S_1}, x^{S_2}, x^{S_3} \ldots\ldots\ x^{S_L})$ at time *t* by aggregating the distances between the samples in the set and the *K* vectors at each of the *L* nodes, i.e.
$$i(\mathbf{v}) = \arg\min_j (method(\mathbf{D}(\mathbf{w}_{i,k}, \mathbf{v}(t))))$$
$$\text{where } j = 1,2, \ldots\ldots, L \qquad (5)$$
5. *Updating*. Adjust the synaptic weight vectors of all neurons, using the update formula

$$\mathbf{w}_j(t+1)=\begin{cases}\mathbf{w}_j(t)+\eta(t)\left([x_1^s(t) \quad x_2^s(t) \quad x_3^s(t) \quad \dots \quad x_k^s(t)]-\mathbf{w}_j(t)\right), & j\in\Lambda_{i(v)}(t)\\ \mathbf{w}_j(t), & otherwise\end{cases} \tag{6}$$

where the ordered lowest $K$ ranks of the training set are used, $\eta(t)$ is the learning-rate parameter and $\Lambda_{i(v)}(t)$ is the neighbourhood function centred around the winning neuron. $\Lambda_{i(v)}(t)$ is varied dynamically during learning for best results.

6. *Continuation*. Continue with Step 2 until no noticeable changes in the feature map are observed.

Samples pertaining to abnormal and normal network conditions are presented to the SOFM after training in order to detect anomalies. Similarity matching is carried out as described in **Step 4** of the algorithm summing all elements of a thresholded *Cross-Order Distance Matrix*. An anomaly is detected should the aggregate distance be higher than a threshold determined during training, i.e. anomalies are detected as outliers should the minimum distance from the nodes of the SOFM exceed a specified threshold. One assumes normal operation should minimum distance be lower than the threshold.

## 6 Experimental Setup And Numerical Simulations

We have evaluated our technique against network traces obtained from a controlled testbed resembling a cloud environment, featuring VM migration as a normal cloud operation, plus network attacks that should be regarded as anomalies. The testbed allows the traces to be labelled with ground truth, about both the expected anomalies and the presence of a migration. The testbed consists of two hosts, which serve as compute nodes running multiple VMs. Another host acts as a controller which initiates migrations and generates background traffic. A fourth host generates attack traffic. Each physical node runs KVM as virtualization infrastructure, and QEM provides hardware emulation. Migration is achieved with *libvirt*. Traces obtained at the virtual bridges are fed into detector to observe its reactions to normal/anomalous traffic. Data collector is composed of various scripts providing feature extraction and normalisation, which is achieved using *tcpStat.c* and *featExract.pl* scripts with configurable binning period. At the network level, the data collector collects traffic data through *tcpdump*[1] from each host network at bridge interface. This traffic is then passed on to a *Summary Extraction Script*, which is based on *libpcap* and converts the traffic into normalised statistical properties on a per packet basis. We extracted both volume-based features (e.g., count of bytes and packets) and distribution-based features (like the Shannon entropy of all values observed in the bin) in order to capture the dynamics of varying attack types. Network traces are split into 1-second bins for the experimental results that are presented in the sequel. A set of statistical properties (features) of the traffic in each bin is computed (**Table 3**) and each feature vector with measurements is submitted to the detector. Background traffic is created by running several HTTP servers and several clients repeatedly requesting dynamically created documents of varying size. Several anomalies are introduced during VM migration like network and port-scan attacks and *Denial-of-Service* under high and low intensities (see **Table 1**). *Denial-of-Service* attacks are realized using LOIC (an open source network stress testing tool). Experiments are characterized by background traffic and anomaly type. *Expectation Maximization* (EM) for *Gaussian Mixture Model* (EM-GMM) is employed as a conventional, parametric method for anomaly detection for the sake of comparison with the proposed approached. Indicative detection statistics are illustrated in **Table 4** for different anomalies as well as for different patterns of background traffic. Results are poor and depend upon the selection of feature vectors used in the measurements. The log-likelihood is simply the log of the probability density function of the Expectation Maximization (EM) mixture model which is used to calculate the anomaly score. The parameters of the Gaussian Mixture Models are estimated from traffic measurements corresponding to normal network operation. The log-likelihood values versus time stamps for the same data that are used to test the proposed algorithm (low intensity net scan - NS) are presented in **Fig. 6a** (two clusters), **Fig. 6.b**

[1] Description of tcpdump/libpcap is given at http://www.tcpdump.org/ and for libpcap API at http://www.tcpdump.org/

(three clusters) and **Fig. 6.c** (four clusters). The results pertain to non-normalized data. The more clusters used to implement the EM-GMM method the less stable is the convergence of the algorithm. Good results may be obtained using two clusters (solid red line) for the general case. Values of log-likelihood below a certain threshold indicate an anomaly at time *t*. The corresponding diagrams of True Positive Rate (TPR) vs False Positive Rate (FPR) are given in **Fig. 12.a**. They are extracted from various threshold values ranging from $-10^2$ to $-10^4$. Anomaly detection using non-parametric data density for the same set of experimental data (low intensity net scan - NS) is presented in **Fig. 6.d** (one cluster) and **Fig. 6.e** (four clusters) for one measurement as well as a group of measurements within a sliding window of ten timestamps in **Fig. 6.f** (one cluster) and **Fig. 6.g.** (four clusters). True Positive Rate (TPR) vs False Positive Rate (FPR) curves are given in **Fig. 12.b** (for a single measurement**)** and **Fig. 12.c** for a group of ten consecutive measurements. Normalization of data is carried out as a preprocessing step. Local density values per cluster for normal as well as abnormal operation are depicted in **Table 5**. Low values indicate some type of anomaly. Average local density values per cluster (as defined in **Section 3**) are illustrated in **Fig. 7**. Detection results do not improve for more than four (4) static clusters determined from measurements during normal operation. Additional anomaly detection results using a 10x10 SOFM and measurements from a single (cloud) server featuring the number of packets per bin, the number of bytes per bin, the number of active flows, the entropies of source IP addresses, the entropies of destination IP addresses, the entropy of destination port distribution and the entropy of packet size distribution (as indicated in **Table 3**) are given in **Fig. 6.h** and in **Fig. 12.d** (TPR vs. FPR in **Fig. 12.d**). The Canberra distance as defined in **Section 4.2** is used to train the SOFM. Detection results are good and support such a choice.

A typical ordering of six samples is given in the sequel in order to test the proposed algorithm under similar cloud conditions (for low intensity net scan anomalies). Ordered sets include a feature vector (expert DB object) with anomaly measurements at *t*, where *t* ranges from 301 to 599 (an attack is introduced during Virtual Machine migration after $t$=300), and five feature vectors (expert DB objects) corresponding to normal conditions ordered in ascending distance order (a distributed scenario). The feature vector ranked as sixth corresponds to anomalous conditions whereas the five first feature vectors correspond to normal conditions. The splitting of the histogram over distance in two parts is indicative of an anomaly according to the block diagram in **Fig. 1.b**. One set of measurements corresponds to outgoing traffic from a server during VM migration whereas the other set of measurements corresponds to incoming traffic to a server during VM migration (see **Figs. 8** for ranking typical feature vectors corresponding to single time-stamps for outgoing and incoming traffic). Splitting of the histogram is a first indication of an anomaly according to our approach. Nevertheless one has to compare ordered sequences of measurements with anomalies against ordered sequences of measurements during normal network conditions. A 10x10 SOFM featuring nodes that represent six rank sequences of measurement vectors during normal operation is trained using the Canberra distance. The projected maps for the first three ranks as well as the attribute planes of the # of bytes vs the # of packets and the entropy of source IP addresses vs the # of active flows are illustrated in **Figs. 9**. Spreading of the distributions from lower to higher ranks is observed.

It turned out that the accuracy of the proposed method is increased by taking a window of multiple time-stamps and extracting separately histograms for each feature. The number of packets over ten (10) consecutive time-stamps is used to construct a histogram of ten bins, the number of bytes is used to construct a histogram of twenty bins, the number of active flows is used to construct a histogram of fifteen bins, the entropies of source IP addresses is used to construct a histogram of twelve bins and so on (see **Table 3**). Ordering multiple histogram samples (which are obtained as described) using the Canberra distance in ascending order is given in **Figs. 10** (for low intensity net scan anomalies). The top five histograms are indicative of normal network conditions whereas the lower histograms correspond to the higher distance and are indicative of an anomaly. The overall histogram over VDM distance for inward and outward migration is split in two parts as expected (see **Figs. 11**). A 10x10 SOFM is used in order to cluster ordered histogram samples. We consider sets of multiple feature vectors consisting of six different vectors selected at random. The Self-Organizing Feature Map (SOFM) is trained using six feature vectors

corresponding to normal conditions ordered in ascending distance order. Six multiple histograms samples corresponding to normal network conditions (from $t$ to $t$+9, where $t$ ranges from 1 to 291 in **Fig. 6.i**) are ordered using the cumulative Canberra distance for all eight (8) feature histograms (**Table 3**). A total of four hundred multiple histogram samples are used in order to train the SOFM. The sum of all elements of the *Cross-Order Distance Matrix* is used in order to train a SOFM using the Canberra distance (i.e. '*measure*'='*Canberra*' and '*method*'='*all*'). The minimum distance from a node of the SOFM after training is given in **Fig. 6.i** for all time stamps ($t$ ranges from 1 to 590). The multiple histogram set includes five more samples corresponding to normal conditions. The sum of all elements of the *Cross-Order Distance Matrix* after thresholding is used to obtain the illustrated result, i.e. if

$$abs\left(d\left(x^{S}(i),x^{S'}(j)\right)-d(x^{S'}(i),x^{S'}(j))\right)\leq d(x^{S'}(i),x^{S'}(j)) \text{ and}$$

$$abs\left(d\left(x^{S}(i),x^{S'}(j)\right)-d(x^{S}(i),x^{S}(j))\right)\leq d(x^{S}(i),x^{S}(j))$$

the corresponding element (VDM) of the *Cross-Order Distance Matrix* is set to zero (this thresholding rule is used to obtain the values in **Table 2a** as well). The ratio of *True Positive Rate* vs *False Positive Rate* is evaluated by taking different thresholds and assuming that a value higher than the threshold indicates an anomaly (see **Fig. 12.e**). A SOFM featuring more nodes yields better results in real world scenarios since local clusters of multidimensional measurements are better represented, nevertheless detection improvements are minor in the artificial experiments which are presented in this paper. Anomaly detection results using a 4x4 SOFM according to the proposed approach for ordered sets of measurements for different anomaly types (Host port scan, Netport post scan, net scan and UDP Denial-of-service (UDoS) yield very good results for our experimental setup (see **Figs. 13**) for mixed training scenarios (histograms of attributes within a sliding of ten time stamps for distributed measurements at six network points).

SOFMs are based upon unsupervised clustering and render the inherent structure of data without regard of specific models for the density function (PDFs) of the parameters. Our approach assumes that each node represents a separate PDF of distributed variables (a subset of the total measurements) that indicate normal network operation. The setup assumes a null hypothesis test $H_o$, where an anomaly is detected if the distribution of a subset of measurements is significantly different from the distributions corresponding to the nodes of the trained SOFM. The SOM method can be viewed as a non-parametric regression technique. Much like a regression plane being an abstraction of the original data, the proposed SOM of ordered sequences generates a non-linear representation of the multiple data distributions (the universe of a hybrid system). Thresholding using different measures defined upon a *Cross-Order Distance Matrix* may be regarded as a non-parametric inference method using generalized statistics. Cross-Order Matrices are directly related to cross-classification tables of preprocessed (i.e. ordered) data.

## 7 Conclusion

Anomalies are classified according to their type and intensity. A novel hybrid system approach for detecting anomalies during typical cloud operation is proposed (see **Fig. 1.b**). The proposed method is based upon ordering histogram feature vectors from several monitoring sites of the network and using them to train a SOFM (which supports the inference logic of the expert system). Each node of the SOFM represents a granule of information. A conventional *Expectation Maximization Gaussian Mixture Model* (EM-GMM) as well as a non-parametric data density approach are used for anomaly detection for comparison purposes. The proposed approach yields better results. An anomaly is detected should the minimum distance from some node of the SOFM exceed a specified threshold. Estimation of the distances between the nodes of the SOFM and the ordered set (which contains a feature vector with anomaly measurements) is carried out according to a "*method*" applied upon the so-called *Cross-Order Distance* Matrix. One has to specify a certain distance measure - like the Canberra distance, which is used in the context of this work - in order to estimate the similarity of histograms of feature values within a sliding time window. Rough set measures can be used along with the *Cross-Order Distance* Matrix and SOFM training. Rough set membership functions are determined as well during training along with the

nodes of the SOFM. The proposed approach yields the server-under-attack and the existence of a network anomaly at the same time since a subspace cluster involves certain servers of the cloud. The direct analogy of local histograms over the aggregate ordering distance and value-difference metrics (VDM) is investigated. The so-called *Neighborhood-based Outlier Factor* (*NOOF*) is defined for reduced/aggregate ordered sets featuring attributes of different types.

**Acknowledgments**

This work is sponsored by UK-EPSRC funded TI3 project, grant agreement no. EP/L026015/1: A Situation-aware Information Infrastructure; and the EU FP7 project SECCRIT (Secure Cloud computing for Critical infrastructure IT) funded by grand agreement no. 312758.

**References**

1. Adamova K, Schatzmann D, Plattner B, Smith P (2014) Network Anomaly Detection in the Cloud: The Challenges of Virtual Service Migration. In IEEE International Conference on Communications, Sydney. ICC, 3770-3775. doi: 10.1109/ICC.2014.6883908
2. Alharby A, Imai H (2005) Hybrid intrusion detection model based on ordered sequences. *MMM-ACNS 2005, LNCS 3685*, Gorodetsky, V., Kotenko, I. and Skormin, V. (Eds.). Springer-Verlag, Berlin-Heidelberg, 352-365, ISBN 3-540-29113-X.
3. Angelov P, Yager R (2011) Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density. In 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS), 62-69.
4. Bahram S, Jiang X, Wang Z, Grace M, Li J, Srinivasan D, Rhee J, Xu D. (2006) DKSM: Subverting virtual machine introspection for fun and profit. Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems (SRDS), Washington, DC, USA, 2010. IEEE Computer Society, 82–91
5. Barnett V (1976) The Ordering of Multivariate Data. *Journal of the Royal Statistical Society Series A*, 139(3):318–355.
6. Basile C, Canavese D, Lioy A, Pitscheider C, Valenza F (2016) Inter-function anomaly analysis for correct SDN/NFV deployment, International Journal of Network Management, Special Issue: Software-Defined Networking and Network Functions Virtualization for flexible network management, 26(1): 25–43, doi: 10.1002/nem.1917.
7. Basseville M, Nikiforov I (1993) Detection of Abrupt changes: Theory and Application. Prentice-Hall, Englewood Cliffs, NJ.
8. Benson T, Sahu S, Akella A, Shaikh A (2010) A first look at problems in the cloud. Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud), Berkeley, CA, USA, USENIX Association, 15-15.
9. Bishop CM, Nasrabadi NM (2006) Pattern recognition and machine learning, vol. 1, Springer, New York.
10. Cha SH (2007). Comprehensive Survey on Distance/Similarity Measures between probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, issue 4, vol. 1.
11. Chen, Y., Miao, D., Zhang, H. (2010) Neighborhood Outlier Detection, Expert Systems with Applications 37:8745-8749.
12. Dempster A., Laird N., Rubin D (1977) Maximum likelihood from incomplete data via the EM algorithm. J. Royal Stat. Soc. B, 39(1):1-38.
13. Elhamifar E, Vidal, R (2009) Sparse subspace clustering. Proc. IEEE Conf. Computer Vision and Pattern Recognition.

14. Gallant, I. S, (1993) Neural Network Learning and Expert Systems, MIT Press, ISBN: 9780262527897.
15. Gionis A, Hinnenburg A, Papadimitriou S, Tsaparas, P (2005) Dimension Induced Clustering. In KDD, ACM.
16. Goh A, Vidal R (2007) Segmenting motions of different types by unsupervised manifold clustering. Proc. IEEE Conf. Computer Vision and Pattern Recognition.
17. Guan Y, Bao J (2009) A cp intrusion detection strategy on cloud computing. International Symposium on Web Information Systems and Applications (WISA), 84–87.
18. He X, Papadopoulos C, Heidemann J, Mitra U, Riaz U (2009) Remote detection of bottleneck links using spectral and statistical methods. Computer Networks, 53:279-298.
19. Hudic A, Hecht T, Tauber M, Mauthe A, Caceres-Elvira S (20140 Towards Continuous Cloud Service Assurance for Critical Infrastructure IT. International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 175-182, doi: 10.1109/FiCloud.2014.36
20. Hussain A, Heidemann J, Papadopoulos C (2006) Identification of repeated denial of service attacks. Proceedings of the Conference on Computer Communications (INFOCOM), Barcelona, Spain.
21. Kapur JN (1989) Maximum-Entropy Models in Sciences and Engineering. New York: John Wiley.
22. Lee JH, Park MW, Eom JH, Chung TM (2011) Multi-level intrusion detection system and log management in cloud computing. 13th International Conference on Advanced Communication Technology (ICACT), IEEE, 552–555.
23. Leopold, H., Bleier, T., Skopik**,** F (2015) Cyber Attack Information System, Erfahrungen und Erkenntnisse aus der IKT-Sicherheitsforschung. Springer, ISBN 978-3-662-44306-4.
24. Liang, J., Li, R., Qian, Y. (2012) Distance: A More Comprehensible Perspective for Measure in Rough Set Theory. In *Knowledge-Based Systems*, vol. 27, 126-136.

25.Lin J (2015) MonArch: Scalable Monitoring and Analytics for Visibility and Insights in Virtualized Heterogeneous Cloud Infrastructure, The Edward S. Rogers Sr. Department of Electrical & Computer Engineering, University of Toronto, Thesis Master of Applied Science.

26.Löf A, Nelson R (2010) Comparing anomaly detection methods in computer networks. Proceedings of the Fifth International Conference on Internet Monitoring and Protection - ICIMP'10, Washington DC, USA, IEEE Computer Society, 7-10.

27.Ma Y, Yang A, Derksen H, Fossum R (2008) Estimation of subspace arrangements with applications in modeling and segmenting mixed data. SIAM Rev., 50(3): 413-458.

28.Munda, Nardo (2003) On the methodological foundations of composite indicators used for ranking countries. OECD/JRC Workshop on Composite Indicators of Country Performance, OECD: Ispra, Italy.

29.Padhy RP, Patra MR, Satapathy SC (2011) X-as-a-Service: Cloud Computing with Google App Engine, Amazon Web Services, Microsoft Azure and Force.com. International Journal of Computer Science and Telecommunications, Volume 2, Issue 9.

30.Parsons L, Haque E, Liu H (2004) Subspace clustering for high dimensional data: A review," ACM SIGKDD Explor. Newslett., vol. 6(1): 90-105.

31.Pascoal C, Oliveira R, Valadas R, Filzmoser PF, Salvador P, Pacheco A (2012) Robust feature selection and robust PCA for internet traffic anomaly detection. IEEE INFOCOM, vol. 2012, 1755-1763.

32.Patcha A, Park JM (2007) An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks, 51(12):3448-3470.

33.Pawlak, Z. (1996) Rough Sets, Rough Relations and Rough Functions. In *Fundamenta Informaticae*, vol. 27, issue 2-3, 103-108.

34.Pedrycz, W., Bargiela, A. (2012) An Optimization of Allocation of Information Granularity in the Interpretation of Data Structures: Toward Granularity Fuzzy Clustering. In IEEE Trans. On Systems, Man and Cybernetics-Part B: Cybernetics, vol. 42, no. 3, 582-590.

35.Rabenoro T, Lacaille J, Cottrell M, Rossi F (2014) Anomaly Detection Based on Aggregation of Indicators, Proceedings of Benelearn.

36.Rhodes BC, Mahaffey JA, Cannady JD (2000) Multiple Self-Organizing Maps for Intrusion Detection. Proceedings of the 23rd National Information Systems Security Conference.

37.Rutkowska J. (2006) Subverting vista kernel for fun and profit. Black Hat Talk.

38.Shirazi S, Simpson S, Marnerides A., Watson M., Mauthe A, Hutchison D (2014) Assessing the impact of intra-cloud live migration on anomaly detection. IEEE 3rd International Conference on Cloud Networking (CloudNet), 52–57.

39.Shirazi, N., Simpson, S., Oechsner, S., Mauthe, A., Hutchison, D. (2015) A Framework For Resilience Management in the Cloud, Electrotechnik & Informationstechnik, 132/2:132/2 122-132. DOI 10.1007/s005002-015-0290-9.

40.Shirazi, S. N. U. H., Simpson, S., Gouglidis, A., Mauthe, A. U., and Hutchison, D. (2016) Anomaly Detection in the Cloud using Data Density, IEEE Internation Conference on Cloud Computing, At San Franscisco, USA.

41.Stanfill, C and Waltz, B (1986) Towards Memory Based Reasoning, Communications of the ACM, 29:1213-1228

42.Takács G et al (2008) Matrix factorization and neighbor based algorithms for the Netflix prize problem. Proceedings ACM Conference on Recommender Systems, Lausanne, Switzerland, 267-274.

43.Tan Y et al. (2012) Prepare: Predictive performance anomaly prevention for virtualized cloud systems. IEEE ICDCS, DOI: 10.1109/ICDCS.2012.65.

44.Thatte G, Mitra U. (2011) Parametric Methods for Anomaly Detection in Aggregate Traffic and John Heidemann, IEEE/ACM Transactions on Networking, 19(2):512-525.

45.Tipping M, Bishop C (1999) Mixture of probabilistic principal component analyzers. Neural Comput., 11(2):443-482.

46.Vidal R (2011) Subspace Clustering: Applications in motion segmentation and face clustering. IEEE Signal Processing Magazine, vol. 28, number 2, 52-68, doi :10.1109/MSP.2010.939739.

47.Vidal R, Ma Y, Sastry S (2005) Generalized principal component analysis (GPCA). IEEE Trans. Pattern Anal. Machine Intell., 27(12): 1-15.

48.Wald A (1947) Sequential Analysis. John Wiley, New York.

49.Wu N, Zhang J (2006). Factor-analysis based anomaly detection and clustering. Decision Support Systems, 42(1): 375-389.

50.Zhang H (2004) The optimality of naïve bayes. AA, 1(2):3

# Tables

**Table 1.** Typical cyber-attack methods that generate anomaly states within IT systems

| Anomaly | Definition |
|---|---|
| ALPHA | Unusually high rate point-to-point byte transfer[1] |
| DOS, DDOS | (Distributed) Denial of service attack against a single victim |
| FLASH CROWD | Unusually large demand for resource/service emerging from common set of sources |
| SCAN | Scanning a host for a vulnerable point (port scan) or scanning the network for a target port (network scan) |
| WORM | Self-propagating code that spreads across a network |
| POINT to MULTIPONT | Distribution of content from one server to many servers |
| OUTAGE | Equipment related events that decrease traffic exchange by an Origin-Destination pair |
| INGRESS-SHIFT | Customer shifts traffic from one ingress point to another |

[1] Alpha flows are high-rate flows from a single source to a single destination which account for a dominant fraction of byte traffic. These can be distinguished from DoS and DDoS attacks, which feature a dominant fraction of packet or flow traffic, all to a single destination.

**Table 2a.** Distances between the three (3) partially overlapping distributions in **Fig. 3a** used as an example (mean values and variance intrerval of ten instances of ordered sequences featuring forty vectors – mean distance between instances of the same distribution should be the lowest)

| | **Distribution** *A* | | | **Distribution** *B* | | | **Distribution** *C* | | |
|---|---|---|---|---|---|---|---|---|---|
| **Distribution** *A* | 2,721 [2,605–2,887] | 1,139 [1,080-1,220] | 1,734 [1,556-2,030] | 10,550 [9,026–11,333] | 4,747 [4,015-5,102] | 10,512 [8,989-11,309] | 12,420 [11,220-13,304] | 5,597 [4,952 -5,959] | 12,418 [11,214-13,304] |
| **Distribution** *B* | 10,550 [9,026–11,333] | 4,747 [4,015 - 5,102] | 10,512 [8,989-11,309] | 8,958 [8,531-9,386] | 3,690 [3,476-3,846] | 3,204 [2,882-3,987] | 9,102 [8,321-9,845] | 3,944 [3,524 -4,231] | 7,813 [6,861-8,527] |
| **Distribution** *C* | 12,420 [11,220-13,304] | 5,597 [4,952 - 5,959] | 12,418 [11,214-13,304] | 9,102 [8,321-9,845] | 3,944 [3,524 -4,231] | 7,813 [6,861-8,527] | 7,136 [6,442-7,711] | 2,900 [2,626-3,166] | 4,416 [4,114-5,280] |
| | sum of all elements | trace | thresholded | sum of all elements | trace | thresholded | sum of all elements | trace | thresholded |

**Table 2b.** Distances between the three (3) partially overlapping distributions in **Fig. 3a** used as an example (rough set fuzzification of the cross distance matrix using elements of low {1-16}, medium {12-28} and high order {24-40} intervals within the brackets are defined upon the sum of all distances between pairs of elements in relative subsets of both distributions)

| | **Distribution** *A* | **Distribution** *B* | **Distribution** *C* |
|---|---|---|---|
| **Distribution** *A*<br>$\mu_A^{low}$ ([0.67 0.84])=1 $\mu_A^{low,medium}$ ([1.08 1.30])=1 $\mu_A^{low,high}$ ([1.72 2.02])=1<br>$\mu_A^{medium}$ ([1.22 1.52])=1 $\mu_A^{medium,high}$ ([1.83 2.18])=1<br>$\mu_A^{high}$ ([2.13 2.51])=1 | 0.5 0.4 0<br>0.1 0.1 0<br>0.3 0.2 0.1 | 0 1 1<br>1 1 1<br>1 1 1 | 0 1 1<br>1 1 1<br>1 1 1 |
| **Distribution** *B*<br>$\mu_B^{low}$ ([2.00 3.11])=1 $\mu_B^{low,medium}$ ([3.33 4.54])=1 $\mu_B^{low,high}$ ([5.27 7.17])=1<br>$\mu_B^{medium}$ ([3.81 5.28])=1 $\mu_B^{medium,high}$ ([5.54 7.66])=1<br>$\mu_B^{high}$ ([6.09 8.85])=1 | | 0 0 0<br>0 0 0<br>0 0 0 | 0 1 1<br>1 1 1<br>1 1 1 |
| **Distribution** *C*<br>$\mu_C^{low}$ ([1.47 2.03])=1 $\mu_C^{low,medium}$ ([2.45 3.01])=1 $\mu_C^{low,high}$ ([3.93 5.24])=1<br>$\mu_C^{medium}$ ([2.85 3.57])=1 $\mu_C^{medium,high}$ ([4.12 5.49])=1<br>$\mu_C^{high}$ ([2.13 4.71])=1 | | | 0 0.8 0.6<br>0.5 0.6 0.5<br>0.3 0.2 0.1 |

**Table 3.** Feature histograms for a sliding time window

| | Centers of histogram bins |
|---|---|
| Number of packets x10,000) | 404 984 1,563 2,142 2,722 3,301 3,880 4,460 5,039 5,618 6,198 6,777 7,356 7,936 8,515 9,094 9,674 10,253 10,832 11,412 (20 bins) |
| Number of bytes | 3,787 4,975 6,162 7,350 8,538 9,726 10,914 12,102 13,289 14,477 15,665 16,853 18,040 19,228 20,416 (15 bins) |
| Number of active flows in time stamp | 283.5 354.4 425.3 496.2 567.1 638 709 779.9 850.8 921.7 992.6 1,063.5 (12 bins) |
| Entropy of source IP address distribution | 2.4423 2.6608 2.8792 3.0978 (4 bins) |
| Entropy of destination IP address distribution | 3.0266 3.6158 4.2050 4.7942 5.3834 (5 bins) |
| Entropy of source port distribution | 2.9906 3.5558 4.1210 4.6862 5.2514 (5 bins) |
| Entropy of destination port distribution | 3.1532 3.8196 4.4860 5.1524 5.8188 (5 bins) |
| Entropy of packet size distribution | 3.1231 3.7693 4.4155 5.0617 5.7079 (5 bins) |

**Table 4.** Anomaly detection statistics using EM-GMM

| **Background traffic with high anomaly intensity** (host port scan) | **Without migration** | **With migration** |
|---|---|---|
| Error rate | 55.89% | 55.06% |
| Detection rate | 44.11% | 44.94% |
| **Background traffic with low anomaly intensity** (host port scan) | | |
| Error rate | 50.13% | 50.33% |
| Detection rate | 49.87% | 49.67% |
| **Background traffic with low anomaly intensity** (denial of service-anomaly directed at migrated host) | | |
| Error rate | 69.42% | 0.4887 |
| Detection rate | 30.58% | 51.13% |
| **Background traffic with low anomaly intensity** (denial of service-anomaly directed at static host) | | |
| Error rate | 35.41% | 26.76% |
| Detection rate | 64.59% | 73.24% |
| **Background traffic with high anomaly intensity** (denial of service-anomaly directed at migrated host) | | |
| Error rate | 57.26% | 26.79% |
| Detection rate | 42.74% | 73.21% |
| **Background traffic with high anomaly intensity** (denial of service-anomaly directed at static host) | | |
| Error rate | 55.56% | 53.71% |
| Detection rate | 44.44% | 46.29% |
| **Background traffic with high anomaly intensity** (network port scan) | | |
| Error rate | 13.40% | 27.44% |
| Detection rate | 86.60% | 72.56% |
| **Background traffic with low anomaly intensity** (network port scan) | | |
| Error rate | 48.81% | 35.04% |
| Detection rate | 51.19% | 64.96% |

**Table 5.** Anomaly detection using local data densities (typical values per cluster) vs the number of clusters for the method depicted in **Fig. 2.d** (low values indicate anomalies)

| | One cluster | Two clusters | Three clusters | Four clusters |
|---|---|---|---|---|
| Normal operation | 0.9922 | | | |
| | 0.9954 | 0.9928 | | |
| | 0.9964 | 0.9921 | 0.9983 | |
| | 0.9978 | 0.9961 | 0.9986 | 0.9918 |
| Insertion of anomaly | 0.0091 | | | |
| | 0.9780 | 0.0064 | | |
| | 0.9752 | 0.0056 | 0.9854 | |
| | 0.9855 | 0.9712 | 0.9749 | 0.0048 |

## Figures

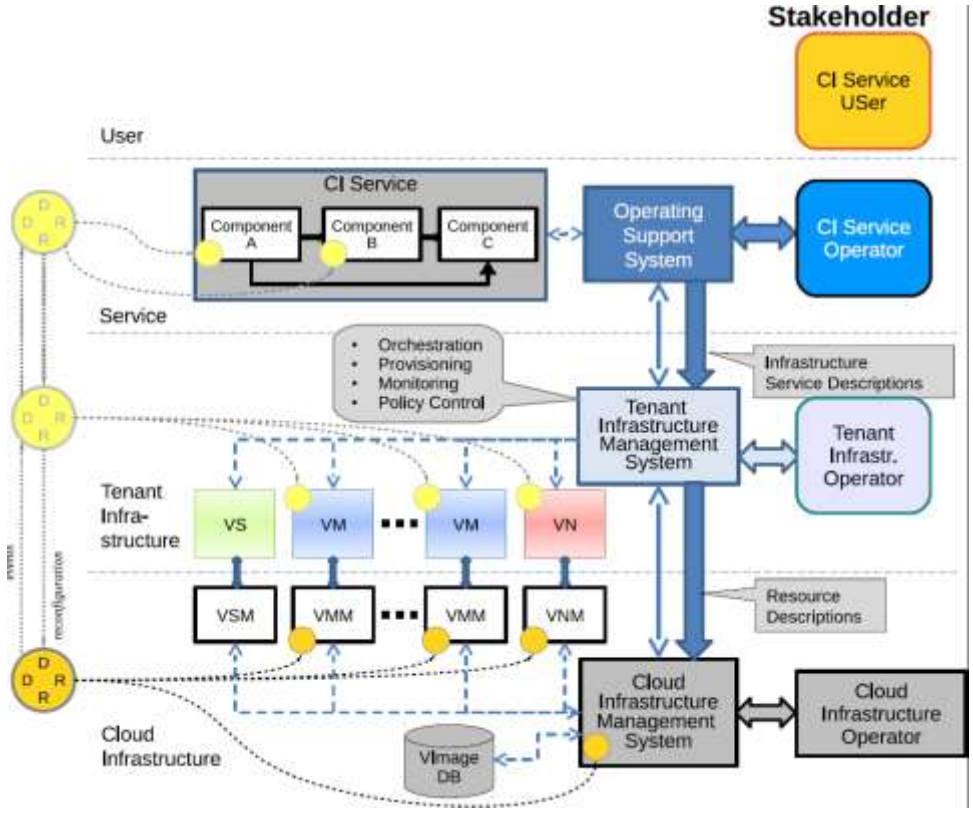


**Fig. 1.a** Reference architectural framework for cloud anomaly detection (SECCRIT)

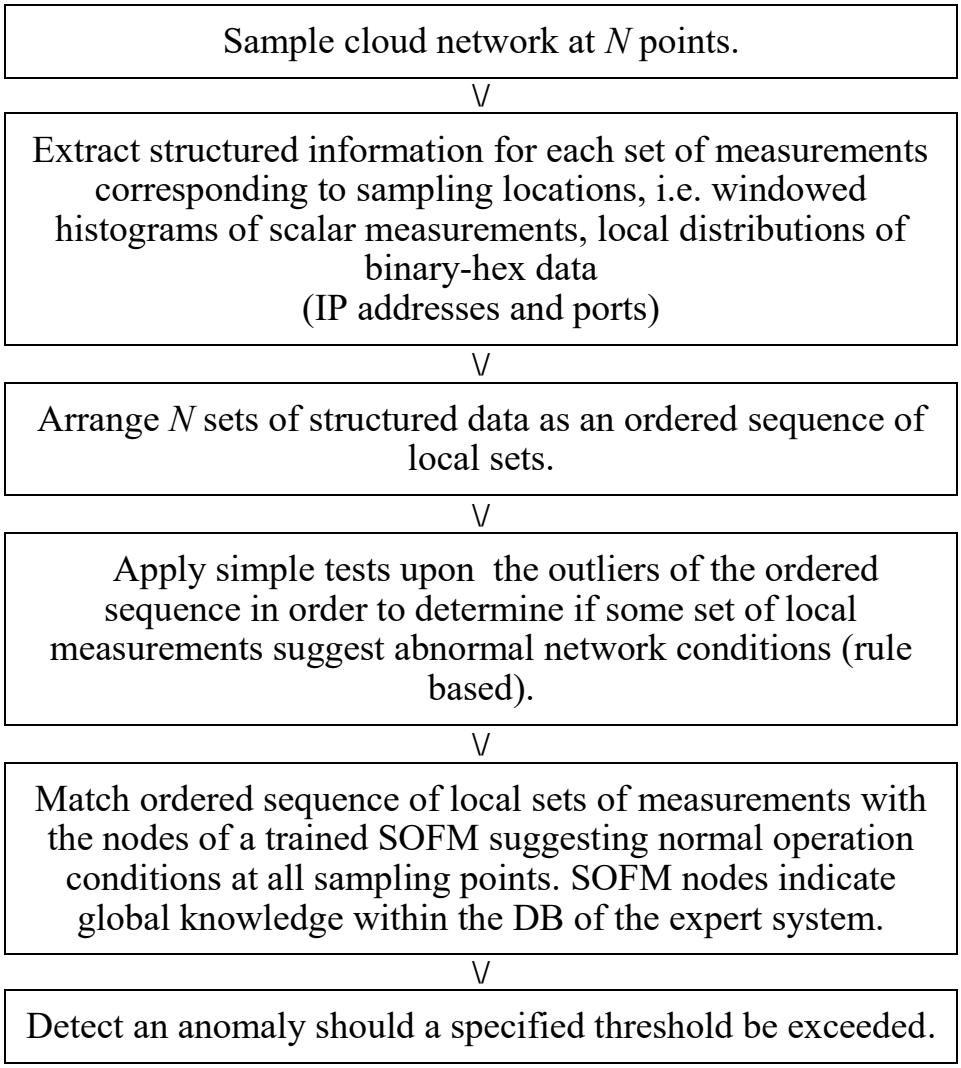


**Fig. 1.b** Block diagram of the proposed approach based upon ordered local sets of measurements per cluster site, SOFM clustering and thresholding

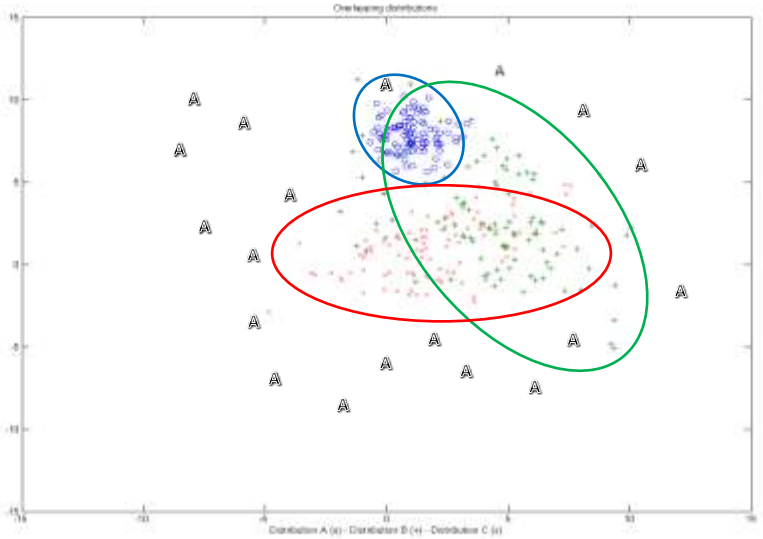

**Fig. 2.a** A mixture of Gaussians may be used to cluster measurements indicating normal network operation - Anomalies are detected as outliers using log likelihood distance

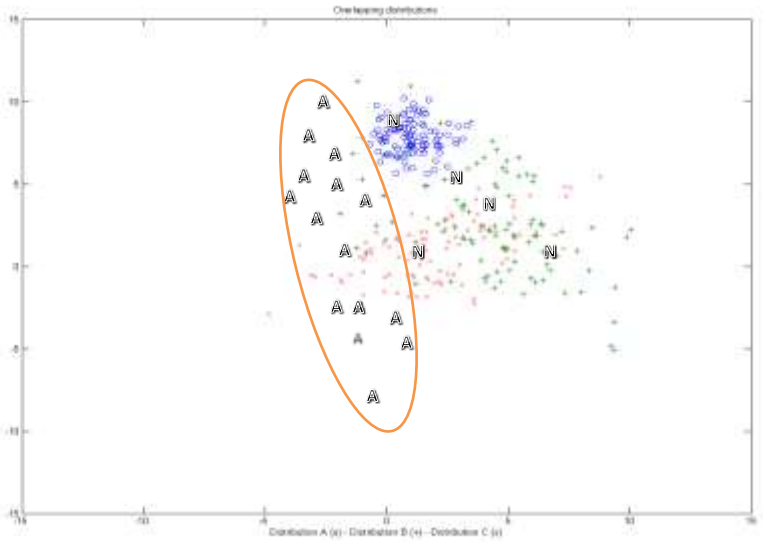

**Fig. 2.b** Local data densities of groups of points indicating anomalies (denoted as A) with respect to cluster centers of measurements indicating normal (denoted as N) operation (which is estimated as $d_a = \frac{1}{1+\frac{1}{N_a}\sum_{i=1}^{N_a}\left\|x_a - c_n^{closest}\right\|^2}$) is low

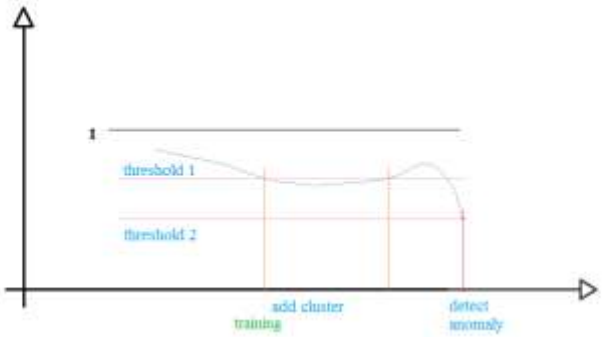


**Fig. 2.c** Anomaly detection (using local data densities of groups of points (training is carried out during normal network operation) – Thresholds for index $d_a$ which ranges from zero to one

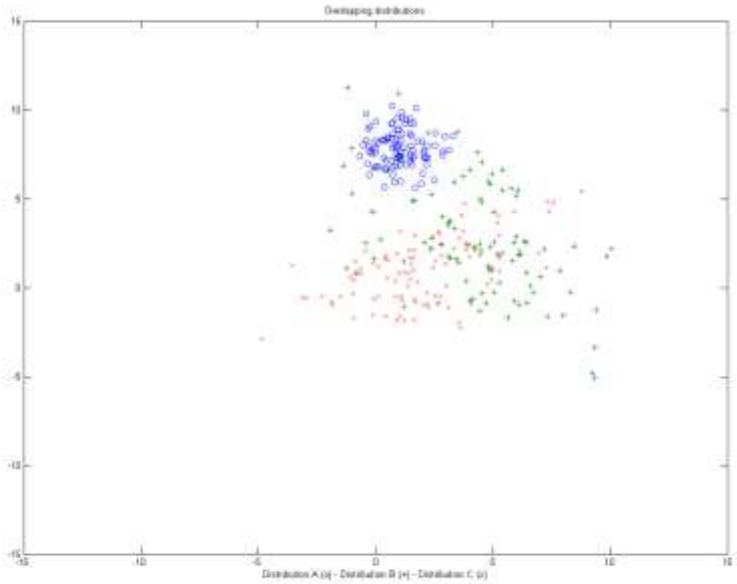

**Fig. 3a Reduced/aggregate ordering of features along with a measure can be used to represent a local cluster**

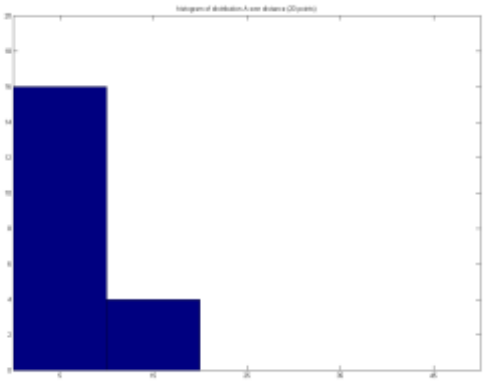

**Fig. 3b Histogram over distance for Distribution A (o) Fig. 3a (20 points)**

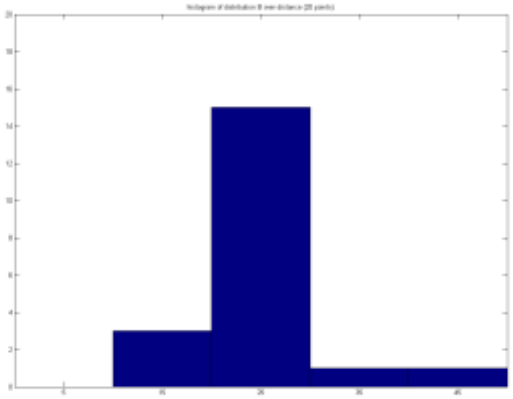

**Fig. 3c Histogram over distance for Distribution B (+) of Fig. 2a (20 points)**

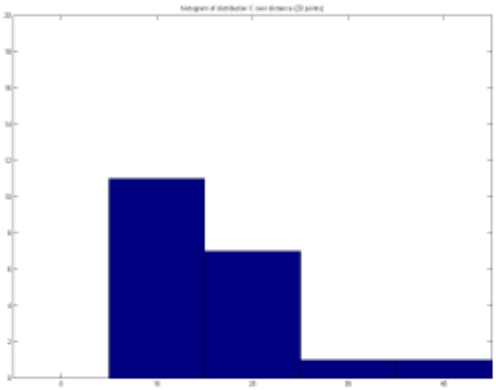

**Fig. 3d Histogram over distance for Distribution C (x) of Fig. 3a (20 points)**

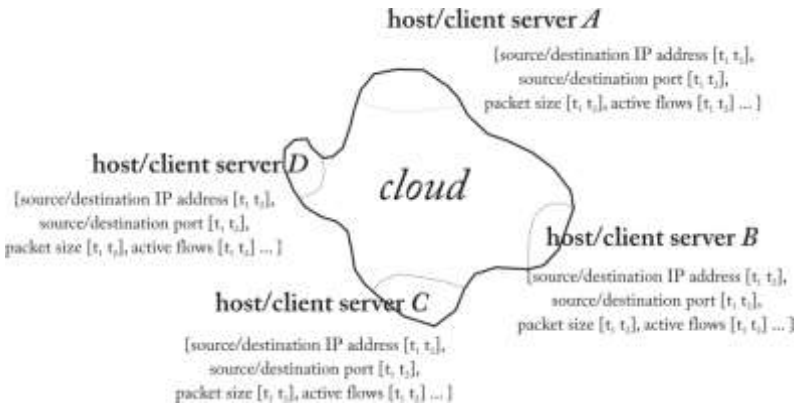


**Fig. 4** Sampling binary and vector features in the cloud

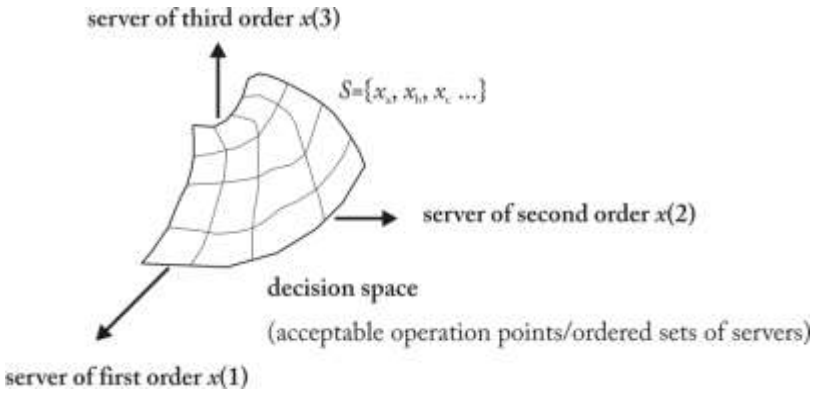


**Fig. 5** SOFM for ordered sets - Each node consists of some ordered set of hosts

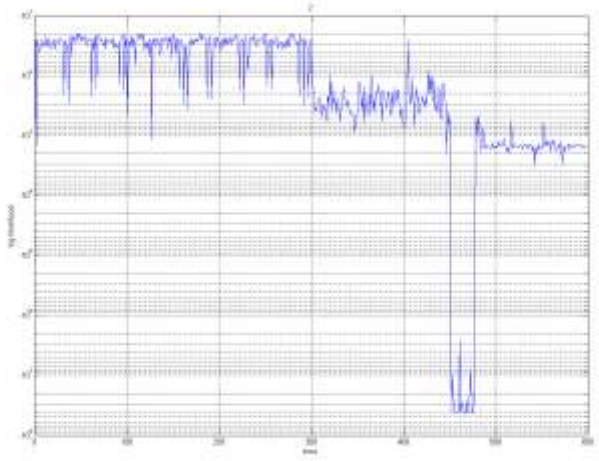

**Fig. 6.a** Log likelihood (EM-GMM for two Gaussians) vs time stamp (an attack is introduced during Virtual Machine migration after t=300)

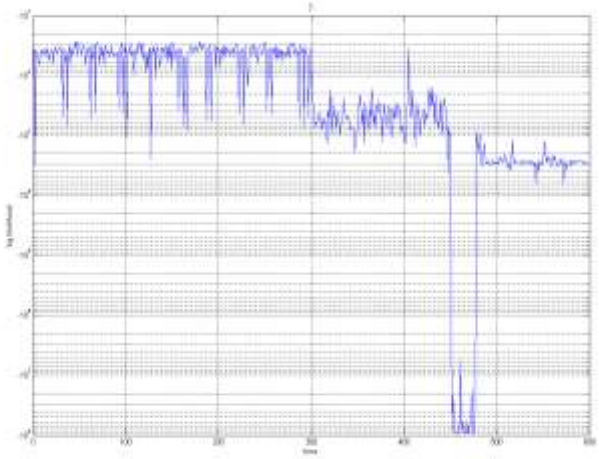

**Fig. 6.b** Log likelihood (EM-GMM for three Gaussians) vs time stamp (an attack is introduced during Virtual Machine migration after t=300)

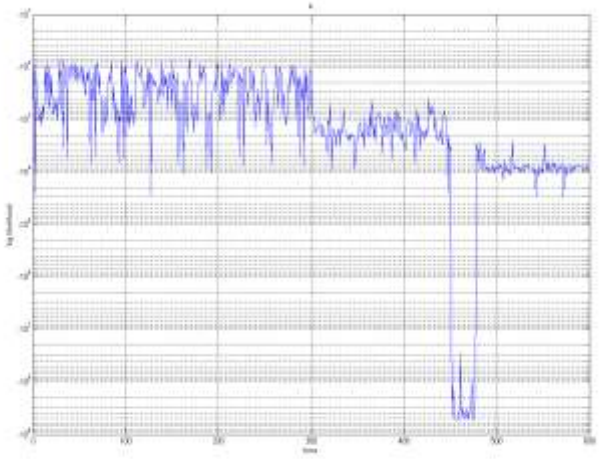

**Fig. 6.c** Log likelihood (EM-GMM for two, three and four Gaussians) vs time stamp (an attack is introduced during Virtual Machine migration after t=300)

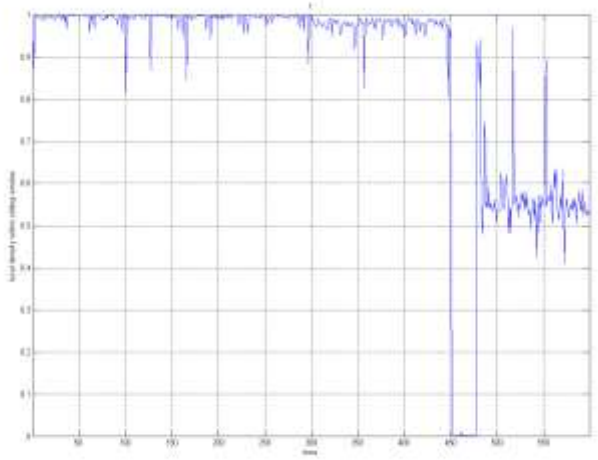

**Fig. 6.d** Local data density over timestamp (for one measurement and one cluster) -
Clusters are estimated using the data density criterion (an attack is introduced during Virtual Machine migration after t=300)

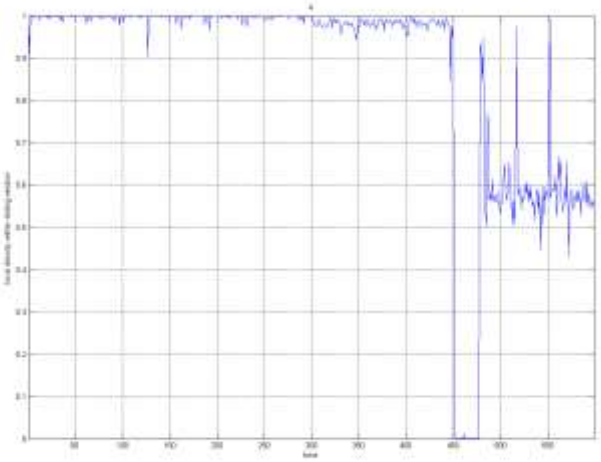

**Fig. 6.e** Local data density over timestamp (for one measurement and four clusters) -
Clusters are estimated using the data density criterion (an attack is introduced during Virtual Machine migration after t=300)

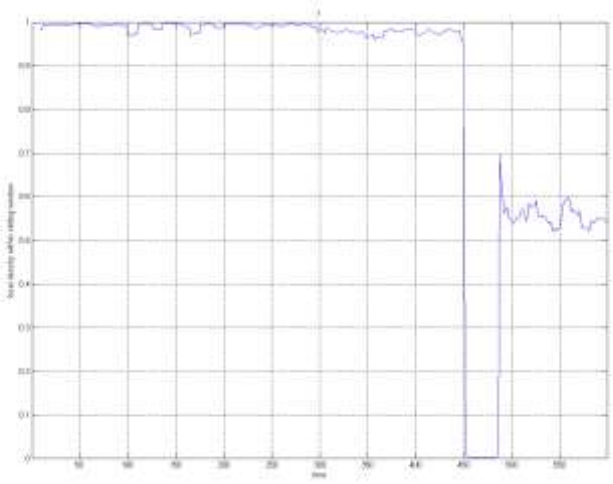

**Fig. 6.f** Local data density over timestamp (sliding window of ten timestamps and one cluster) - Clusters are estimated using the data density criterion (an attack is introduced during Virtual Machine migration after t=300)

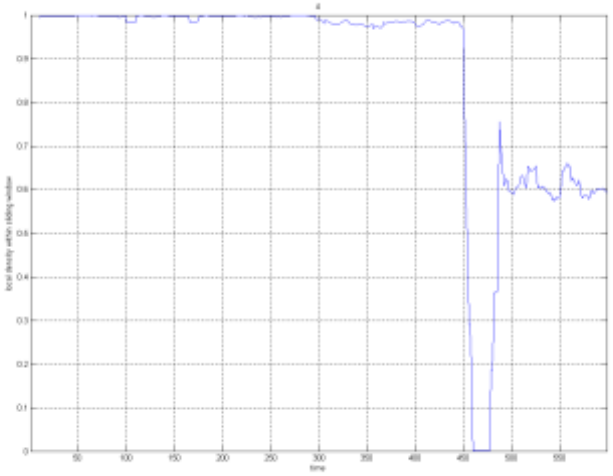

**Fig. 6.g** Local data density over timestamp (sliding window of ten timestamps and four clusters) - Clusters are estimated using the data density criterion (an attack is introduced during Virtual Machine migration after t=300)

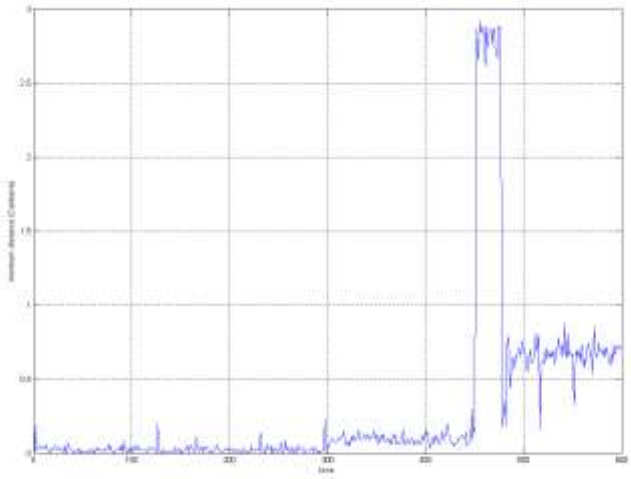

**Fig. 6.h** Closest Canberra distances from a 10x10 SOFM used to cluster single vectors of measurements (an attack is introduced during Virtual Machine migration after t=300)

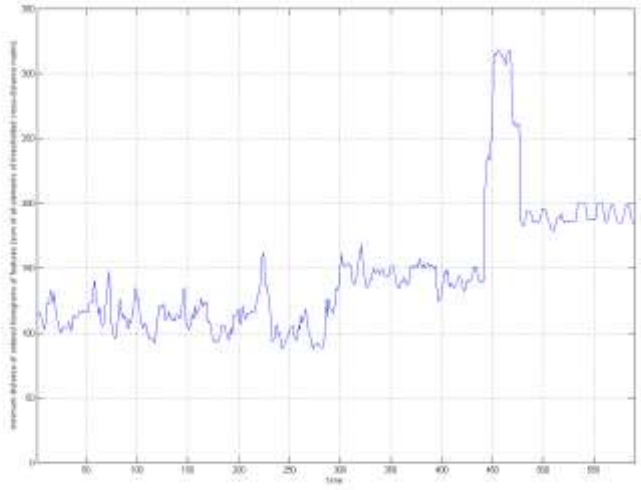

**Fig. 6.i** Outlier distance (proposed method) vs time stamp (an attack is introduced during Virtual Machine migration after t=300)

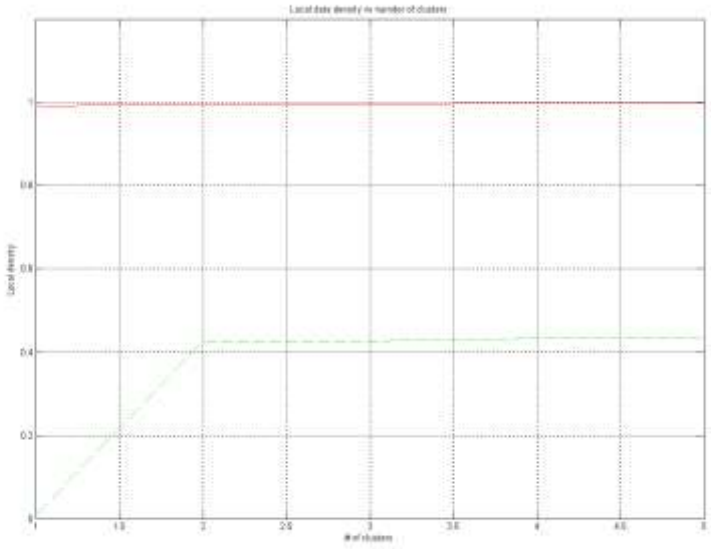

**Fig. 7** Average values of local data density per cluster for normal (solid line) as well as abnormal operation (dashed line) for all measurements

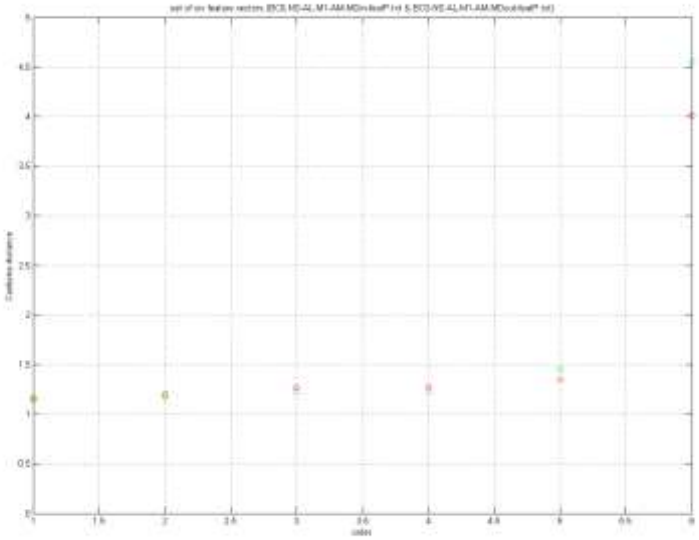

**Fig. 8a** Distances vs rank of elements – 6$^{th}$ rank element corresponds to anomaly (single-stamp feature vector for inward - red circles - and outward - green circles - migration)

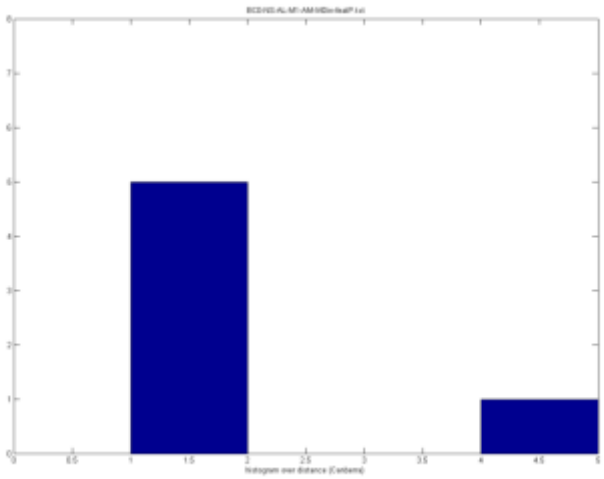

**Fig. 8b** Number of objects (sets of local measurements) over aggregate distance as a histogram – 6th rank corresponds to anomaly (single stamp feature vectors, background traffic with low anomaly intensity - net scan - and inward migration)

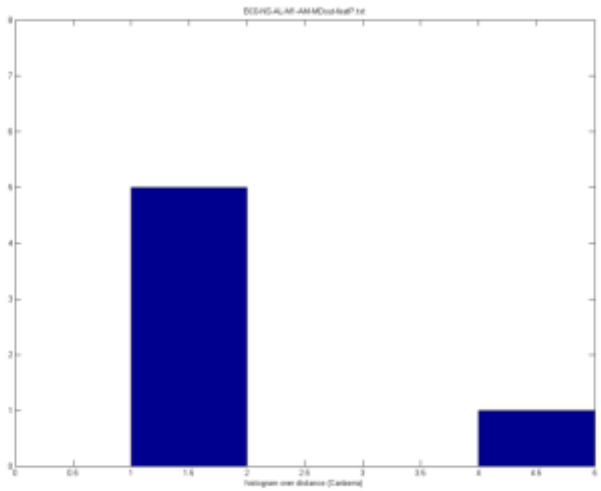

**Fig. 8c** Number of objects (sets of local measurements) over aggregate distance as a histogram – 6th rank corresponds to anomaly (single stamp feature vectors, background traffic with low anomaly intensity - net scan - and outward migration)

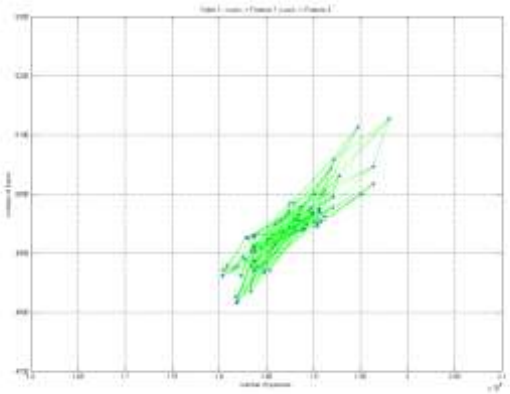

**Fig. 9a** SOM projected upon the plane of the # of bytes vs the # of packets for 1st rank vector

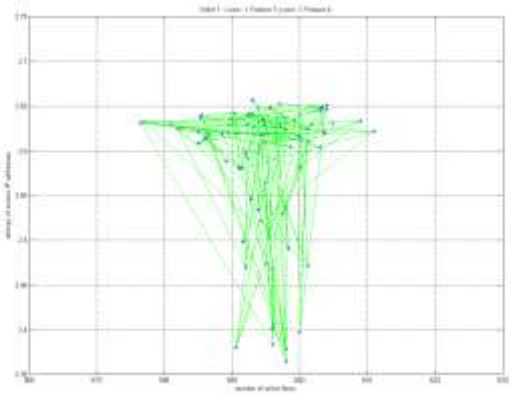

**Fig. 9b** SOM projected upon the plane of the entropy of source IP addresses vs the # of active flows for 1st rank vector

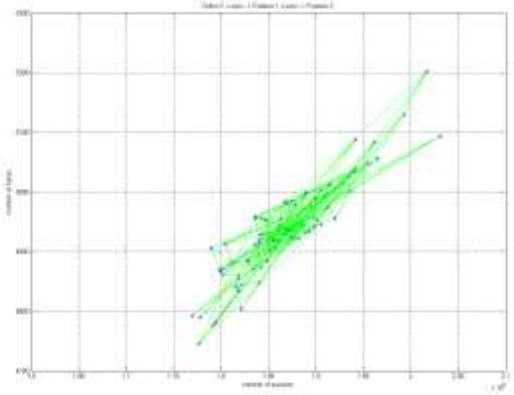

**Fig. 9c** SOM projected upon the plane of the # of bytes vs the # of packets for 2nd rank vector

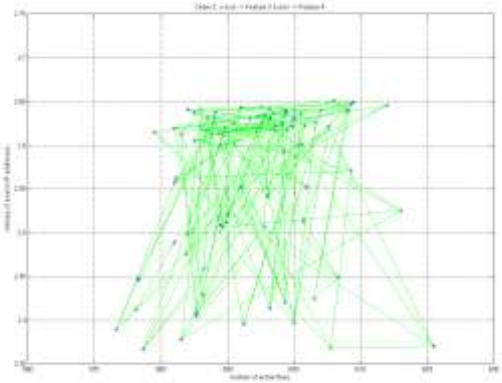

**Fig. 9d** SOM projected upon the plane of the entropy of source IP addresses vs the # of active flows for 2nd rank vector

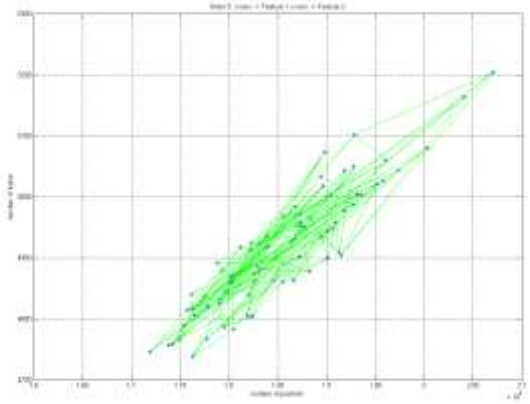

**Fig. 9e** SOM projected upon the plane of the # of bytes vs the # of packets for 3rd rank vector

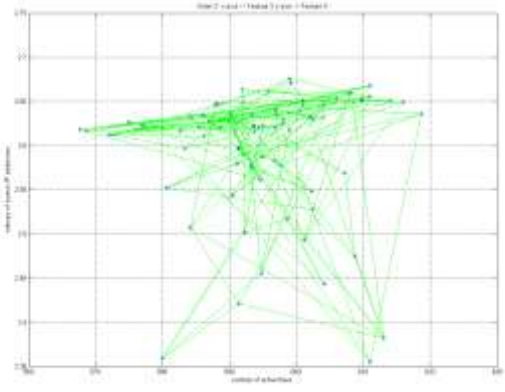

**Fig. 9f** SOM projected upon the plane of the entropy of source IP addresses vs the # of active flows for 3rd rank vector

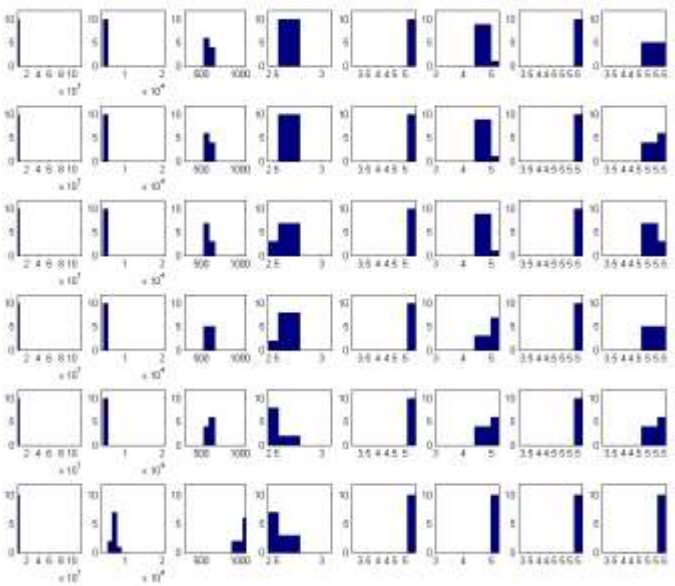


**Fig. 10a** Histograms of features for a ten (10) stamp sliding window for ordered sets of local measurements (background traffic with low anomaly intensity - net scan - and inward migration)

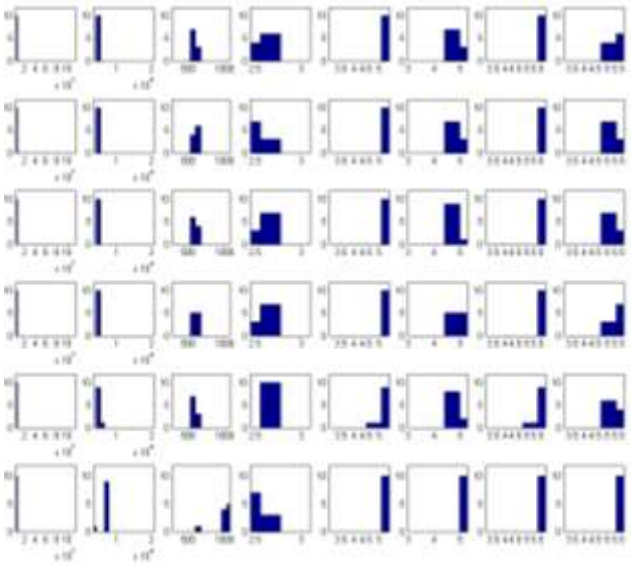


**Fig. 10b** Histograms of features for a ten (10) stamp sliding window for ordered sets of local measurements (background traffic with low anomaly intensity - net scan - and outward migration) from left to right # of packets, # of bytes, # of active flows, entropy of source IP addresses, entropy of destination IP addresses, entropy of source ports, entropy of destination ports, entropy of packet sizes

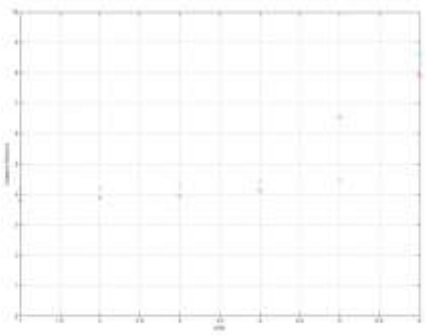

**Fig. 11a** Distances vs rank of elements – 6th rank corresponds to anomaly
(for the histogram feature vectors depicted in Fig. 9.a - red circles corresponding to inward migration - and Fig. 9.b - green circles corresponding to outward migration)

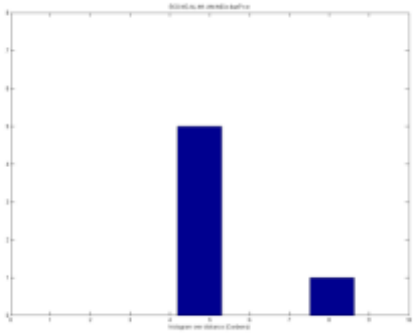

**Fig. 11b** Distribution of histogram feature vectors (depicted in Fig. 9.a) over ordering distance (NS inward migration)

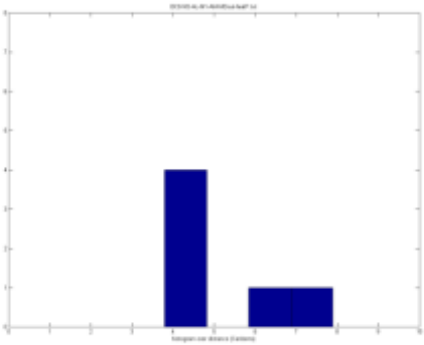

**Fig. 11c** Distribution of histogram feature vectors (depicted in Fig. 9.b) over ordering distance (NS outward migration)

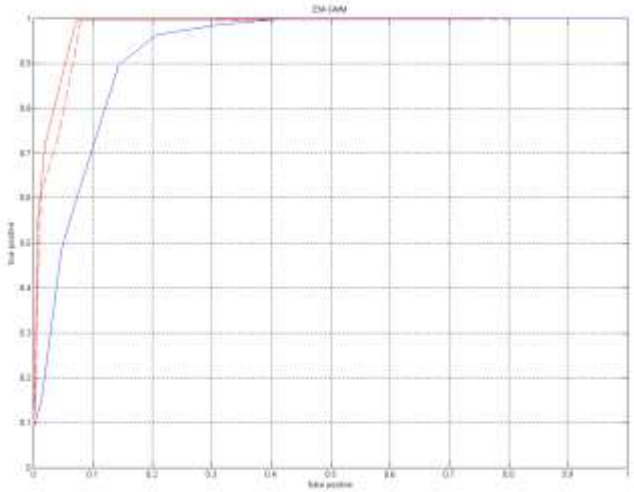

**Fig. 12.a** True Positive Rate (TPR) vs False Positive Rate (FPR) for low intensity net scan (EM-GMM using EM-GMM for two (solid red), three (dashed red) and four (solid blue) Gaussians in **Fig. 6.a** to **Fig. 6.c**)

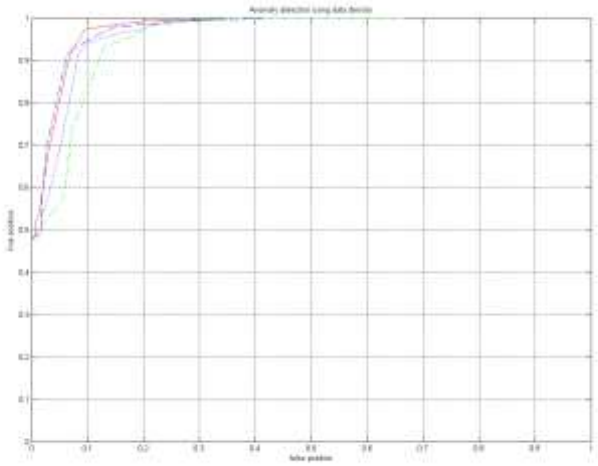

**Fig. 12.b** True Positive Rate *vs* False Positive Rate for anomaly detection using data density for one measurement-timestamp - low intensity net scan (solid red line four clusters, dashed blue three clusters, dotted blue two clusters, dashed green one cluster)

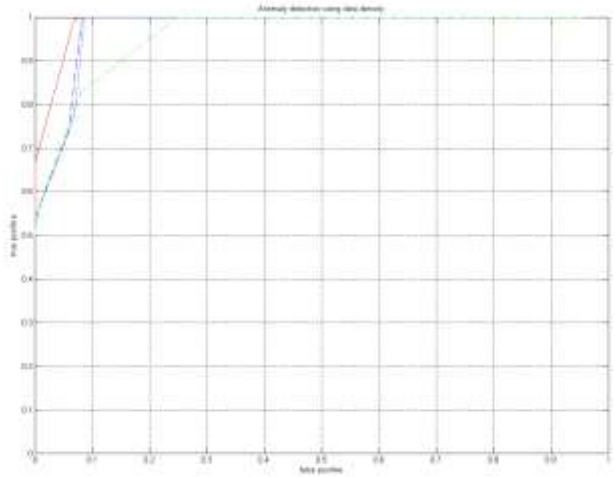

**Fig. 12.c** True Positive Rate *vs* False Positive Rate for anomaly detection using data density for a sliding window of ten timestamps - low intensity net scan (solid red line four clusters, dashed blue three clusters, dotted blue two clusters, dashed green one cluster)

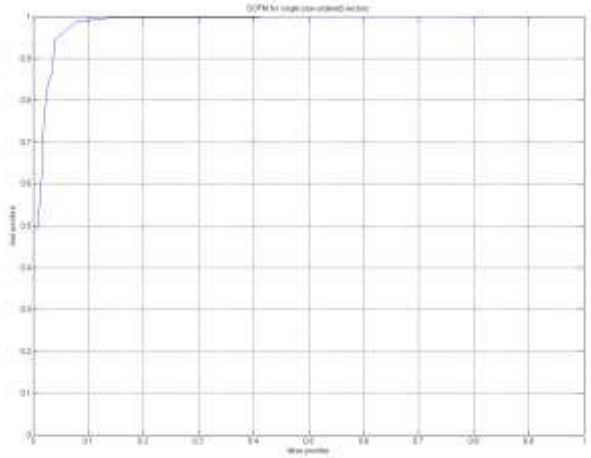

**Fig. 12.d** True Positive Rate (TPR) vs False Positive Rate (FPR) for anomaly detection using a 10x10 SOFM to cluster single vectors of measurements - low intensity net scan (an attack is introduced during Virtual Machine migration after t=300, closest distances from the SOFM using the Canberra distance are given in **Fig. 6.h)**

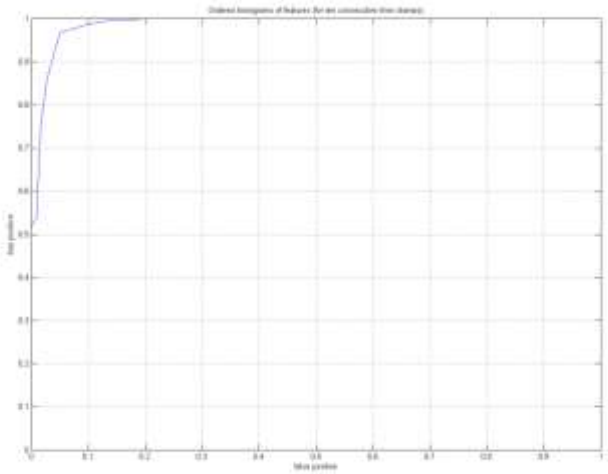

**Fig. 12.e** True Positive Rate (TPR) vs False Positive Rate (FPR) for anomaly detection using ordered vectors of histograms - low intensity net scan (for threshold values ranging from 90 to 190 in **Fig. 6.i**)

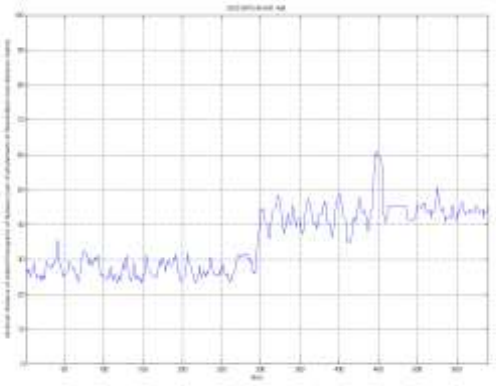

**Fig. 13.a** Outlier distance for anomaly detection (high intensity *Network Port Scan*) using ordered vectors of histograms over a sliding window of then timestamps (a set of three ordered vectors for a 4x4 neural network)

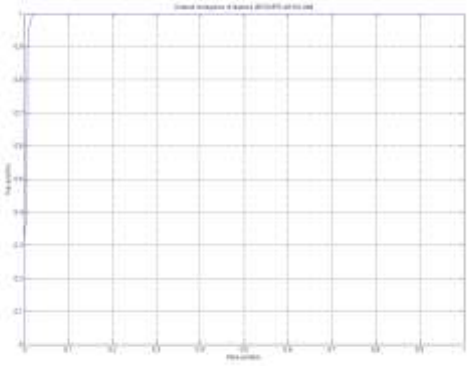

**Fig. 13.b** True Positive Rate (TPR) vs False Positive Rate (FPR) for anomaly detection (high intensity *Network Port Scan*) using ordered vectors of histograms over a sliding window of then timestamps (a set of three ordered vectors for a 4x4 neural network)

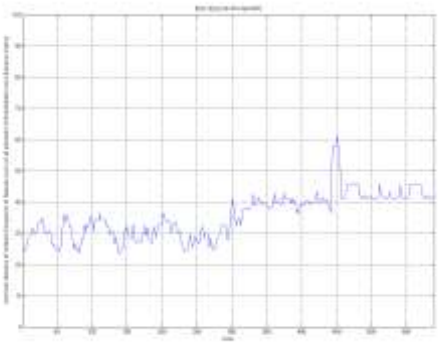

**Fig. 13.c** Outlier distance for anomaly detection (high intensity UPD *Denial of Dervice*) using ordered vectors of histograms over a sliding window of then timestamps (a set of three ordered vectors for a 4x4 neural network)

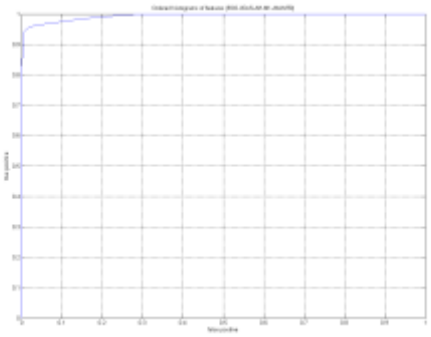

**Fig. 13.d** True Positive Rate (TPR) vs False Positive Rate (FPR) for anomaly detection (high intensity UDP *Denial of Service*) using ordered vectors of histograms over a sliding window of then timestamps (a set of three ordered vectors for a 4x4 neural network)