ONE-CLASS LEARNING WITH AN AUTOENCODER BASED SELF ORGANIZING MAP

by

Deepthi Rajashekar

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
March 2017

Table of Contents

List of Tables

List of Figures

Abstract

Cyber-security defense techniques have begun to transcend from one-size fits all approach to personalized solutions. These techniques factor in user autonomy by monitoring the temporal and spatial changes in the user's behaviour. With time, such a system is known to develop a comfort to the user's interaction with the device. The motivation of this research is to enable a device to differentiate its owner from another user, by discovering behavioural patterns in the contextual and physiological information of smartphone usage. Naturally, this poses a one-class learning constraint. The proposed framework quantifies: (i) the dissimilarity in behaviours among any two users; (ii) the exclusivity of each users behaviour (inclass) from the world (outclass). The crucial aspect of this framework is to construct a representation of the most important properties of each user. To this end, the utility of a feed forward multilayer perceptron (MLP) in identifying an encoding that rebuilds the input data with least loss is examined. The claim is that such an encoding step poses improved data representations prior to clustering, a data description technique. However, both the encoding and clustering steps respect the one-class learning restriction *i.e.,* relative to a single user. The evaluations on publicly available smartphone datasets, show that the resulting (user specific) behavioural models are capable of uniquely identifying each user. In particular, encoded contextual information are better anchors to behaviour modelling in comparison with encoded physiological information of smartphone data.

List of Abbreviations

AE	Autoencdoer.
AESOM	Autoencoder based Self Organizing Map.
BMU	Best Matching Unit.
ER	Exclusivity Rate.
GCU	Glasgow Caledonian University.
ML	Machine Learning.
MLP	Multilayer Perceptron.
MSE	Mean Squared Error.
NMI	Normalized Mutual Information.
PCI	Principal Component Initialization.
SOM	Self Organizing map.
SVM	Support Vector Machine.

List of Symbols

$Cmax_i$	Radius of the i^{th} cluster C_i.
$\mathbf{D}$	Decoding function.
$\mathbf{E}$	Encoding function.
$\mathcal{K}$	Number of clusters.
$\mathcal{N}$	Number of input samples.
$\mathcal{N}_c$	Coarse neighourhood function.
$\mathcal{N}_f$	Fine neighourhood function.
$\vec{b}$	Bias terms to the encoder.
$\vec{w}$	Weight vector of neurons in the encoder.

Acknowledgements

My sincere gratitude to my supervisors *Dr. Nur Zincir-Heywood* and *Dr. Malcolm I Heywood* for granting me the opportunity to higher education. My heartfelt thanks to Appa, Amma and Guru for shaping my self-worth regardless of my successes and despite my failures.

Chapter 1

Introduction

A significant growth has been witnessed in the ability of a mobile device to be connected to the internet, personal computers, other hand-held devices and presumably any computing device. Along with the increase in the available number of interactive devices, mobile phones now house small programs *a.k.a* applications that enable its user to connect to their banking server, search engines, cloud based storage repositories, email servers *etc.* This added computational ability, allows users to perform online banking, browsing, email, and remain connected to social media anywhere and at anytime. Thus improving the usability scope of mobile devices and, marking the transition from granular calling devices to 'smartphones'. As a consequence, smartphones host a huge amount of the user's personal, social, financial and private information. Andersson *et.al.,* confirm that users hold perceived value in using these smartphone application in relevance to [1]:

1. Convenience: banking, maps, radio, email *etc.*
2. Control: calendar, alarm, fitness trackers *etc.*
3. Motivation and inspiration: social media, news *etc.*
4. Monetary savings: expense trackers *etc.*
5. Entertainment: news media, gaming, social media, music *etc.*
6. Knowledge: news media, on-line course content *etc.*

Despite its implications to security management and risk mitigation, users find this arrangement convenient because their mobile phones are *with* them all the time. However, these personal devices pose security risks through which the user's privacy can be compromised. Malicious softwares *a.k.a.* malwares are one of the prominent ways in which user confidentiality and security is threatened. These programs are known to disrupt regular operations on the mobile device and/or misuse sensitive information such as user/device identity, by gaining unauthorized remote access to the smartphone [15].

A study confirming the usage of malwares as privacy attack vectors across smartphone platforms *viz.,* Android, iOS, Blackberry, Symbian and Windows, also concludes that successful malwares can be built by Bachelors of Computer Science students without any rigorous training. This is indicative of the ease with which implementation specific vulnerabilities of a smartphone can be exploited [35]. Another study on 30 randomly chosen applications in the Android market, found that 15 of them sent location information to advertisement servers, 7 sent the device ID to content servers and 2 sent the phone number to a remote server. This was done without obtaining direct or indirect consent from the user [14]. In the recent years, several researchers have aimed to detect malicious applications from legitimate ones, as discussed further below.

AppInspector is an automated tool to inspect and report applications involved in security breaches, privacy violation or both [16]. This tool uses system-wide taint tracking. Taint tracking is a technique of tracking the flow of data from its source to its sink. If the sink results in an outgoing network transmission to an unauthorized or illegal destination, then the application is said to violate user privacy. If the tracked data is collected without explicit user consent, then the application is said to breach security. This established technique was seconded by many other researchers [37, 14]. Rastogi *et.al.,* advance this technique with API and kernel-level monitoring to automatically detect privacy leaks and nefarious applications [42, 46]. However, most of those works have focused on free applications from the third-party application markets without specific categories. They have not investigated the potential for threat analysis of mobile malwares or applications related to malicious functionality and *high-risk user behaviours* [41]. This is important because an application can *be made to* misbehave as discussed further below.

Smartphone vulnerabilities not only arise from the external environment, but are also internal to the phone. These include implementation, incompatibility and user related vulnerabilities. Of these, a majority of security threats are known to occur in the 'userland', as a result of user unawareness (or ignorance) [24, 17]. Typical situations of user unawareness include [20]:

- loss of the smartphone,
- insider exploits through social engineering attacks

- improper smartphone settings (such as browser settings)
- connecting to un-trusted WiFi and/or installing applications from un-trusted sources

Each of these channels of exploitation lead to loss of crucial information assets. Mylonas *et.al.,* study quality metrics that assess the awareness and impact of security threats among Android and iOS users. To this end, subjects are grouped into security savvy and non-savvy categories. Their research goal was to establish whether, being aware of security threats and the corresponding mitigation strategies, forced users to interact cautiously with their devices. The study ends concluding that neither categories of users took protective measures to avert the threat of devices being exposed to unauthorized physical access and remote access [36]. Furthermore, unauthorized physical access demands efficient authentication strategies. Some prior works on mobile device security have focused on physical aspects and/or access control methods such as strong passwords, voice recognition, or fingerprints [43, 21]. However, such approaches do not *continue* to protect the mobile device user from malicious attacks in the post-authentication state. These solutions belong to the class of static defense strategies.

The limitation of these static defense strategies (automated malware detection methods is a part of them) is that they do not factor in user autonomy. That is, the interaction between the user and the device or applications in the device largely (also) depend on the the end user. However these methods assume that end user behaviours are, for the most part, *consistent*. This approach to security offers a 'one-size-fits-all' solution which lacks reliability and scalability. Alternatively, the social cognitive theory can be used to hypothesize unconventional security assurance techniques. Lawerence *et.al.,* posit that users are likely to approximate their peer's behaviours [40]. That is, victim users can be subjected to learning secure behaviour amidst a group of security practicing personnel'. Along similar lines, it can also be inferred that there exist patterns of similar thought processes amongst offenders (while the attacker builds his/her malware or during his interaction with the smartphone). *Then*, it is very likely that patterns of similar high-risk user behaviour will emerge. This notion opens a new possibility of analyzing the entire threat surface,

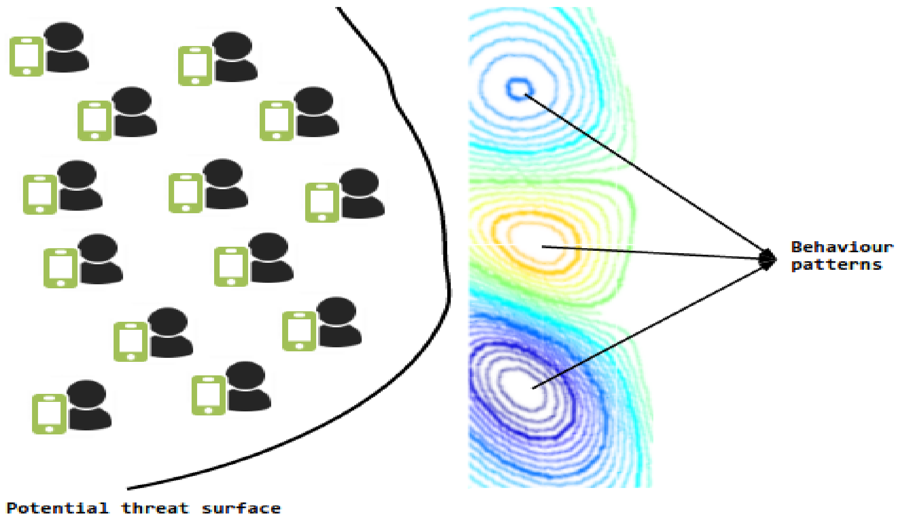


Figure 1.1: Illustration of analyzing the threat surface for similar user behaviours.

in particular, that of a smartphone. The goal is to group patterns of similar behaviour. It must be noticed how this perspective can be applied to grouping similar malwares as well as similar users. This steers the research to finding more dynamic solutions that ensure two key aspects of cyberspace risk management: *(i)* categorization of similar behaviours across threat surface (see Figure 1.1); *(ii)* continued even in post-authentication stage.

The continuous and ubiquitous security mechanisms post-authentication are of utmost importance in the current cyberspace situation. To this end, one might ask questions such as the following to identify and quantify different user behaviours:

1. What are the obvious patterns of behaviour that emerge in one user's interaction with his/her smartphone?
2. How similar are the discovered patterns of any two users (say $user_A$ and $user_B$)?
3. Given the quantification in step (2), can $user_A$ and $user_B$ be labelled to belong to the same 'group'?
4. After repeating steps (1-3) for everyone in the user community, how many 'groups' of users do we obtain overall?

In short, there is still a need to model and analyze different user (and/or application) behaviours for cyber-security purposes. Within this context, this thesis focuses on mining user behaviours by studying the extent to which phone behaviours can be associated with a single user. Formally, it is required to build a model for each user summarizing patterns in his/her phone usage. A particular class of pattern recognition that is meant to discover previously unknown patterns in the data, is called unsupervised learning. For the purposes of this research, it is necessary to conduct unsupervised learning exclusively for each user. This constraint is termed as *one-class learning* in the Machine Learning (ML) fraternity.

In summary, this thesis targets user behaviour mining, under the context of one-class unsupervised learning. The aim is that, learning frameworks of this nature, can begin to address the consistent challenge of factoring user autonomy in defense mechanisms. To this end, the use of a multilayer perceptron configured to act as an Autoencdoer (AE), prior to the application of a Self Organizing map (SOM), an unsupervised learning approach is explored. Such a configuration operates under a one-class learning constraint, with the objective of providing a unique characterization of user behaviour. The evaluations on publicly available datasets show that it is possible to identify patterns to characterize each user's behaviours using the proposed one-class learning approach.

The upcoming chapters in the thesis are organized as follows: research previously conducted in the field of behaviour analysis, using machine learning algorithms are summarized in Chapter 2. The adaptation of known clustering methods to smartphone datasets, thereby quantifying a similarity metric among users is presented in Chapter 3. The main contribution of this thesis *i.e.,* the utility of Autoencoders to obtain a discriminating representation of each user is detailed in Chapter 4, along with the visualizations from resulting SOMs. Chapter 5 draws conclusions of the research by providing inference from the results of the proposed framework; and discusses future research directions.

Chapter 2

Background

The discussion in this chapter begins with previous works in the cyber-security for mobile phones using machine learning techniques. This provides validation that ML methods are adopted in the security domain. Furthermore, the chapter also summarizes the research conducted to model users using smartphone datasets. Applications of SOMs related to behaviour modelling are then detailed. The chapter concludes by summarizing all the challenges faced thus far in the field of behaviour analysis using smartphone sensor information.

2.1 Machine learning in cyber-security

Clarke *et.al.,* investigate the feasibility of using keystroke information to improve authentication of mobile phone users using feed forward Multilayer Perceptron (MLP) [7]. A Nokia phone was modified to retain only the keypad interface and the data was collected in-house. The two cases of this study was: *(i)* text based input and *(ii)* number based input. At the time of enrolment, each participant was required to provide 30 repeated inputs from 32 users, two-thirds of which was used to create a template of the user's typing characteristic. This signature was quantified in terms of the keystroke latency and hold-time characteristic respectively. A feed forward MLP is employed to classify user input as 'authentic'. The MLP was given authentic ans imposter data as input to enable classification *i.e,* this method is not one-class. The authors acknowledge that *(i)* the performance of the MLP varies for each user; *(ii)* a reduction in the performance of the framework was noticed when dynamic authentication was attempted; and *(iii)* data collection was laborious, in that, filtering outliers (typing mistakes), in each of the 30 inputs per user was time consuming, yet important for research.

Alternatively, to aid keystroke analysis, smartphone sensor readings were used from the start to finish of the touch-based keystroke [4]. The authors posit that while

it might be easy to replicate authentic content *i.e.,* password PIN, it remains unlikely and near impossible to mimic the posture of the authentic user. To this end, in built sensors from Android smartphone were used to record the acceleration, orientation, magnetometer and gyroscope readings. They also use template-matching style, supervised, binary Bayes classifiers to classify an PIN input as authentic *bio-metrically.* TouchStroke was the android application developed to collect training data from 12 participants.
The approach to multi-sensor based authentication was seconded recently by Lee et.al., [25]. This study incorporates the accelerometer, magnetometer and orientation sensors to provide continuous verification of the current user *after* he/she has been successfully authenticated using the traditional password mechanism. To this end, a Support Vector Machine (SVM) is employed to train the user's profile using one day's sensor data. Each day the SVM model is re-trained to update genuine variation in the user's typing behaviour. For classification, one user's data samples are labelled positive, all other user's data samples are marked negative and both classes are sampled uniformly. The authors evaluate their system on their in-house PU dataset and the publicly available GCU version 2 dataset. Both datasets contain 4 users each using android smartphones. Many other researchers have used Android smartphone sensors to classify users as authentic or imposters [38, 26, 5, 50, 55, 18].

These studies vary in the considered set of sensors, their sampling frequency, choice of classification algorithm and the datasets being evaluated on. The commonality across all the aforementioned studies is that, none of them investigate one-class learning. That is, all these classifiers require the negative and the positive class during training (model generation). Furthermore, they are all supervised learning methods and it is well established that finding labelled data is very challenging (and/or is expensive) in real life.

Three notable research works stand out in utilizing human-gait as a behaviour-based metric for authorizing users, of which one study employs a one-class learning framework. Derawi *et.al,* investigated the utility of low-grade accelerometer readings to authenticate smartphone users [9]. This study was performed by placing an Android mobile device on the hip of a 51 participants, that measured accelorometer readings across the x-axis as the volunteers walked 4 times up and down a hallway.

One of these 4 measurements was used to create a reference template, while the remaining were used to obtain features for comparing against the user's reference walk. Since walking duration varied, the resulting input instances were of varying length. In order to compute similarity between these input patterns and the pre-computed reference walk, Dynamic Time Warping was the chosen distance metric. The authors confirm a 50% improvement in error rate in comparison to previous works. They also acknowledge that this method is sensitive to: *(i)* placement of the mobile phone; and *(ii)* choice of sensors chosen to profile human-gait *wrt* the sensor placement.

Later, Wei *et.al.,* demonstrate that derived features such as average, standard deviation, variance, skew, kurtosis and correlation are preferred to raw accelorometer readings, to summarize human-gait [54]. Furthermore, they employ graph-based clustering mechanism *i.e.,* unsupervised learning to identify walking patterns amongst users. Their solution was evaluated on a study group of 4 users, each resulting in 3 repeated samples for walking, race-walking and running. These authors also resort to a custom built smartphone application, and device placed on the participants' S1 vetebra, to obtain accelerometer readings. They acknowledge that their study is sensitive to sampling rates and the window size used to engineer features, but conclude that higher the sampling rates yield better recognition.

Casale *et.al.,* investigate the use of a one-class learner to enhance user authentication *after* training a multi-class AdaBoost classifier to discern user activities [6]. User activities such as walking, running, sitting *etc* are fed as input with ground-truth to the ensemble AdaBoost classifier. Only those patterns that are validated to belong to the 'walking' class are given to the one-class classifier for user-authentication. It is to be noticed that the one-class learner builds over the pre-trained supervised activity classification system. All the walking instances from a user is projected onto a subspace. However, the method of projection/ dimensionality reduction is not discussed in the study. A convex hull is built around all the inclass user's input as shown in Figure 2.1. During the user verification process, the binary one-class classification (belongs to authentic user or not) is reduced to estimating if the unseen input sample lies within the inclass convex hull. Removal of outliers and approximating convex hull to non-convex projections are done until linearly separable convex hulls are obtained.

Going forward, an input sample is said to not belong to the inclass user, if it does

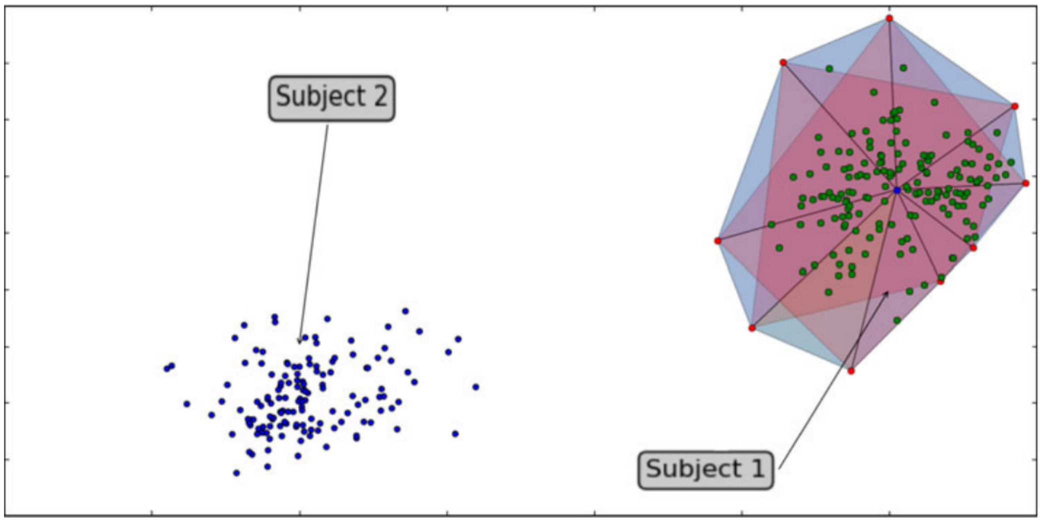


Figure 2.1: Bi-dimensional convex hull from 2D projections of inclass user samples [6].

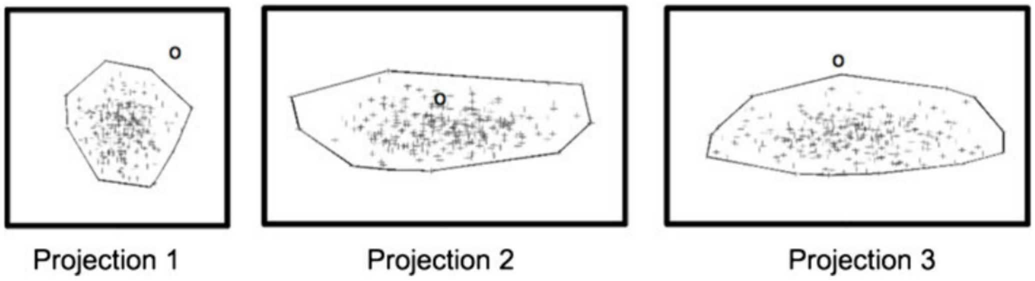


Figure 2.2: Projections of the inclass convex model onto 2D space: in this case, the input sample (denoted by circle) is said to belong to the inclass user, because it lies within the 2^{nd} projection of the convex hull [6].

not lie within either of the projections of the inclass' convex hull (see Figure 2.2). The authors acknowledge that obtaining characteristic reference walk patterns (in the wild) is challenging, owing to the variations in a subject's walking styles when faced with an obstacle, walking on rough terrain, in crowded places *etc.*.

In summary, literature suggests the following main take-aways for future research when adopting smartphone based sensors that capture the physical attributes of the phone, thus its legitimate owner:

1. Data collection and preprocessing is laborious.
2. Quality of data is highly sensitive to placement of the mobile phone on the human body.
3. Labelled data for human activity modelling rarely exists.

4. Developing templates of sensor profiles (as a surrogate ground-truth) is non-trivial.
5. Choice of sensors to best model a user's physical interaction with the phone is non-trivial.
6. Difficult to maintain the performance of the ML based user recognition method when transitioned from a static solution to a dynamic framework.

2.2 User modelling related to smartphone usage

Eagle and Pentland, from MIT Media Labs collected (and made publicly available) one of the earliest mobile phone datasets capturing Cell tower, bluetooth, application usage, calling and SMS logs from a 100 participants (students and faculty included) [12]. The posit that regardless of the variation in human behaviour, one could arrive at predictive models of user *routine* patterns. To this end, they begin by separating users with high entropy *i.e.,* less routine from those who exhibit low entropy. An example of users with low entropy are professors who maintain a consistent work-ethic; while freshman year students are typical examples of high entropy users (as they are likely to change their locations based on class hours). Entropy is measured using the probability density function (PDF). The slightest variation in amount of time spent connected to a cell tower can hugely impact the cell tower PDFs for two very similarly positioned users. Thereby the PDFs ensure discriminative representations for each user within the same demographics. In addition, the PDFs (of cell tower and bluetooth IDs) serve as inputs to Hidden Markov Model, conditioned on hour of the day and if the day was weekend or not. Subsequently clustering was performed to distinguish users from one another with accuracies >95%. The authors acknowledge that: *(i)* using non-linear techniques will provide higher accuracy; and *(ii)* learning a user's application usage routines will yield better models of the user's spatio-temporal 'behaviour'.

Noticing the difficulty in incorporating temporal patterns across different timescales, Eagle and Pentland proposed a new solution to behaviour modelling using mobile data [13]. This time, principal components analysis was used as a dimensionality reduction technique. These principal components represent the most recurring behaviour each day *i.e.,* routine. A weighted combination of 6 most primary eigenvectors of user's data, resulted in 90% detection accuracy. Thus the name *eigenbehaviours*. The authors also show that, by reducing the dimensionality of the user data, inferences of how similar two users are can be drawn using a simple Euclidean distance metric. This results in modelling a 'behaviour space' of users with common routine behaviours. A user's affiliation to a behaviour space of this nature will also allow researchers to *predict* the user's future behaviour patterns. This study is the closest in relation to the current research topic of grouping users with similar behaviours for cyber security purposes.

A study that best utilized smartphone application data to model users was conducted by Do and Gatica-Perez [10]. They propose a probabilistic framework that identifies emergent patterns in an individual's daily application usage. Using these patters, they retrieve a list of all users that are likely to have similar daily application usage. Each user's application logs are initially represented in bag-of-apps format, based on the frequency of using an application in the morning, noon, evening and night. A set of 5 applications are considered *viz.,* voice, SMS, internet, camera and gallery. One row of the bag-of-apps consists of a 20D vector indicative of usage frequencies. The resulting matrix is fed as input to a topic modelling algorithm. The purpose of topic modelling, is to find the most frequently occurring associations between the application and time of use (one topic). Daily usage patterns is thus an array of user-topic tuples with respect to time. A repository of such daily application usage patterns is built from a control group of 111 users (private dataset collected with consent by distributing smartphones to 111 participants). These users are ranked according to the posterior probability of containing a query usage pattern, *i.e.,* inclass user pattern. Using Bayes theorem with a uniform prior probability, the framework is now able to identify users with similar application usage routines. The current research borrows the initial representation of applications in the bag-of-words format. However, the proposed framework refrains from adopting topic modelling approach since, this method does not fit the one-class learning constraint.

LiKimWa *et.al.,* study the correlation between a user's mood and his/her interaction with the device [28, 29]. In particular they factor in application usage, phone calls, SMS,emails, web browsing history and location to engineer discriminating features for each user. Only the ten most frequently occurring values are used to construct the feature vector in each category. The location feature is derived from the result of the unsupervised DBSCAN algorithm answering, the top ten most frequently visited 'location cluster'. Within this context, the authors conclude that under the supervised framework, building a 'one-size-fits-all' mood model yields a poor accuracy of 61%; whereas building personalized mood models for each user improves the prediction accuracy to 91%. However, they remark that obtaining ground-truth and training data for each user is expensive. This study is augmented with a richer feature vector consisting of: duration spent actively using each application (earlier only count of launches), category of the application used, and the two previous mood averages recorded by this user. Going forward, mood averages of each user per day is computed that serves as labels to a linear regression model that infers the current mood of the user using the input feature set. Sequential feature selection was used to select the most distinguishing feature descriptor for each user. The authors conclude that: (i) each user used a different dimensionality of features; (ii) mood modelling works better for some users than others; (iii) personalized models report high accuracy but demand extended training duration (2 months of training resulting in 85-93% accuracy); (iv) selection of sensors is imperative yet empirical.

Stöber *et.al.,* associate a user with the nature of traffic created as he interacts with applications installed on his phone [49]. This is achieved using traffic features engineered from the `tcpdump` logs from Android smartphones. Furthermore, the nature of the traffic created by the applications (due to background processes such as sync) is said to vary with respect to the combination of active applications running on the user's handset. A kNN classifier and SVM are employed to conduct a supervised binary classification (user or not-user) task. The training set consists of equal proportions of the user's fingerprints and those from other users. The authors show that SVM outperforms kNN in terms of accuracy in classifying unseen data by nearly 4% drop in error rates. In summary, the authors indicate a new avenue *i.e.,* network applications traffic to improve user identification under a multi-class supervised learning framework. This study was performed with a control group of 20 participants.

Following are the observations from literature review in the field of modelling smartphone users, as a characteristic of their device interactions:

1. Majority of the studies are conducted on private datasets that are not publicly

available.

2. The number of users in each dataset is varying *viz.,* 4, 20 and 111.
3. Application usage, cell tower information and website browsing behaviour has gained lesser research attention in comparison to sensor based user modelling methods.
4. All learning algorithms investigated thus far are supervised in nature, although they build personalized user models.
5. No study is akin to clustering groups of similar users and/or applications.

From the research opportunities outlined in this section, my study investigates user behaviour modelling using both contextual information *i.e.,* application usage, cell tower information and website browsing history as well as physiological information namely, accelerometer, magnetometer, orientation, and sensors alike. The central idea is to attempt behaviour modelling as a one-class unsupervised learning problem devoid of labels and non-user information during the training phase of the model. To this end, the utility of a vector quantisation based neural network, namely SOM, solution is to be investigated. The next section details established uses of SOM in behaviour modelling.

2.3 Related applications of self organizing maps

This section discusses the use of SOMs to discover underlying patterns in the data. While the scope of SOMs used in this type of research is huge, the focus of this section remains within the realm of application or user behaviour clustering.

Barrera *et al.* employed SOMs to analyze permission-based security models for Android platforms [3]. In this study, 1100 applications most frequently downloaded in the Android market were analyzed. The research intention was to understand the patterns of usage of the Android permission model, by application developers. To this end, a 2D hexagonal SOM was employed in order to discover clusters of applications that requested the same set of permissions. To the SOM, each application is represented as a bit vector, where each bit location corresponds to whether that permission was requested or not. The authors acknowledge that SOM is a suitable approach to analyze bit vector based representation of data. The study concludes with inferences such as: “Given that SOM places similar input patterns in the same region, this indicates that applications from the same category do not necessarily behave similarly (i.e., do not request similar permissions). This reflects the fact that the categories defined by Google are based on semantic activity classes rather than the technical features used to implement them"[3]. Furthermore, the ability to visualize the permission usage pattern in a 2D plane allowed researchers to verify the frequency of usage of each permission across the application space. This study is indicative of: *(i)* ability of SOM to group bit vectors by similarities on a 2D space; and *(ii)* importance of visual inspection of the trained SOM model.

Literature suggests the use of SOM to categorize different high-level behaviours, Stevanovic *et al.*, use a simple SOM to detect malicious web-crawlers [48]. Specifically, the SOM successfully differentiates malicious and non-malicious internet users. In that, the trained SOM is also able to delineate crawling behaviours of automated web crawlers: well-behaved or malicious. A 10D feature vector is engineered from web-server logs, fed as input to a 2D dimension of 10x10 size, trained for 200 epochs. The authors verify SOM is sensitive to the density distribution of training data. That is to say, a relatively larger number of neurons are likely to respond to the majority class in the data.

Drachen *et al.* used an emergent SOM to model player behaviours in a game [11]. A toroid shaped map of 50x100 dimension was employed to discover patterns in player behaviour. The toroid map was preferred over the 2D sheet, in order to overcome border effects. Post training (100 epochs), 4 non-overlapping clusters emerged. To verify the cluster separation, density-based SOM visualization was used. The clusters separated: skilled players who die very few times; players who die very frequently from falling but, never ask for hints during a puzzle; players often killed by opponents and take long time to finish the game; and players who die often but finish the game quickly. These insights from SOM are said to be useful when analyzing whether the game is being played as intended or if any surprising playing behaviours emerge. It is noteworthy that an unsupervised learning solution results in well characterized clusters, ensuring a reasonable trade of between computational effort (map size) and training time (epochs required for convergence). This study is also an evidence of how the topology preserving attribute of SOMs results in coherent clusters.

Verdu *et al.* study the application of SOM in filtering anomalous electrical demand behaviours . [52]. Firstly, the SOM results in discerning typical demands from reduced demands (exam season) and demands during holidays (less people occupying university, where electrical consumption was studied). For all typical demands, SOM based clustering is re-employed to classify the customers to different categories (university, residential, commercial *etc,*) based on their load profiles. K-means clustering of the resulting SOM optimized for the least Davies-Bouldin (DB) index results in the final categorization of customers. The trained SOM was tested with two new unseen customer load profiles. It was found that the quality of the resulting SOM map was better when frequency domain data was used as input (in comparison to time domain).

In summary, SOMs have been used to model behaviours in various fields, using binary vector based features, small map sizes and reduced training time. The clusters thus obtained possess unique characteristics due to the topology preserving nature of SOMs. The 2D 'projections' of data allows visual inspection of clusters formed using high-dimensional input data. SOMs can be used both as a standalone clustering technique or as an entity of a clustering ensemble. Several different data mining approaches are used for understanding user behaviour in wired and wireless networks as well as smart phones. However, to the best of my knowledge none of these works explored the usage of an autoencoder based SOM to encode and cluster data to characterize mobile phone user behaviours.

Chapter 3

Behaviour Modelling: under one-class learning

The trivial solution to group similar user behaviours without *a-priori* information of the context of each user's smartphone usage, is to employ an unsupervised clustering paradigm. Given the task to identify patterns of behaviour in *each* user, it demands us to build a cluster model for each user independently. That is, the system is expected to learn the user's behaviour by continuously being exposed to one user *only*, the positive class. Formally, this machine learning approach is referred to as a one-class learning setup. We refer to the positive class *i.e.* the user being modelled, as the inclass user. All non-positive classes *i.e.* other users in the dataset, are referred to as outclass users. The user specific model is learned from data pertaining exclusively to the inclass user, no outclass data is made available at the training stage (one-class learning).

The focus of this thesis is to investigate:

- the use of partitioning and model based clustering methods that borrow principles of competitive learning as well known examples of unsupervised learning algorithms;
- the feasibility of pattern mining under a one-class learning constraint.

Through competitive learning, input data is summarized by a group of prototype vectors. A finite number of randomly initialized prototypes, compete to respond to an input vector. The prototype that responds most strongly is referred to as the 'winner'. Through training, each prototype specializes to respond to a 'type' of input data. This leads to formation of clusters. The algorithms that use competitive learning principles are self-organizing maps (prototypes are neuronal weights) and vector quantization (prototypes are cluster centroids). The aim of vector (samples) quantization (grouping/binning) is to discover/build a representative vector for each input datum such that distortion is minimized. Distortion is defined as sum of all absolute differences between each sample and its associated centroid (representative vector). The aim of self-organizing maps is to train a set of similar neurons, such that each neuron organizes its synaptic weights towards representing a particular input pattern.

In this chapter, clustering solutions from K-means, a well known quantization algorithm; and Self Organizing Maps (SOM) are discussed. The theory, adaptation and results from both algorithms are detailed. A statistical significance test is conducted to choose the better learning paradigm for further improvisation. These results act as a benchmark to develop the contribution of this research.

3.1 Smartphone Usage Datasets

These datasets are derived from re-programmable applications that log sensor usage in the device that houses the application. All datasets used in this theis are made publicly available for research purposes. It is ensured that the traces are anonymized (dataset contains no identifying information about subjects participating in the data collection process) and hashed (website URLs for example). However, the datasets vary in the total logged duration, sensor specific sampling frequency and the set of sensors monitored.

The applicability of this research is, to model user behaviours despite their similarities (a phenomenon that also manifests in the real world). For example, students attending the same university would by and large connect to the same cell tower during work hours, visit the university website every so often, and given that each dataset was built on a particular firmware, there will be a set of common pre-installed applications. Nonetheless, phone interactions vary in terms of third party applications used, periodic visits to a favourite news channel, routine check-ins to a cafe at a specific time of the day and thereby connecting to a different cell tower and/or WiFi access point *etc.*

For the purposes of this research, I classify the set of sensors into two broad categories *viz.* discrete and continuous. Discrete sensors are those that provide contextual information about the user's location, most browsed websites, preferred applications *etc.* This is usually represented in binary: 0 representing 'not-connected/launched' and 1 representing 'connected/launched successfully', hence the name discrete. Continuous sensors are those that provide information about the way the user physically handles the devices or the user's physiology such as, height, gait, walking profile, speed with which a phone is grabbed to answer a call *etc.* These sensors log real valued sensor information across the physical axes, hence the name continuous. To substantiate, foreground applications, web histories, associated WiFi and cell tower data are categorized as discrete. Sensors that capture real valued information such as acceleration, rotation, system state are categorized as continuous. The subsequent goal of this research is, to establish which of the two sets of sensors are most suitable for modelling a user's interaction with his/ her device. The sensor subset that yields maximum dissimilarity for each user is deemed to best suit behaviour modelling. Intuitively, this translates to asking questions such as the following: *Does internet surfing pattern better summarize a user-device interaction, in comparison to accelerometer readings?*

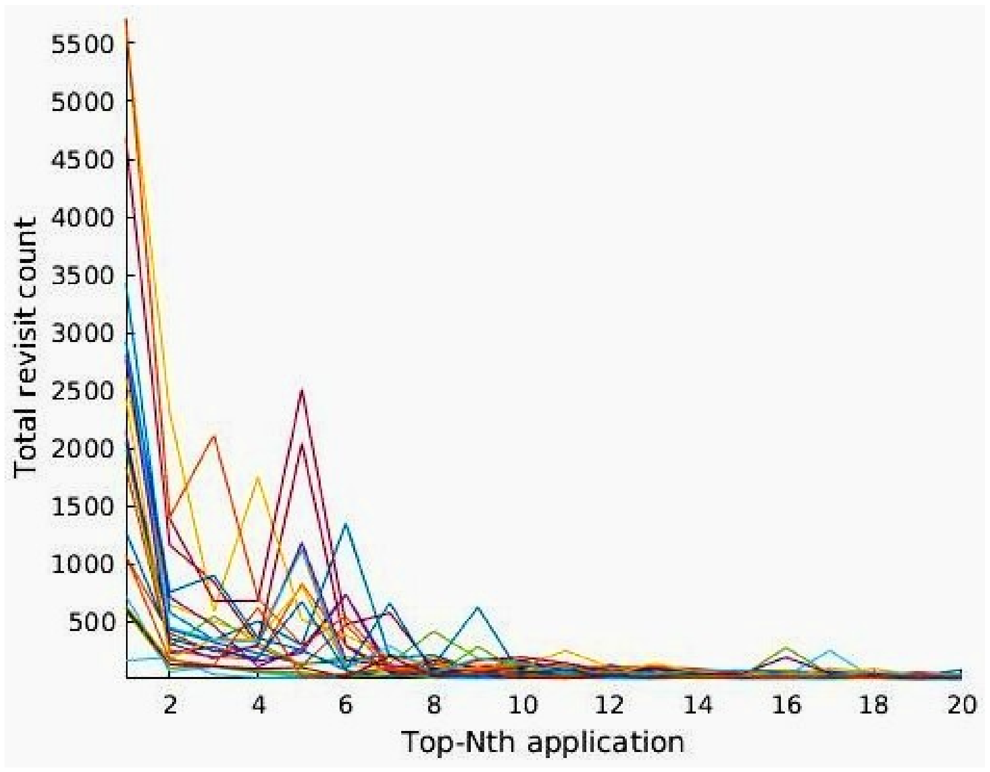
3.1.1 Rice Live Lab Traces

In this thesis, I envisaged building a system that is exposed to real world situations such as:

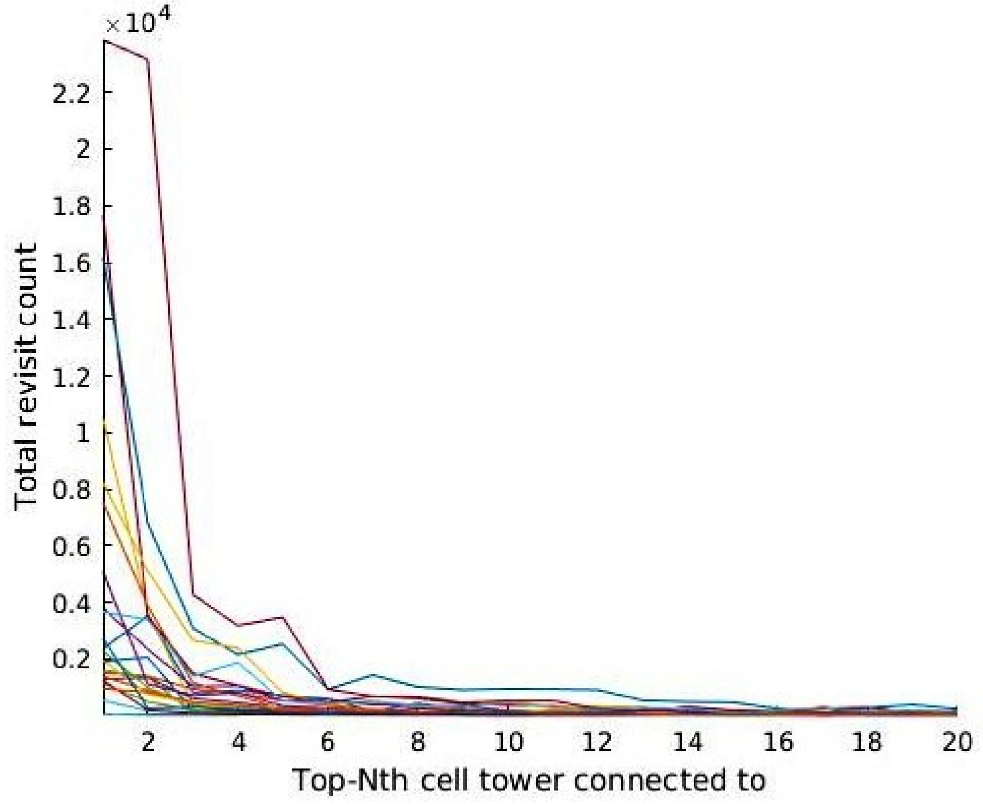
- encrypted web history,
- user clearing application logs,
- similar correspondence with the mobile device across different geographical locations.

I thus choose event driven logs of running applications, periodic logs of connected cell tower and hashed web browsing history, from the LiveLab dataset [45]. These traces contain 23 users. Figure 3.1 and Figure 3.2 shows the usage of, the 20 most frequently used applications, cell towers and websites. It is worth noticing that there are a few applications and cell towers that belong to the $Top20$ set of all participants in the dataset. For the continuous sensor subset, a $12D$ feature vector was built using the following sensors:

- Acceleration: measured acceleration logged along the *x,y,* and z axes, once every 15 minutes when the phone is active.
- Charge duration: the amount of time spent in charging (1) state and not-charging (0) state, logged periodically *i.e.,* interrupt driven.
- Display duration: the amount of time spent in display-on (1) state and display-off (0) state, logged when state changes *i.e.,* event driven.
- Power: percentage of battery level, battery voltage in milivolts, and amount current flowing into the battery in miliampers, event driven logging.
- Sleep duration: the amount of time spent in low power mode asleep (1) and awake (0) despite the display status.



(a)



(b)

Figure 3.1: Dataset statistics (per-user) for LiveLab traces. (a) Applications used; (b) Cell towers connected

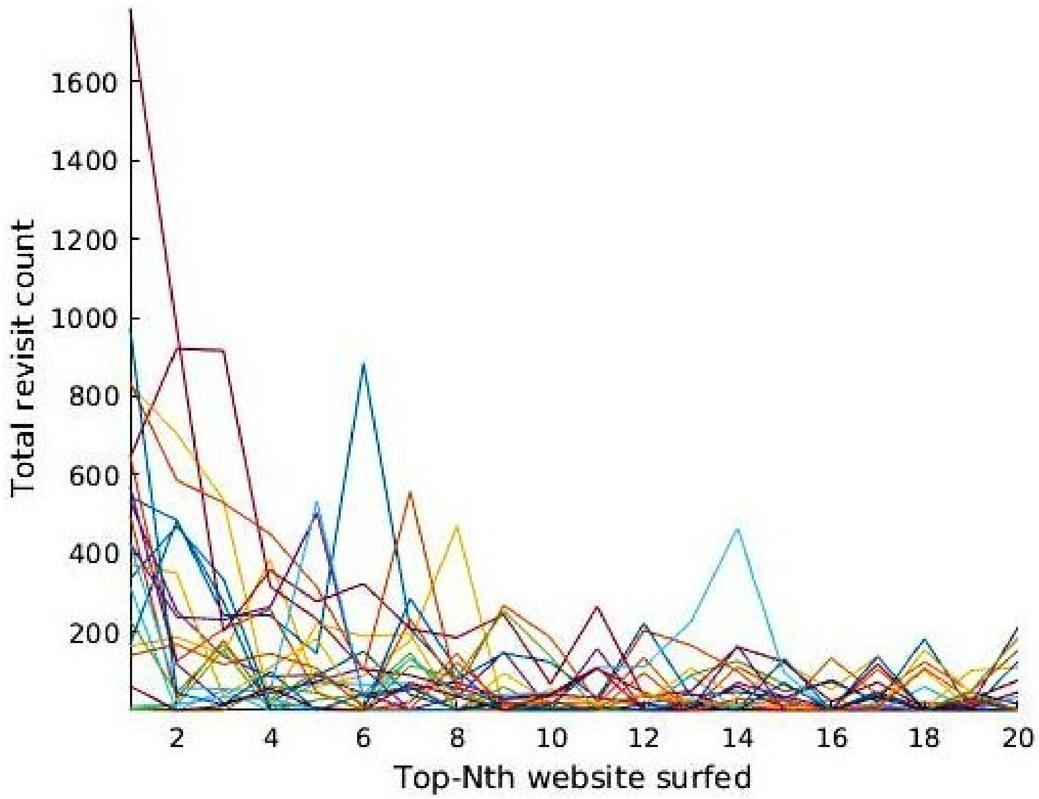


Figure 3.2: Dataset statistics (per-user) for LiveLab traces: Websites visited.

User ID	No. of days	User ID	No. of days
A00	375	A12	343
A01	365	B00	283
A02	378	B02	366
A03	368	B03	372
A04	368	B04	399
A05	383	B05	365
A06	376	B06	366
A07	413	B07	367
A08	434	B08	366
A09	280	B09	390
A10	366	B10	304
A11	369	B11	363

Table 3.1: Duration (in days) of logged sensor data for all users in Rice LiveLab dataset.

3.1.2 GCU version 1

The GCU version 1 dataset is collected from 7 different users, on Android devices in the year 2013 at Glasgow Caledonian University (GCU) [2]. It contains data pertaining to the discrete sensor category:

- Applications: list of all active applications at a particular timestamp. The raw data consists of one entry for each functionality of the application used. The data has been preprocessed to account for the same application. For instance, *com.google.android.location.geocode.GeocodeService* and *com.google.android.location.fused.FusedLocationService* are both represented by *com.google.android.location.*
- Cell tower: the connected cell tower ID
- WiFi network: list of all WiFi signals detected at a particular timestamp.

The duration of the data varies from 2 weeks to 14 weeks per user (see Table 3.2). To ensure consistency, earliest available data for a period of 14 days is used for each user.

User ID	No. of days
U1	81
U2	69
U3	14
U4	16
U5	22
U6	15
U7	17

Table 3.2: Duration (in days) of logged sensor data for all users in GCU version 1.

3.1.3 GCU version 2

The GCU version 2 dataset consists of smartphone data collected from android devices in the year 2014 [2]. It contains data pertaining to application usage, cell towers connected to, WiFi networks used, acceleration, rotation, noise and magnetic field statistics, logged over a period of 3 weeks. This second version of the dataset includes 4 users. However, the authors of the dataset do not comment on which of these 4 users (if any) are the same with the 7 users of the GCU version 1 dataset.

The nature of the sensor information under the discrete categorization is akin to that explained in subsection 3.1.2. The dataset consists of the following sensors under the continuous categorization, resulting in a $16D$ feature vector.

- Acceleration: measure of the acceleration force in ms^{-2} that is applied to a device on all three physical axes (x, y, z)
- Magnetic field: measure of the ambient geomagnetic field for all three physical axes (x, y, z) in $\mu\tau$.
- Noise: the minimum, mean and maximum noise levels (in decibels).
- Rotation: measures of the degrees of rotation that a device makes around all three physical axes (x, y, z).
- System: CPU usage by the user, CPU usage of the system, and number of active processes.
- Light: binary value to represent the ambient light level: on (1) or off (0).

User ID	No. of days
U1	21
U2	25
U3	30
U4	16

Table 3.3: Duration (in days) of logged sensor data for all users in GCU version 2.

3.2 Representation to learning algorithms

This section discusses the manner in which all three datasets *viz.,* Rice LiveLab traces, GCU version 1 and GCU version 2 are represented to the learning algorithms. The resulting user matrices from the discrete and the continuous set, are linearly normalized to the $[-1, +1]$ interval.

3.2.1 Discrete sensor information

With this feature subset, user behaviours are characterized by the most frequent applications, the most frequent web sites and the cell tower information for each user over a given time interval. To this end, a fixed length representation is employed for each observed time interval. I consider only the N most frequently used and/or visited applications, websites and cell towers to represent each user. In a way, this is similar to a bag-of-words representation for the three categories of features as follows:
$sample_i = [app_1, app_2, ..., app_N, web_1, web_2...web_N, tower_1, tower_2...tower_N]$
where, app_1, web_1 and $tower_1$ indicate the (first) most frequently used application, website and cell tower (respectively) by $user_i$.

3.2.2 Continuous sensor information

In the continuous domain, user behaviours are characterized by sensor values across the *x, y, z* axes and their active/inactive state. The feature vector thus comprises of 3D vectors for accelerometer, gyroscope, rotation and magnetometer readings; and 2D vectors that log the duration the sensor spent in active mode and the duration spent in inactive or sleep mode.

In summary, the discrete sensors are those that contain binary information (0 or

1) about the contextual usage of the device. On the contrary, the continuous sensors are those that log real valued data pertaining to the user-device interaction.

3.3 K-means

The most well known partitioning method is the centroid based K-means algorithm. In order to group $\mathcal{N}$ exemplars, K-means requires $\mathcal{K}$ initial seeds. These seeds represent the centroid of the k^{th} cluster. The samples are grouped such that the distance (euclidean) of each sample within the cluster is less than intra-cluster distances.

3.3.1 Theory

The algorithm begins by calculating the pairwise distance from each exemplar to all the cluster seeds. The exemplar is assigned to the closest cluster (hard clustering). The position of the cluster centroid is updated by a factor of the mean of all the exemplars that belong to that cluster. These updates continue until the distance function of interest has been optimized. Formally, consider a training set of $\mathcal{N}$ samples: $x_1 ... x_i .. x_n$. Assuming $c_1 ... c_k$ initial centroids such that, $x_i, ci \in \mathrm{I\!R}^{\mathrm{n}}$, the label assignment and centroid updates are as follows:

$$l_i = \arg\min_j ||x_i - c_j||^2 \tag{3.1}$$

$$c_j = \frac{\sum_{i=1}^{m} (l_i = j).x_i}{\sum_{i=1}^{m} l_i = j} \tag{3.2}$$

where l_i is the cluster label assigned to the i^{th} exemplar. Consequently, the efficacy of such a clustering solution depends heavily on the choice of the $\mathcal{K}$ initial seeds and the distance function. The quality of clustering is measured based on the cohesiveness and separation among the clusters. It is possible to estimate the optimum number of clusters *i.e.,* $\mathcal{K}$ by empirically testing the cluster qualities on a range of desired number of clusters. To this end, a high silhouette score of +1 indicates well separated clusters containing very similar samples from the training set. Conversely, a silhouette score of -1 is indicative of cluster overlaps [44]. Finally, in order to address the initial seed value of the highest silhouette $\mathcal{K}$, the algorithm is run multiple times with varying random seeds and the average of cluster centroids is considered.

3.3.2 Practice

The adaptation of K-means to a one-class learning scenario is as follows:

1. Obtain optimum number of clusters for inclass data by silhouette scoring (say $\mathcal{K}$).

2. Re-initialize $\mathcal{K}$ random cluster seeds through multiple runs of K-means (say 20).

3. Calculate the farthest distance from each of the resulting cluster centroids to samples that belong to that cluster (say $Cmax_i$).

4. Compute pairwise distances from all samples in outclass data to each of the cluster centroids and identify the closest centroid.

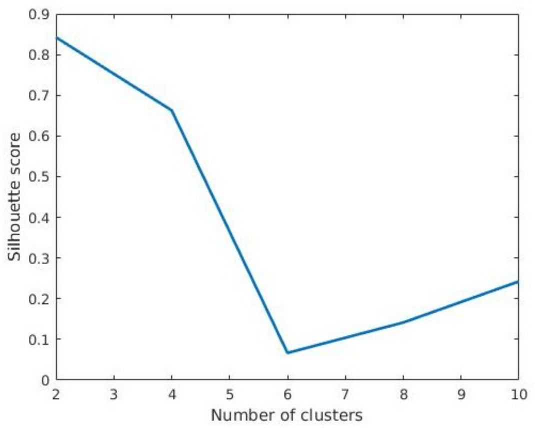
Measure of exclusivity

Following the nomenclature in subsection 3.3.2, the ratio of samples in outclass data that are within the distance $Cmax_i$ for each cluster $i \in range(\mathcal{K})$. The resulting ratio is an approximate measure of dissimilarity.

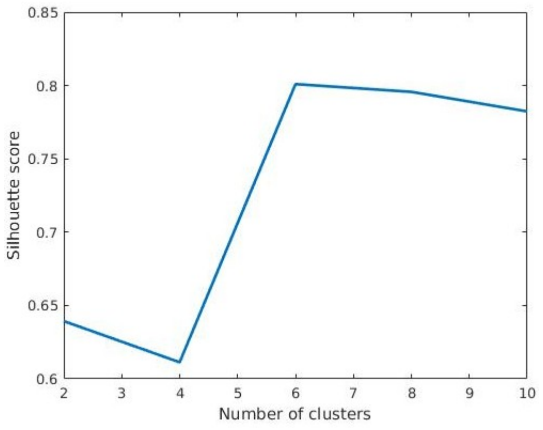
Ideally two very similar behaviours would result in all samples belonging to either of the inclass clusters *i.e.,* dissimilarity ≈ 0 and 1 otherwise. The average dissimilarity of one user with respect to all other users in the dataset is termed as Exclusivity Rate (ER). The average of per-user ER across 20 runs is called the average ER.

3.3.3 Results & Discussion

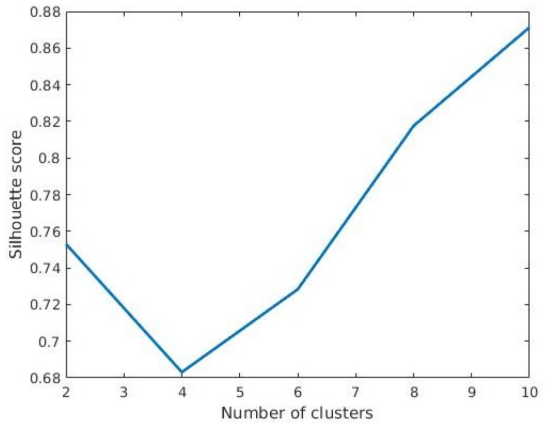
The silhouette scores for a sample inclass user from each dataset is shown in Figure 3.3. The number of clusters tested were in the range [2-10]. The cluster count with the maximum silhouette indicative of optimum separation was chosen as the inclass cluster 'model'. No user in the GCU set (version 1 and 2) ever attained a negative silhouette score. 8 users from the Rice LiveLab dataset obtained scores in the range $\approx [-0.02 : +0.88]$ for the discrete subset. These 8 users obtained improved scores in the range $\approx [0.6 : 0.8]$ for the continuous set. However, the measure of dissimilarity is very poor for the discrete sensor information and zero for the continuous sensor information. Table 3.4 contains the results for all 3 datasets.



(a)



(b)



(c)

Figure 3.3: Varying silhouette score for each user's discrete sensor data: (a) Rice Live Lab sample: K = 4; (b) GCU V2 sample: K=6; GCU V1 sample: K=10.

Dataset	Subset	Average ER
Rice LiveLab	Discrete	2.25
	Continuous	$7e^{-4}$
GCU Version 2	Discrete	19.94
	Continuous	$2.63e^{-2}$
GCU Version 1	Discrete	29.51

Table 3.4: ERs from K-means on all datasets.

The poor performance in K-means can be attributed to the spherical nature of clusters, the possibility of obtaining local optimums and the algorithm's sensitivity to noise. It is not trivial to tune the parameters for 'big data' and that questions scalability. Furthermore, it is difficult to verify and/or interpret high dimensional clusters. Despite the plethora of dimensionality reduction techniques, inferences drawn from K-means clusters at lower dimensions cannot be extended to higher dimensions. This is because there is no intermediate procedure that preserves the topology of high dimensional data in partitioning based clustering techniques.

In order to compute clusters on large datasets, K-means requires batch updates which in turn necessitates training data to be available in advance. Given that the scope of behaviour mining projects lies in the realm of security systems, a near real time learning model is preferred. In the next section, the use of an online and topology preserving model-based clustering technique is investigated.

3.4 Self-Organizing Maps

Self organizing maps (SOM) belong to a model-based data analysis method in which, similar input samples are grouped to belong to similar models (*a.k.a,* map units) SOMs summarize the distribution of data following a framework for unsupervised learning. Moreover, the topology of the resulting neurons resembles that of the original data care of the interaction between: 'best matching' unit, updating neighbourhood and annealing schedule (for the neighbourhood); albeit as projected into a typically 1 or 2D topology. SOMs have been used to provide intuitive descriptions of data over a wide range of applications and dimensions. Formally, the SOM algorithm for an n-dimensional input $\mathcal{D}$, can be summarized as follows [34, 23]:

3.4.1 Theory

1. Initialize a 2D lattice of neurons (map unit) such that the position of each neuron is described by n-dimensional weight vector whereas each neighbour of the lattice is predefined. The lattice defines the concept of a map/map neighbours and the weight vector defines the location and therefore distance to an exemplar.
   $\mathcal{M} = \{ m_i \mid m_i \in \mathbb{R}^{\mathrm{n}} \}$

2. Begin coarse training of the SOM to establish a high-level ordering of the 'neural map'

   (a) For each training exemplar $d_j \in \{ \mathbf{d} \}$, find the Best Matching Unit (BMU) *w.r.t.* the Euclidean distance metric. Create a list of all BMU's and corresponding data instance in $\mathcal{M}$.

   $$\mathcal{BM}(m_i) = \{ d_j \in \{ \mathbf{d} \} \mid \arg\min_i(\mid |d_j - m_i||^2) \} \tag{3.3}$$

   (b) Update the weight vector in a batch mode within the neighbourhood function $\mathcal{N}_c$ which decreases linearly at each iteration. For neurons outside the neighbourhood, the weight vectors remain unchanged.

   $$\vec{w}^{t}_{m_i} = \vec{w}^{t-1}_{m_i} + \alpha * ||\overline{\mathcal{BM}(m_i)} - \vec{w}^{t-1}_{m_i}||^2 \tag{3.4}$$

   where $\overline{\mathcal{BM}(m_i)}$ is the average of all data samples that are associated with each BMU $m_i \in \mathcal{N}_c$. Thus, for each BMU all neurons within the lattice neighbourhood are updated. The size of the neighbourhood is subject to an annealing schedule (incrementally decreases as training epochs increase).

3. Repeat Step (item 2) for fine training the SOM lattice with neighbourhood function $\mathcal{N}_f$ given that $\mathcal{N}_f < \mathcal{N}_c$.

4. Repeat Step (item 3) with the recent scalar neighbourhood radius (R_s) until convergence, i.e. weight changes below some minimal threshold.

Post training, the resulting vector of all BMUs obtained for the entire training set is referred to as 'SOM hits' from which the most frequently 'hit' neurons can be identified.

3.4.2 Utility

The similarity in the objective function of K-means and SOM can be observed in Equation 3.1 and Equation 3.3. Both of the algorithms have the same notion of updating centroids (K-means) and/or map units (SOM) by a factor of the average of all samples closest to it. This implies, that SOM is similar to traditional vector quantization techniques. The benefit of employing SOM over K-means is to ensure that the similarity of the map units (lower dimension, 1D, 2D or 3D) approximate the similarities in the feature space (high dimension). This is referred to as SOM's ability to maintain the topographic order of map units (models) through the learning process. Consequently, it is also possible to visualize, interpret and verify the clusters thus formed. Essentially, SOM results in mapping groups of similar inputs (behaviours) to one model (pattern) on a lower dimension. Thereby establishing the utility of SOM in clustering, for the purpose of this research.

3.4.3 Practice

A 2D hexagonal lattice of neurons (map units) is created. Initialization of the SOM can be done randomly and linearly. In particular, linear initialization is done along the two greatest eigenvectors of the inclass data, thus Principal Component Initialization (PCI). With the batch training learning algorithm, these weights converge earlier in comparison with a randomly initialized map[23]. The usefulness of PCI has been tested empirically using topographic error (TE), a measure to asses the quality of resulting SOMs. It is the percentage of input data having non-adjacent 1^{st} and 2^{nd} BMUs. Figure 3.4 presents the TE for the two different initialization methods of the SOM. It is clear that the map topology is well preserved with PCI. The training cycle for the SOM occurs in two stages. In order to obtain a global order of map units in the SOM codebook (*w.r.t.* the inclass data), a monotonically decreasing neighborhood radius is used. Once complete, the most recent Gaussian function is employed for stage two *viz.,* fine training. The training cycle continues until all the map units in the SOM codebook cease to change. Finally, the absolute count of how often each neuron in the codebook was fired as BMU in 'response' to inclass data is termed as the inclass hit response. The parameters used in configuring the AE and SOM are listed in Table 4.1.

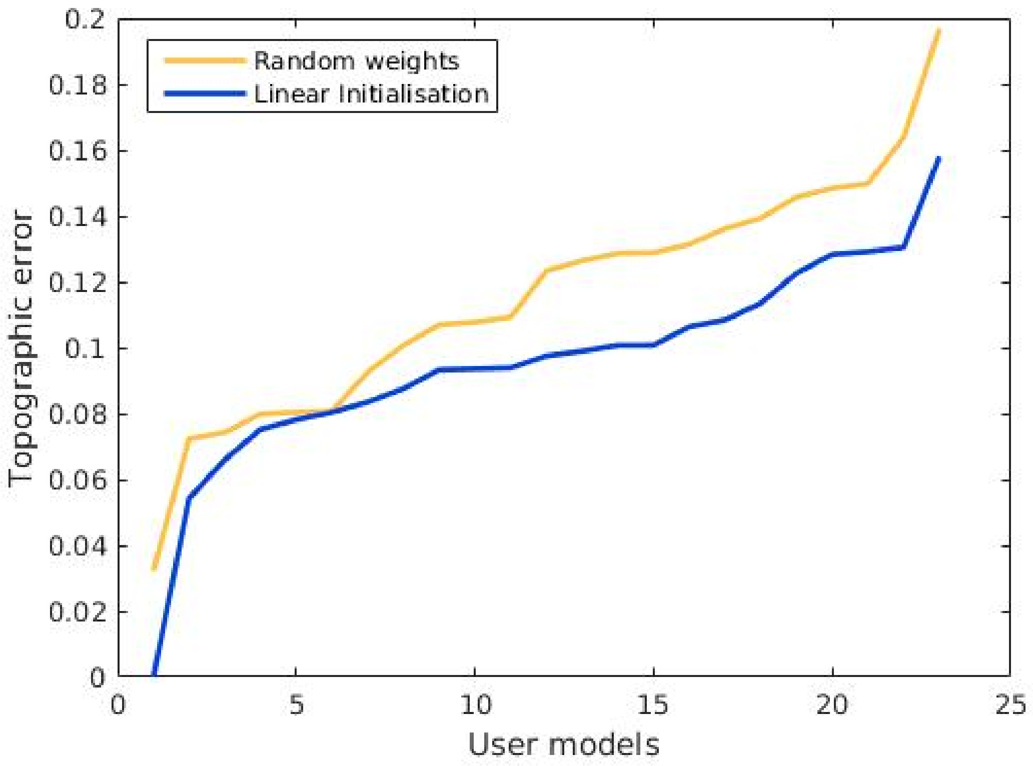


Figure 3.4: Reduced topographic error when neurons initialized along the space spanned by two greatest eigenvectors.

Measure of exclusivity

As stated earlier, when outclass data is given as input to a pre-trained inclass SOM, the result is a hit response array. Ideally, the inclass and outclass hit responses should have different distributions, for two unique users. To quantify the extent of dissimilarity from one user, we use the Normalized Mutual Information (NMI) metric. NMI by entropy for each labelling. This is done by calculating the expected value for the MI [47] [39].

Therefore, if the clustering responses of two users results in a NMI of 1, they represent the same behaviour patterns. Alternatively, if the NMI is close to 0, it is said that the two users have very distinctive behaviours. Given that, dissimilarity is defined as the inverse of NMI.

$$D_{ij} = \begin{cases} 0, & \text{if } i = j;\ \text{same user} \\ <1, & \text{if } i \simeq j;\ \text{users with some similarity} \\ 1, & \text{if } i \neq j;\ \text{distinct users} \end{cases} \tag{3.5}$$

One run of the SOM results in a n-dimensional matrix representing user dissimilarities. Its principal diagonals represent dissimilarity of a user with himself/ herself; and thus it is ideally zero. Given that all of us will have common websites we frequent (Google, Facebook *etc.*), or connect to the same WiFi (university access points), or use standard applications (alarm, FM radio, gallery *etc.*), it is unlikely to ever obtain an ideal dissimilarity of 1 for any two users.

It is also important to quantify the extent of exclusivity for each user's behaviour. For an inclass user, exclusivity is defined as the average dissimilarity from all other users in the dataset. The degree of uniqueness is termed as ER.

$$ER_i = \sum_{j=1}^{n} D_{ij};\ \text{n=total number of users} \tag{3.6}$$

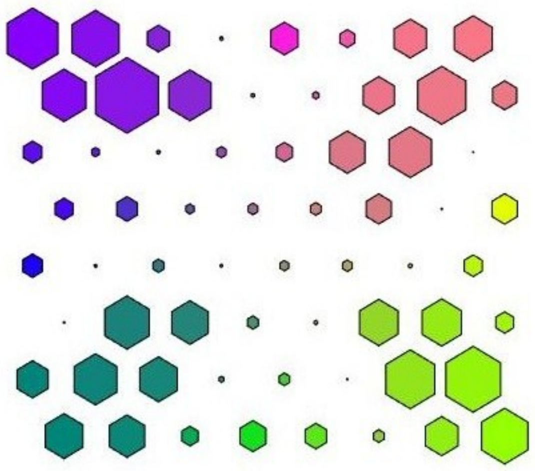
This is analogous to answering: *"How different (or exclusive) is the inclass user from every other user tested against?"*.

3.4.4 Results & Discussion

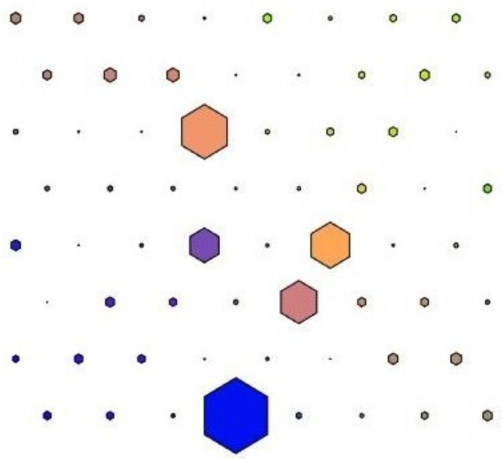
SOMs provide a visual aid to the data in high-dimensional space, called the unified distance matrix or U-matrix. Distance between neighboring map units of the SOM codebook are quantized in grayscale. The neurons responding to similar inclass samples, will converge to have similar neuronal weights. Thus, the distance between similar neurons on the SOM map is small, represented by grayscale value ≈ 0 (black). Whereas, map units with very different weight vectors shall have a greater distance, represented by grayscale value ≈ 255 (white). Intuitively, the white demarcations on the U-matrix, correspond to regions of similarity of the inclass data 'projected' onto the SOM. The diagonal subplots of Figure 3.6 are the U-matrices indicative of cluster separation ordered by color for each similarity group. It is worth noticing the updated map vectors per variable after the fine training stage of the SOM (compare upper right subplots of Figure 3.6 and Figure 3.7).

Dataset	Subset	Average ER (from SOM)	Friedman's test (K-means v/s SOM)
Rice LiveLab	Discrete	25.98	$2.7754e^{-07}$
	Continuous	23.30	$1.6200e^{-06}$
GCU Version 2	Discrete	37.92	0.0455
	Continuous	48.32	0.0455
GCU Version 1	Discrete	45.72	0.0253

Table 3.5: ERs from SOM on all datasets



(a)



(b)

Figure 3.5: SOM hits: (a) sample inclass SOM model; (b) sample outclass response.

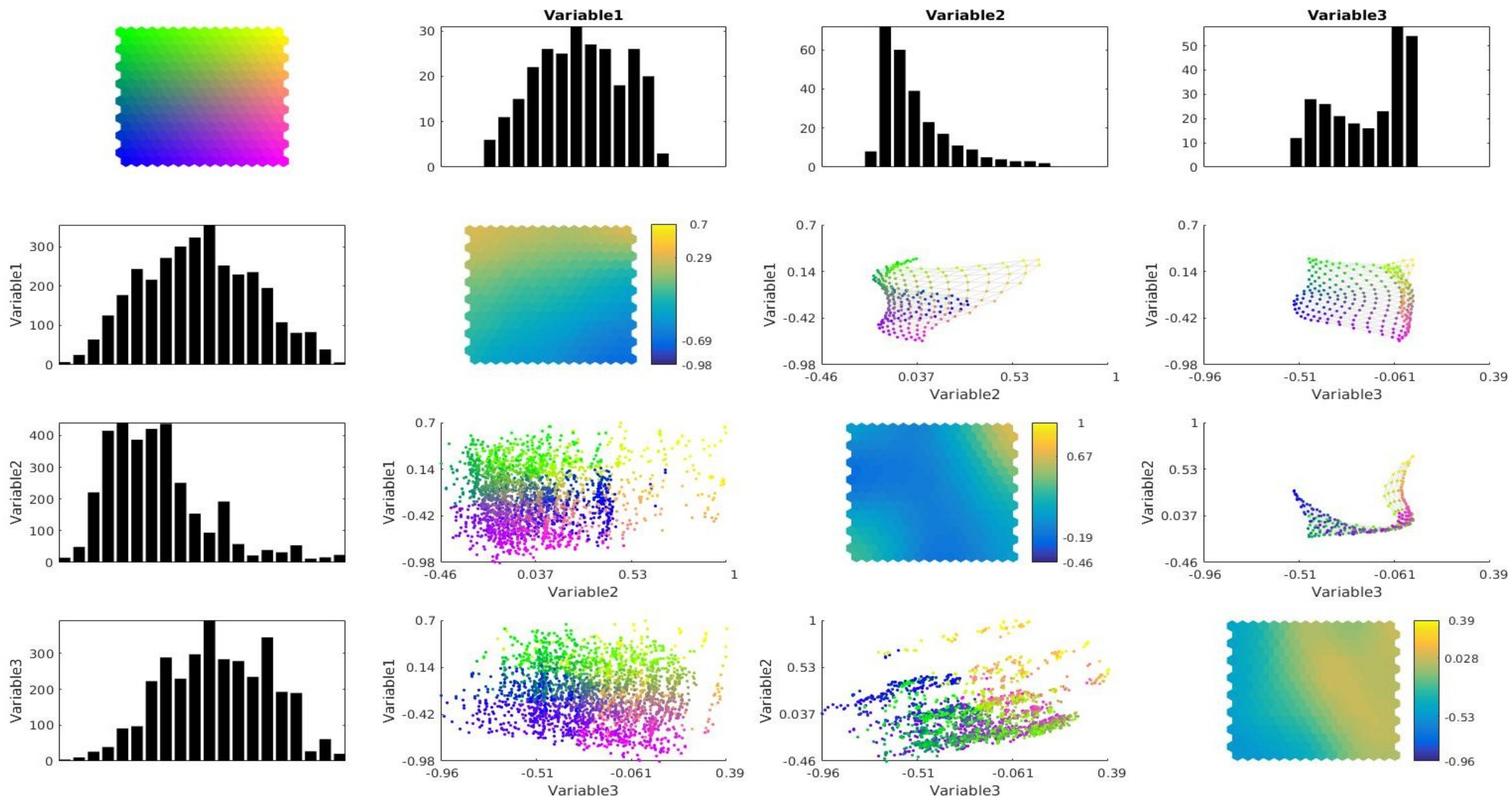


Figure 3.6: Visualizations from somtoolbox: result from coarse updates

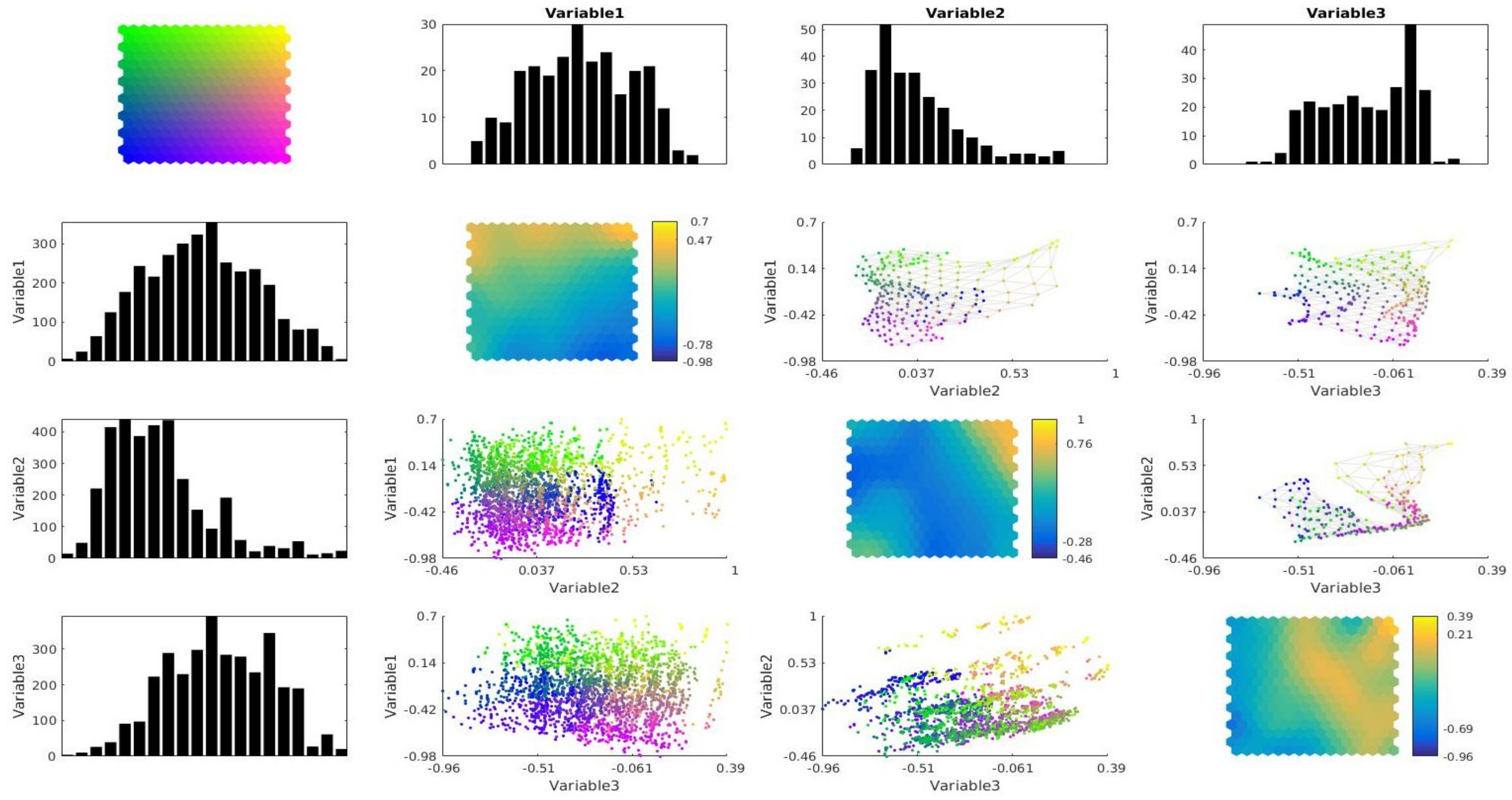


Figure 3.7: Visualizations from somtoolbox: result from fine updates.

On the smartphone datasets, the resulting SOM clusters are as shown in Figure 3.5. To assist visual inspection of the clustering process, the somtoolbox contains APIs that allow for us to plot the hit response from each neuron normalized by the ratio of samples they 'respond to' [51, 53]. It is easy to see that the most responsive model for the outclass data, *i.e.,* the blue hexagon is one of the least responsive neuron models in the learned inclass model.

Table 3.5 summarizes the detection rates from SOM. The Friedman's test is performed to identify whether there is any statistical significant difference between the performances of K-means and SOM learning techniques across multiple runs. Friedman's test is a non-parametric statistical significance test, used to validate the differences in treatments to a set of samples [8]. For the purpose of this research, treatments refer to the clustering techniques and samples are the derived ERs per user. If the $p - values$ are <0.05, it is concluded that the improvement in ERs is statistically significant. The results in Table 3.5 demonstrate that the performance of the SOM is statistically significantly better than the performance of the K-means learning algorithm.

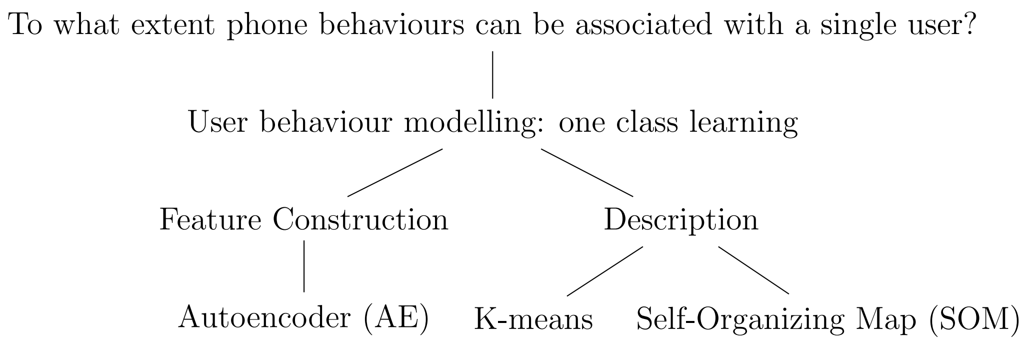
3.5 Summary

Henceforth in this research, SOM is continued to be used as the clustering method. In order to improve the detection ability of the high dimensional smart phone logs, the use of encoding is investigated.

Chapter 4

Proposed Method: AESOM framework

In this chapter, the proposed framework to improve the detection of behavioural patterns under one-class learning by using an encoding mechanism is explained. Two learning algorithms are used to build the inclass user behaviour model: autoencoder and self organizing map. The autoencoder is used to encode a 'signature' of the inclass behaviour. Self organizing map builds the data description against which anomalous behaviour will be characterized. The two neural networks are trained independently, but employ only the 'inclass' (user) data. The uniqueness of this approach is to drop the output layer of the autoencoder post training, and introduce self organizing map for expressing the user behaviour in a meaningful way without stepping outside the one-class learning constraint. The corresponding mind map is illustrated below:


To what extent phone behaviours can be associated with a single user?
User behaviour modelling: one class learning
Feature Construction
Description
Autoencoder (AE)
K-means
Self-Organizing Map (SOM)

4.1 Theory of Autoencoder

An autoencoder (AE) is a multilayer perceptron (MLP) with a bottleneck topology [30]. The input and output layers are of the same dimensionality, whereas the number of neurons in the hidden layers successively decreases. An iterative process might be adopted to incrementally construct steadily more complex AE. The goal is to identify the minimal topology to support a prior error goal. During training the input and output are simultaneously presented with the *same* exemplar, or one-class learning [30]. Naturally, the AE attempts to find a representation that 'autoencodes' the underlying properties of the data; care of the bottleneck topology. Such an architecture has been shown to be more robust (less sensitive) to parameter choices than 'one-class' SVMs [30]. The AE learning algorithm for a n-dimensional input $\mathcal{D}$, can be summarized as follows:

1. Initialize random weights to the network.

2. Define encoding function $\mathbf{E}$ as a mapping from $\mathrm{I\!R^n}$ to $\mathrm{I\!R^h}$, such that $h<n$. Let the encoding take place across two consecutive hidden layers of decreasing dimension. Each layer assumes neurons defined by:
   $\mathbf{E} = f(\vec{w} \cdot \vec{d} + \vec{b})$, where $f(x) = \frac{2}{(1+e^{-2x})} - 1$, $\vec{d}$ is the neuron input vector, $\vec{w}$ is the vector of weights between two consecutive layers of the MLP, and $\vec{b}$ is a vector of bias terms (one per neuron).

3. Define decoding function $\mathbf{D}$ that maps $\mathbf{E}(d)$ to d. Such a function is merely the neurons from hidden to output layer.

4. The weight ($\vec{w}$) and bias terms ($\vec{b}$) are identified using the back propagation algorithm with learning rate for each neuron adapted using conjugate gradient/scaled conjugate gradient information.[1]

5. Learning continues either until a minima is obtained or a maximum number of training epochs is encountered.

The most important design decisions take the form of establishing the number of hidden layers (generally two) and the corresponding number of hidden layer neurons. For this purpose, a greedy search for number of neurons in the hidden layer is performed under the contraint that, that hidden layer 1 $>$ hidden layer 2 and hidden layer 1 $<$ the n-dimensional input. The preferred architecture is that which encodes the training data with a minimal neuron count while also satisfying the training error goal. Post training, the output layer is dropped (it just recreates the input) and the

[1]Conjugate gradient (CG) optimization ensures that all previously computed 'descents' (albeit less steep) are toward the global minima [33]. This translates to no descent being 'discarded' during optimization, deeming it to be computationally less expensive in comparison to the steepest descent approach.

'exposed' hidden layer employed as the 'output' (to the self organizing map) fulfilling the one-class learning constraint [19, 30, 31, 32].

The feasibility of scaling the proposed solution on a mobile platform largely depends on the computational cost of the encoding step. This is due to the greedy search employed to design *each* layer of the AE. Empirically, two hidden layers balanced the trade-off between sufficient encoding and the computational cost in arriving at the minimal AE architecture.

4.2 Procedure

As discussed earlier, the aim is to explore the use of the AE and SOM pairing for user behaviour characterization based on smart phone usage information. The proposed approach is shown in Figure 4.1. The autoencoder, AE, reconstructs the data through a MLP configured in a bottleneck topology thereby encoding the most important properties of the users behaviour. The resulting feature encoding is then input to the SOM, so potentially establishing a set of signatures for the encoded user behaviours. The hypothesis of this research is that the use of the AE provides a more robust representation for characterizing user behaviours than would applying the SOM without the AE. Thus, a better discrimination should appear between different user behaviours under the architecture employing the AE step. Note that in both cases the AE does not require any additional training data than the SOM, or a one-class learning constraint.

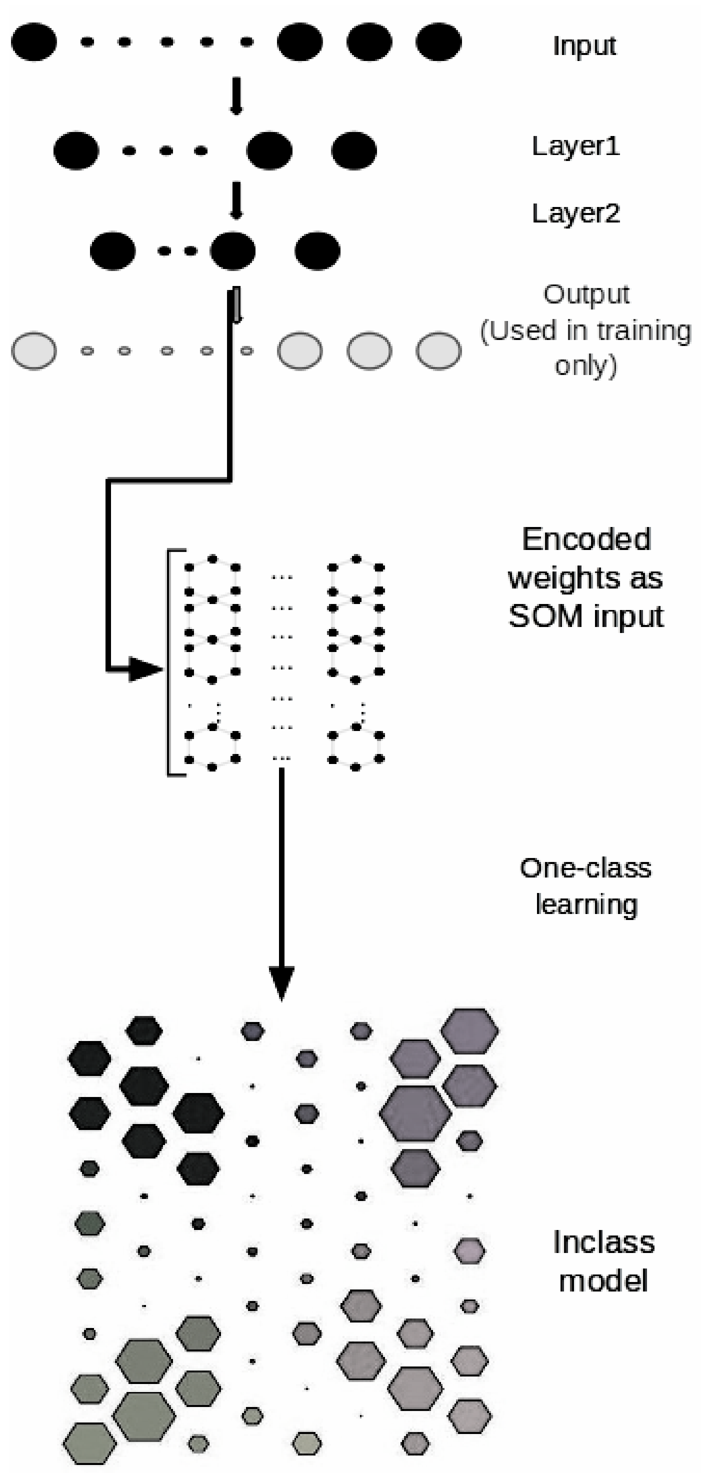


Figure 4.1: The proposed autoencoder based self organizing map approach for user behaviour characterization.

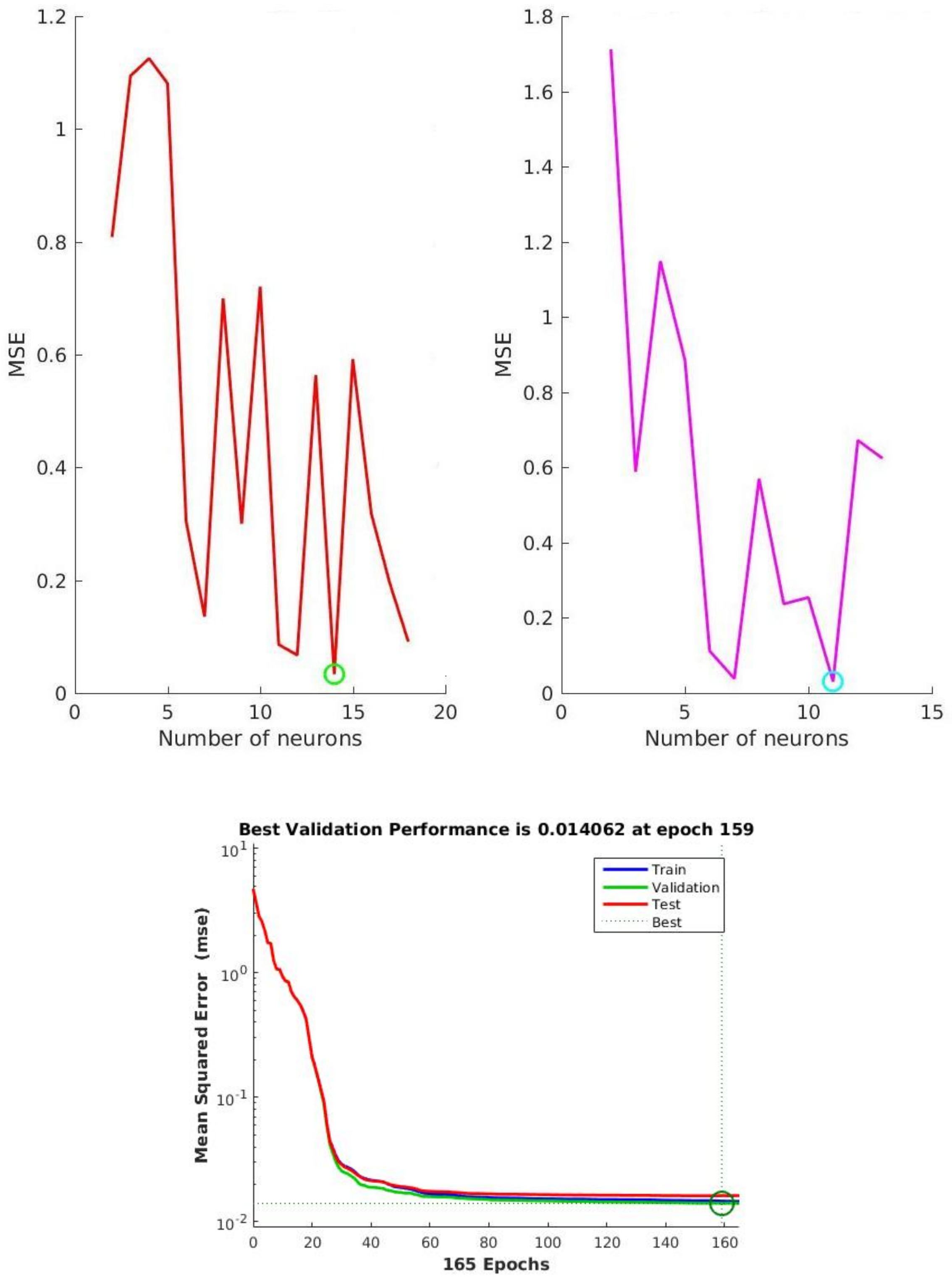


Figure 4.2: Obtaining the autoencoder architecture: (a) Elbow graphs for first and second hidden layers respectively; (b) Performance plot of the chosen architecture.

As mentioned in section 4.1, the autoencoder runs a greedy search to empirically fix the number of neurons in both the hidden layers. The size of the first hidden layer is the number of neurons that yield the least Mean Squared Error (MSE) in the Input-Hidden Layer-Output design, *i.e.* one hidden layer only. Using this as the upper bound, another search is performed to estimate the number of neurons in the second hidden layer that result in the least MSE. Figure 4.2 is illustrative of this process.

As shown in Figure 4.1, the encoded representation of inclass data is fed in to the SOM. The Table 4.1 details all the platforms and algorithm plugins used in this research.

4.3 Patterns identified: inspection

The proposed system was able to discover patterns of similar user behaviours. In the interest of space, the SOMs for GCU version 2 have been shown in the following subsection. However, all the SOMs for all the data sets can be found in the Appendix at the end of this thesis.

Discrete sensors

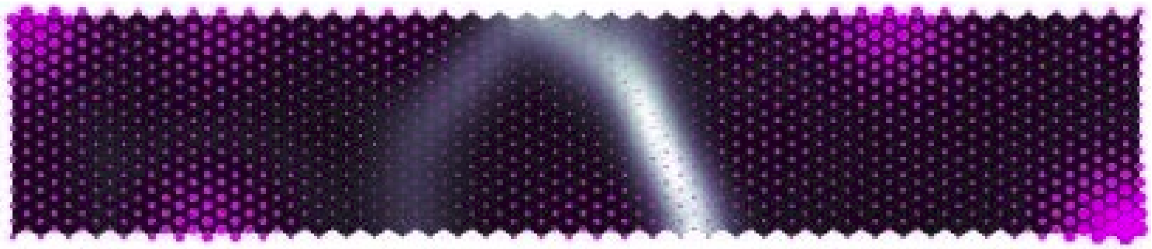
A behaviour model *i.e.*, SOM was built for each inclass user (one class learning). The resulting inclass SOM was subjected to outclass user data. Intuitively, similar behaviours are to receive similar responses from the SOM. This can be visually inspected in the Figure 4.3. The first subfigure represents the learned SOM model for the inclass user 1 and the corresponding SOM responses for the outclass users $2, 3$, and 4. The Figure 4.4(b) is the response from the 2^{nd} user's SOM model, and so on. The consistent observation from all the models is that, there are 2 distinctive clusters of SOM responses *viz.* pink-red and cyan-blue response patterns (see center of the SOM map).

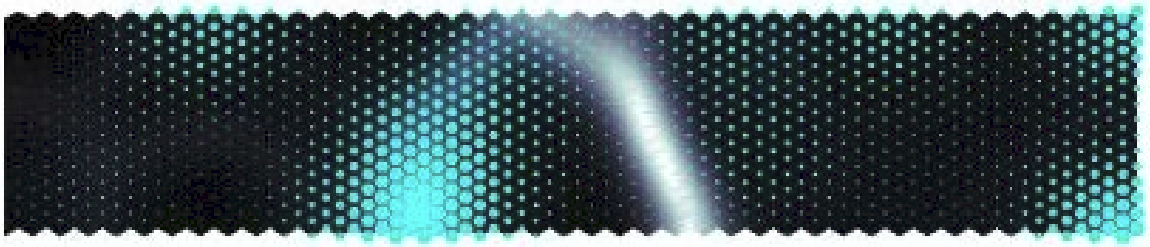
Continuous sensors

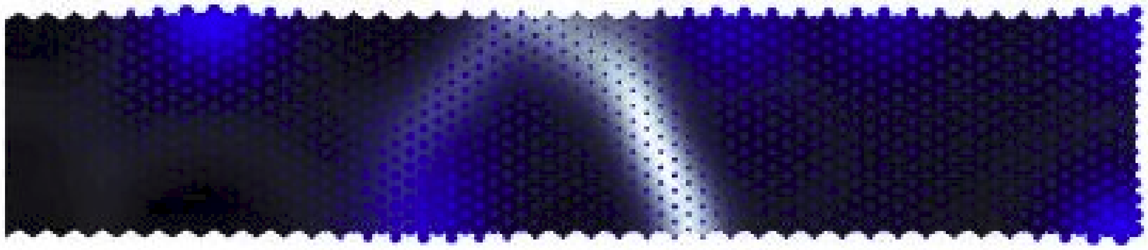
The similarity in patterns detected with this sensor subset is shown in Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10. With these sensors the 2 distinctive clusters of SOM responses are the pink-cyan and blue-red response patterns. Intuitively, it is

Module	Software	Parameter List	Parameter Value
Preprocessing	Python	Sampling Frequency	LiveLab: top 20 features, 15 minutes GCU: top 10 features, 15 minutes
		Normalization	[-1 +1]
Encoding	MATLAB Neural Network Toolbox	Epochs; Number of hidden layers	500; 2
		Activation function	tansig
		Back propagation	Scaled conjugate gradient
		Error	Mean squared error
Self Organizing Map	MATLAB somtoolbox [53]	Initial weights	Principal components initialization
		Lattice size	$5\sqrt{\text{No. of samples}}$
		Neighbourhood radius	N_c: [0.2*long edge, 0.05*short edge]
			N_f: [0.05*short edge, 0.05*short edge]
		Convergence	Until codebook ceases to change; N_s: 0.5*short edge

Table 4.1: Parameter configurations for the proposed framework.


(a)


(b)


(c)

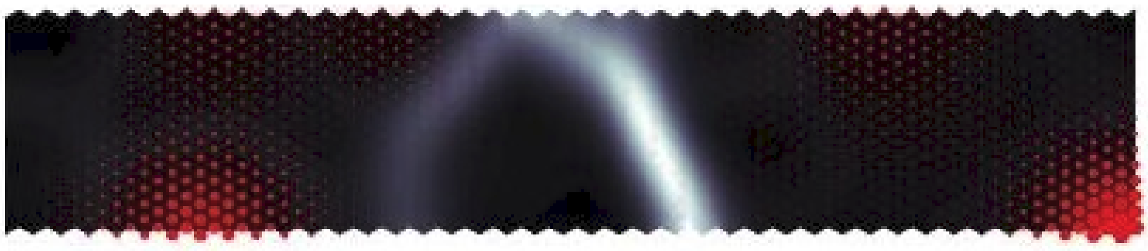
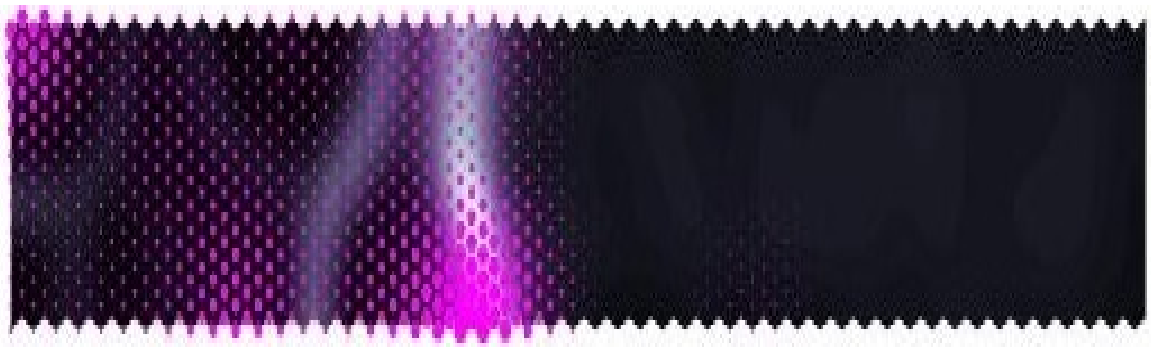
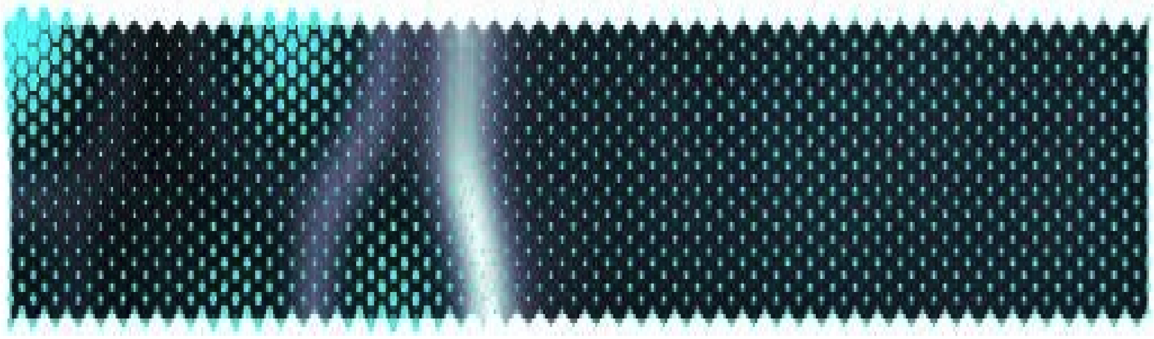
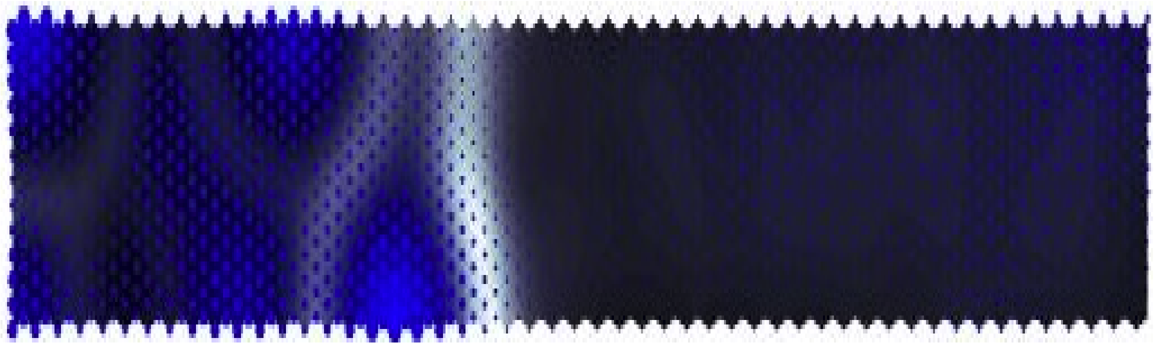

(d)

Figure 4.3: GCU version 2, discrete sensor subset: (a) U1 as inclass; (b-d) ¬ U1 as outclass.


(a)


(b)


(c)

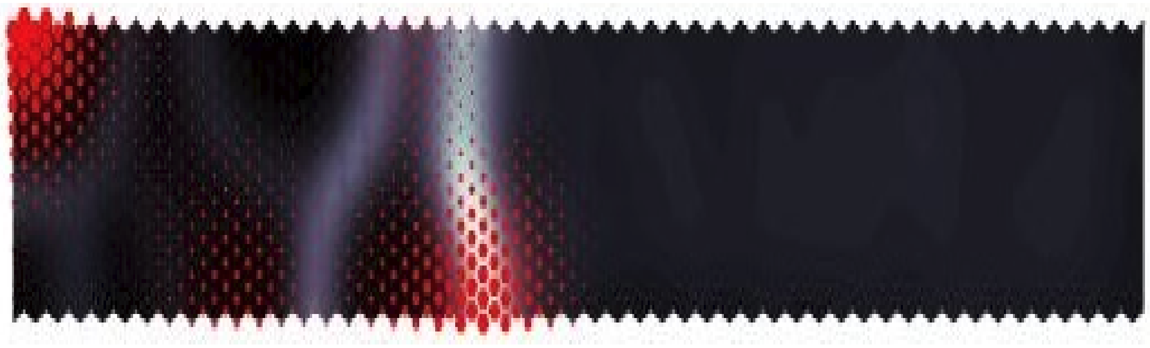
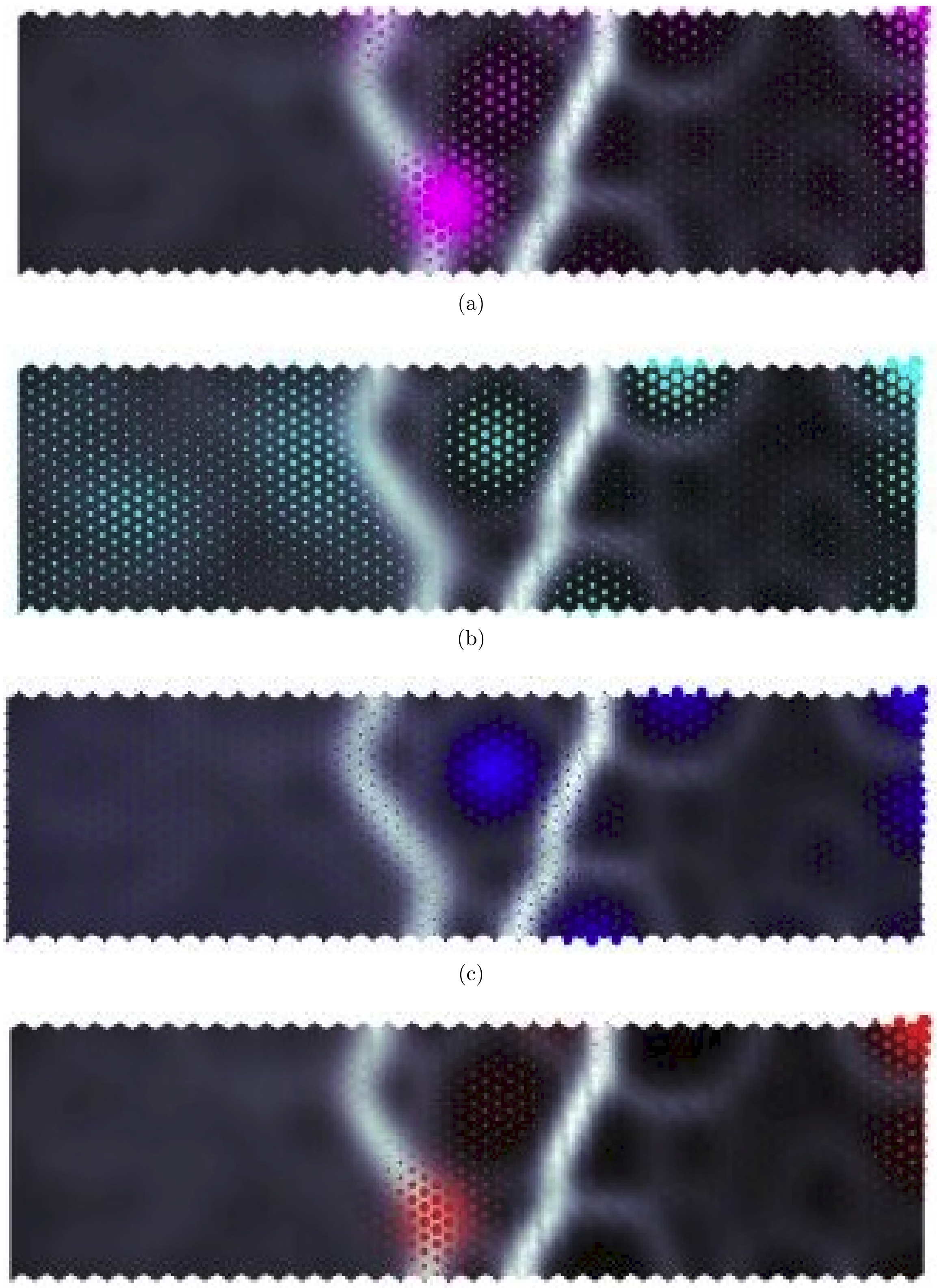

(d)

Figure 4.4: GCU version 2, discrete sensor subset: (a) U2 as inclass; (b-d) ¬ U2 as outclass.



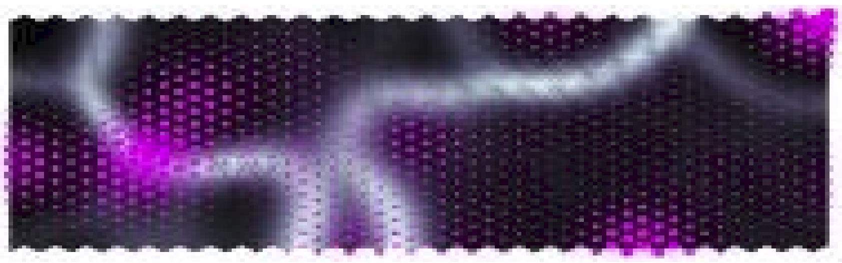
(a)

(b)

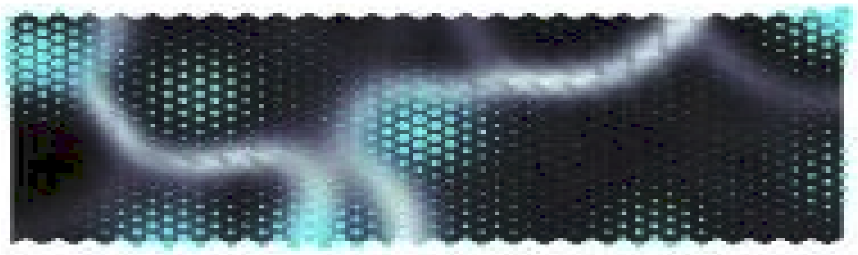
(c)

(d)

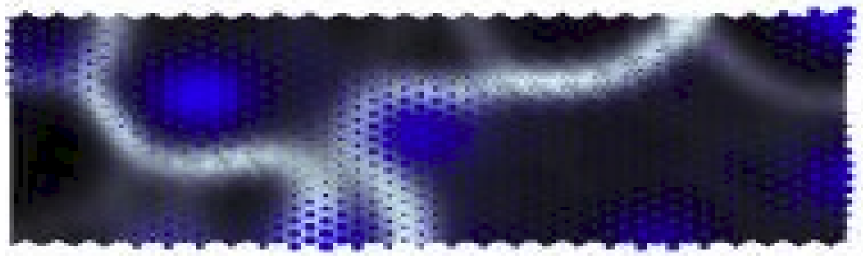
Figure 4.5: GCU version 2, discrete sensor subset: (a) U3 as inclass; (b-d) ¬ U3 as outclass.



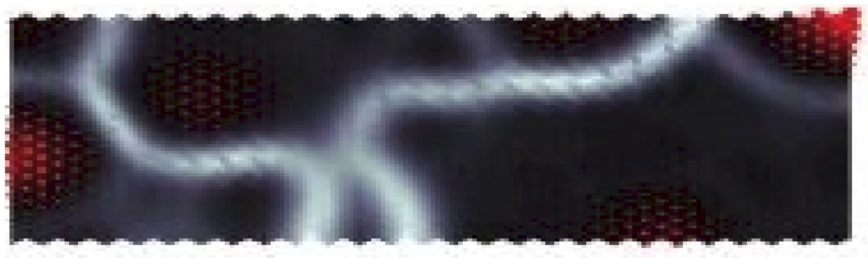
(a)



(b)



(c)



(d)

Figure 4.6: GCU version 2, discrete sensor subset: (a) U4 as inclass; (b-d) ¬ U4 as outclass.

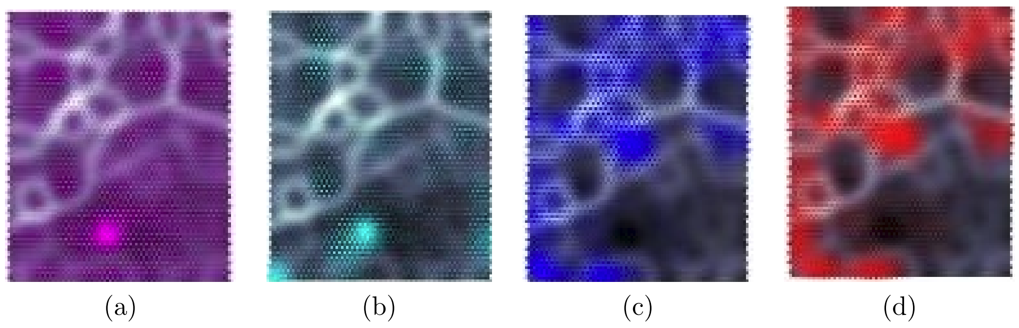


Figure 4.7: GCU version 2, continuous sensor subset: (a) U1 as inclass; (b-d) ¬ U1 as outclass.

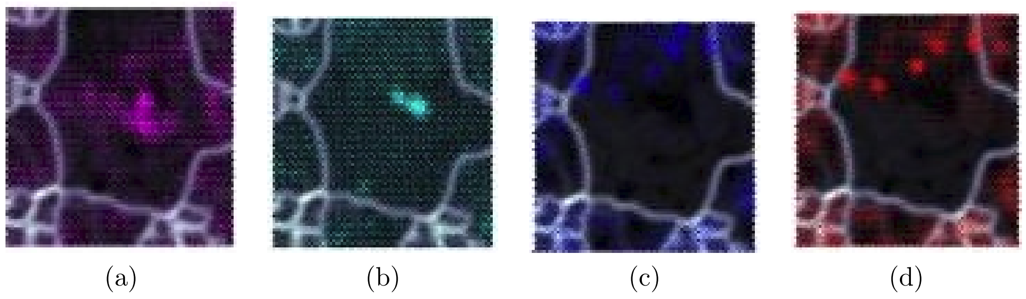


Figure 4.8: GCU version 2, continuous sensor subset: (a) U2 as inclass; (b-d) ¬ U2 as outclass.

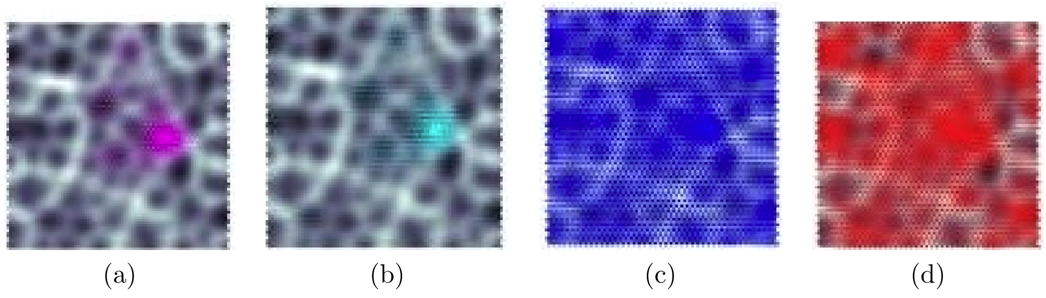


Figure 4.9: GCU version 2, continuous sensor subset: (a) U3 as inclass; (b-d) ¬ U3 as outclass.

safe to presume that the framework is able to discover similarities in user behaviour regardless of the chosen sensor subset.

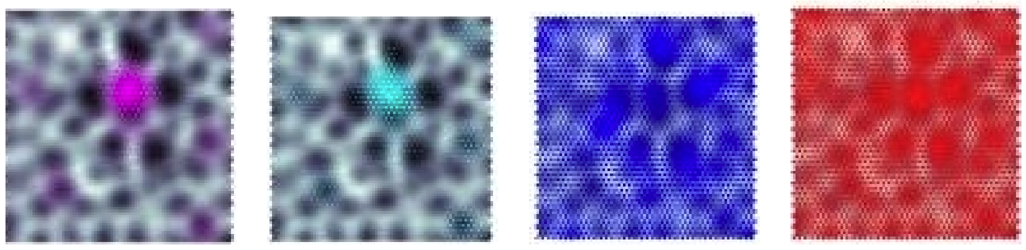
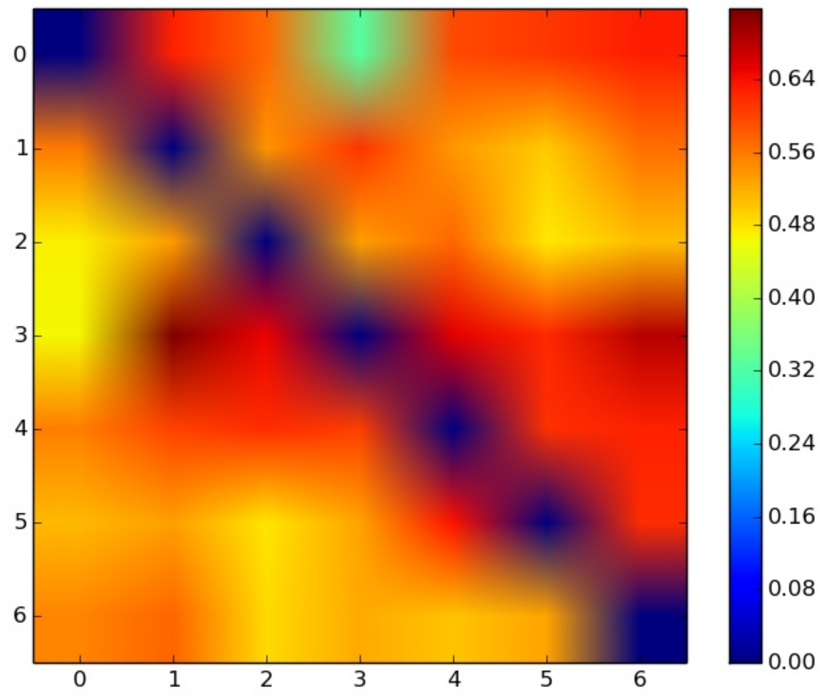


Figure 4.10: GCU version 2, continuous sensor subset: (a) U4 as inclass; (b-d) ¬ U4 as outclass.

4.4 Heat Maps of User Dissimilarities

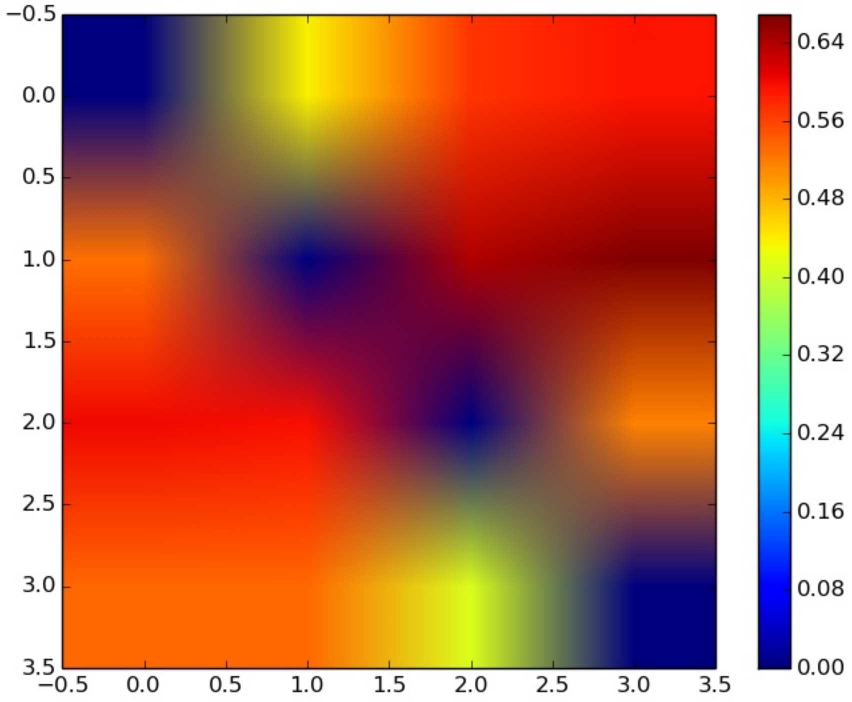
Heat maps are a 2D graphical representation of data contained in a matrix format. Each row-column indexed value of the data matrix is organized by a color scale. The darker shades of the color bar are used to represent higher numeric values and the lighter shades are used to represent the lesser values. This type of visualization helps to acknowledge that user exclusivity is not symmetric especially, in the one-class learning paradigm. Furthermore, the heat maps are perceptually more convenient (in comparison to bar plots) for visualizing results for large datasets.

Figure 4.11 illustrates the heat map of average user dissimilarities from 20 runs of the proposed method for all the 3 datasets. The discrete sensor subset (applications, cell tower and websites) yield a higher rate of user exclusivity, in comparison to the continuous sensor subset. From the continuous sensor subset, the framework is able to discern two users at 56% in the LiveLab dataset and at 64% using the GCU data set. On the contrary, discrete sensor subset allows for 90% and 80% dissimilarity detection respectively. It is to be noted that, under the one-class learning constraint, the learning of inclass user behaviour begins without any prior information. Dissimilarity is only attributed to how differently the trained SOM responds to unseen data (outclass), and that no *a-priori* information about behaviour patterns are forced in the quantification.

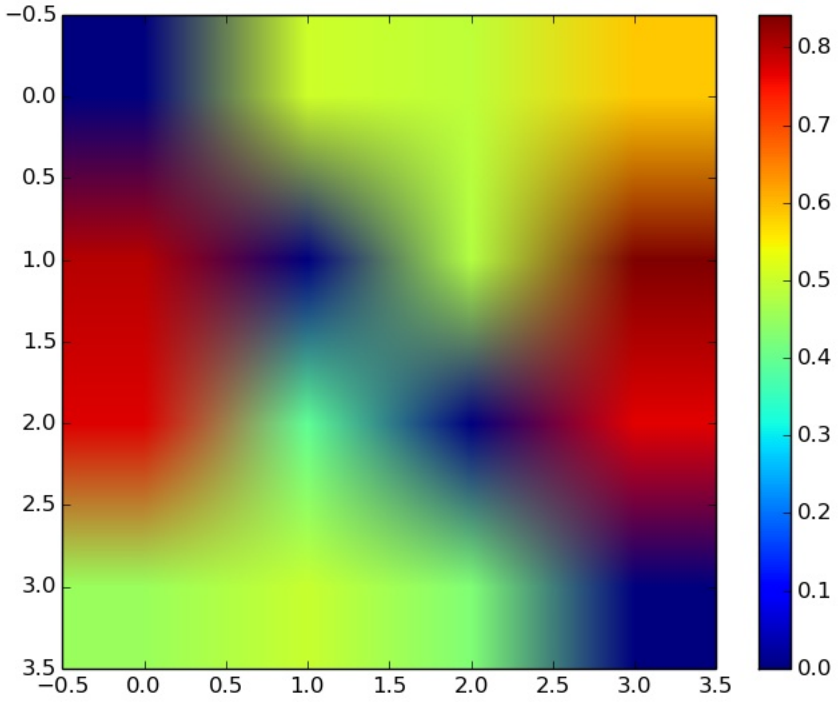


(a)

Figure 4.11

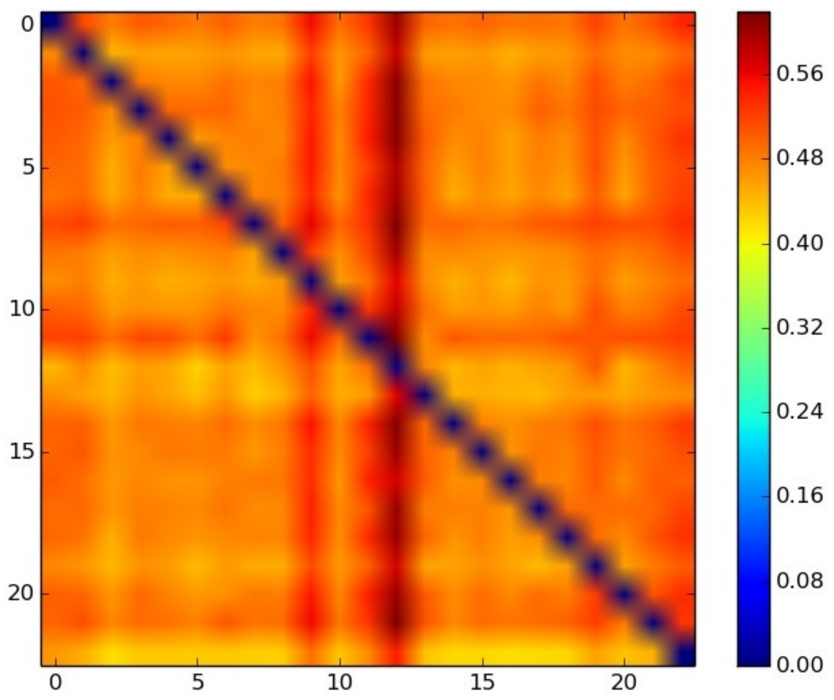


(a)

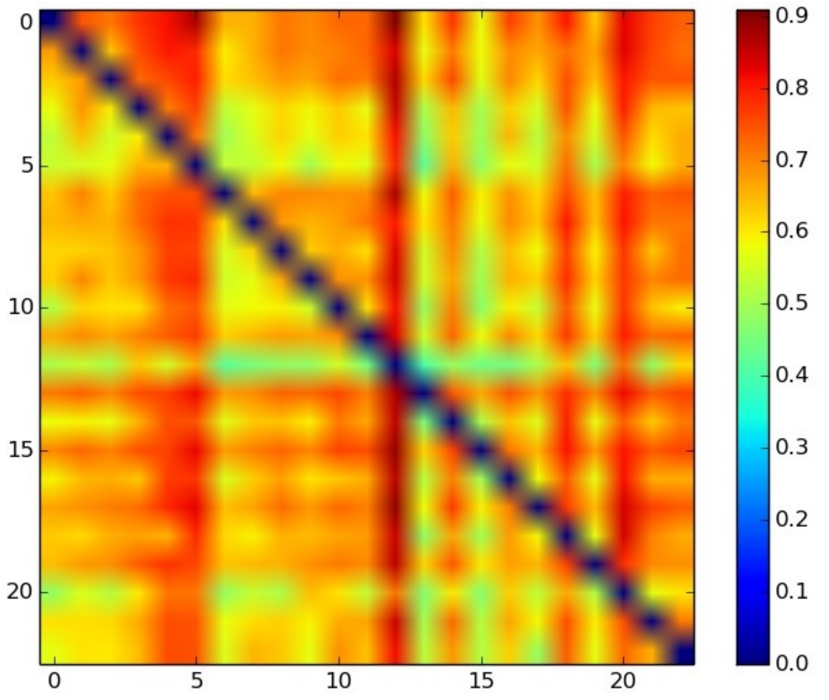


(b)

Figure 4.11



(a)



(b)

Figure 4.11: Average ERs (among any two users) across 20 runs of the AESOM framework: (a) GCU-V1 Discrete sensors; (b) GCU-V2: Continuous sensors; (c) GCU-V2: Discrete sensors; (d) LiveLab traces: Continuous sensors; (e) LiveLab traces: Discrete sensors.

4.5 Contribution of the AE: Improvement in detection rates

The average of all per-user dissimilarity rate (as reported in the previous section), results in the per-user exclusivity rate (ER). Figure 4.12, Figure 4.13, Figure 4.14 show the distribution of average ERs across 20 runs of the model; with and without the encoding step. The improvement in ERs from the autoencoder remains consistent across both sensor subsets for the LiveLab and GCU datasets. T-test and Friedman's test results summarized in Table 4.2.

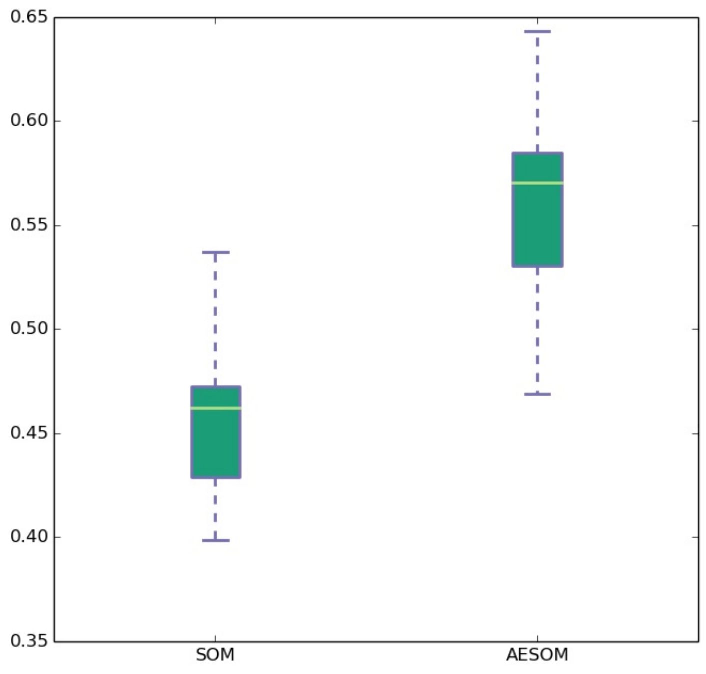
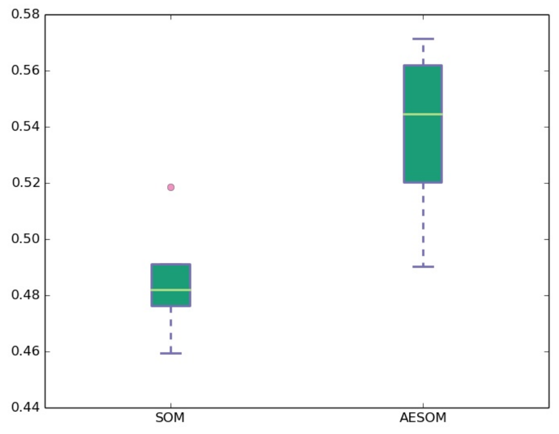
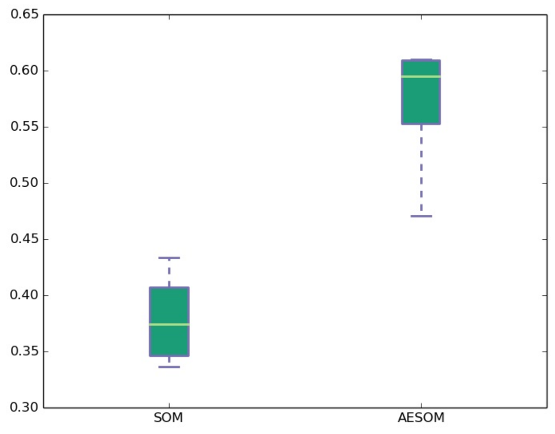


Figure 4.12: Box plots of ERs comparing SOM and AESOM framework: GCU-V1 discrete sensors.

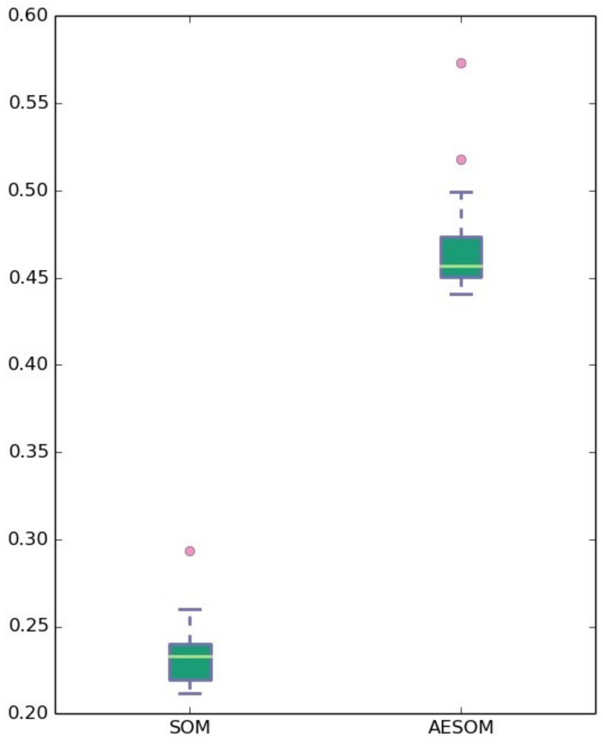


(a)

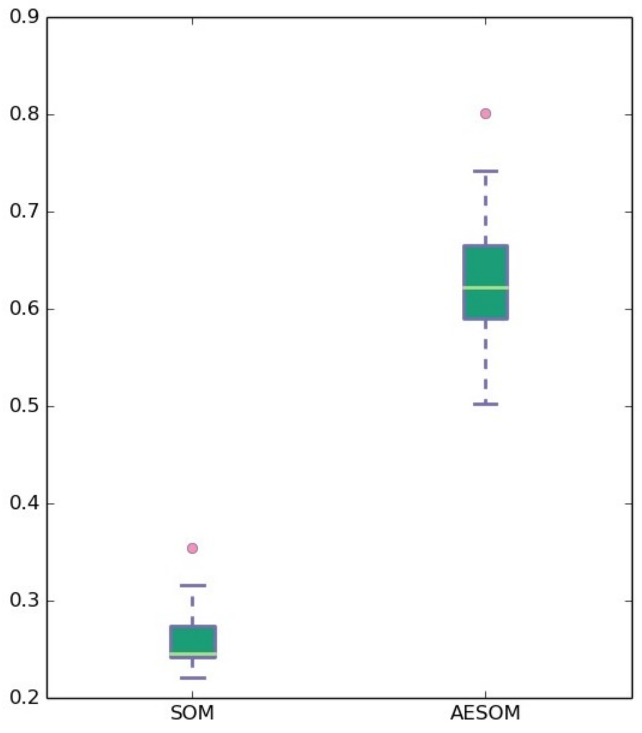


(b)

Figure 4.13: Box plots of ERs comparing SOM and AESOM framework: (a) GCU-V2: Continuous sensors; (b) GCU-V2: Discrete sensors.



(a)



(b)

Figure 4.14: Box plots of ERs comparing SOM and AESOM framework:(a) LiveLab traces: Continuous sensors; (b) LiveLab traces: Discrete sensors.

4.6 Summary of results

In order to demonstrate the reliability of the behaviour mining framework on smartphone usage datasets, it is necessary to test the statistical significance of our results. To this end, we assess the following:

1. Variation in ERs across multiple runs of the Autoencoder based Self Organizing Map (AESOM) framework.

2. Improvement in ERs from encoding the features of the data.

We employ the two tailed T-test and Friedman's test to understand whether the improvement in the results of the AE based SOM are statistically significant or not, in terms of increasing the detection rates across multiple runs. The results show that the performance improvement of the AESOM is statistically significant compared to the SOM.

Dataset	Sensor subset	SOM	AESOM	T-test	Friedman's test	Overall benefit
LiveLab	Continuous	23.30	46.81	$1.32e^{-25}$	$1.62e^{-6}$	23.5
	Discrete	25.98	63.31	$3.94e^{-32}$	$1.62e^{-6}$	
GCU-V2	Continuous	48.32	54.85	0.117	0.0455	2.11
	Discrete	37.92	56.74	0.00326	0.0455	
GCU-V1	Discrete	45.72	55.91	0.00259	0.0073	10.19

Table 4.2: Results of dissimilarity from the proposed AESOM framework across all the 3 datasets.

Chapter 5

Conclusion and Future work

In summary, the focus of this thesis was to model user behaviour using smartphone data. To this end, an autoencoder was used to build a descriptive representation of each user's behavioural signatures. Then, an unsupervised SOM was employed as the descriptive clustering solution. Various initialization and training modes for the SOM were investigated. A linearly initialized SOM, *w.r.t.* principal components of the encoded data, was finalized using topographic error to measure the resulting map qualities. Both the AE and SOM modules are trained under a one-class learning constraint. This proposed framework was evaluated on 3 publicly available datasets under two use cases *viz,* contextual (discrete) and physiological (continuous).

The LiveLab traces get the highest per user average ER, 90% in the discrete sensors set. This can be attributed to nearly 1 year long usage trace in the LiveLab dataset. The improvement in using the discrete sensor set (in comparison to the continuous set) is recorded as 23.5% for LiveLab and 2.11% for GCU (see Table 4.2). However, it is to be noted that the GCU version 2 datasets consists of similar users behaviours. Analysing the improvements among 'similar' and 'dissimilar' users in the continuous sensor subset of GCU version2, shows an improvement of 12% from the autoencoder (see Figure 5.1). Evaluation on GCU version 1 reports an improvement of 10.19% from using the autoencoder. The results from SOM across all the datasets show that, a user is better modelled using his/her contextual information such as application/ website usage and cell tower/ WiFi connectivity.

Since the smartphone is always exposed to interaction with its owner only under normal situations, any outclass data would be suspicious behaviour. Hence the one-class learning constraint and its usage for security domain. In the realm of single user interacting with his/her own device, it is safe to conclude that the AESOM framework is able to identify similar users, within the one-class learning restriction.

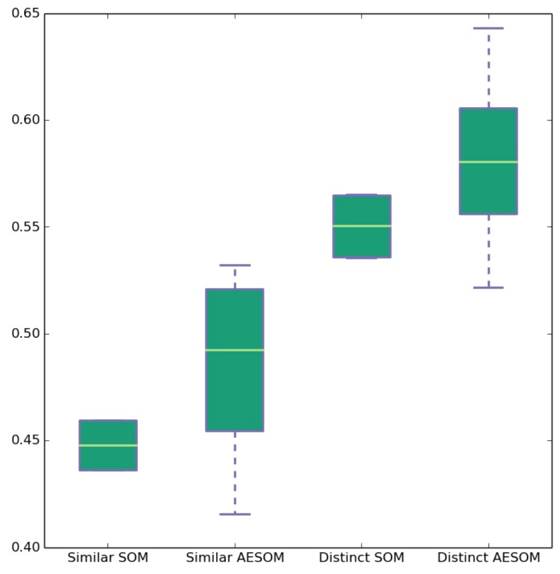


Figure 5.1: Improvement in average ER across similar users as detected by SOM indepedently.

Extending the current scope of AESOM framework

The following key inferences are drawn from this research:

- Phone behaviours can be associated to a large extent to a single user.
- An encoding of input data improves the quality of the data description *i.e.,* clustering solution.
- Any user is better profiled by subliminal engagement with the device (via applications and internet usage) in comparison to the user's gait, posture or how the phone is physically handled (via the continuous sensors).

However, a major reservation in the current scope of the research is, the temporal *sequence* in smartphone usage. The standard SOM does not have the ability to 'remember' a sequence (or historic) events. Research has shown that a sequence of events is a better temporal anchor in comparison to a snapshot of events at a particular time namely, timestamp information [22, 27]. The suggested solution to enhance the ability of the AESOM framework is to incorporate shift registers. The resulting inclass data frame takes the following form.

$$t_4, t_3, t_2, t_1, t_0$$

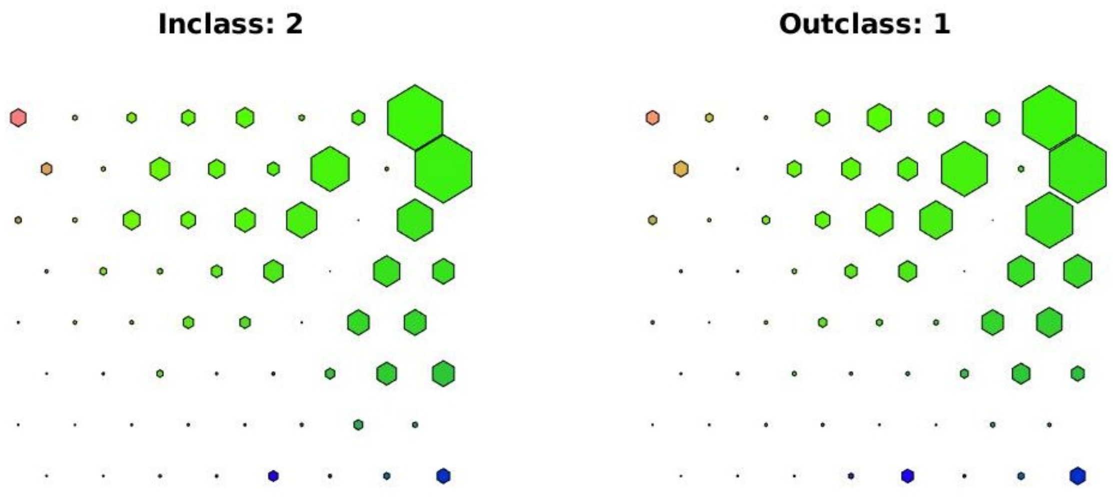
$$t_5, t_4, t_3, t_2, t_1$$
$$t_6, t_5, t_4, t_3, t_2$$
$$t_7, t_6, t_5, t_4, t_3$$
$$\text{and so on..}$$

where, t_i : $\mathit{encoded}\,(s_1, s_2, s_3, ...s_n)$ at i^{th} minute for all n sensors. Therefore, each shift is the *estimated* change in behaviour every minute. The window of the shift register is to be parameterized empirically. Figure 5.2 shows the SOM responses of the two similar users from GCU version 2 data set. It is evident (albeit visually) that, incorporating sequence information to summarize change in behaviour allows SOM to better distinguish even the very similar user behaviours.

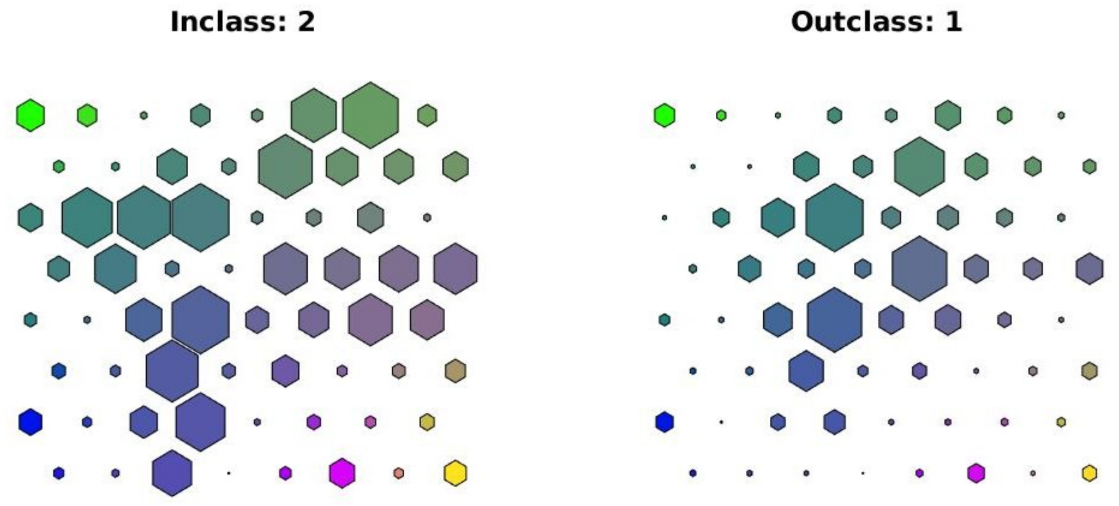
Consider the case of two users who might use the same set of applications, connect to the same WiFi access points and/or be located in the same neighbourhood, thus using the same cell towers. Although these users seem to be similar by the content of phone usage, they may vary substantially by the sequence of use, *i.e.,* history of behaviour. Therefore, employing shift registers to the encoded feature map will instigate a temporal pattern of user-device interaction. As a result, isolated changes in the user behaviour will not belong to any historic behaviour sequence. Therefore, the shift register is better able to differentiate 'shift' in the inclass user's behaviour from the rarely occurring abnormal activity. Furthermore, the ability of embedding temporal sequences to the clustering, reduces the need for frequent re-training of the model.

Sampling frequency and adaptive sampling

One of the important parameters in this study is the rate at which the features are sampled/monitored through the day. To understand the effect of the sampling interval on the performance, the system was using data sampled at every $5, 15$ and 30 minute intervals. For each sampling interval, the $\mathcal{N}$ most frequently used applications, cell towers, websites are also empirically analyzed. To this end, I tested $\mathcal{N} = 5, 10, 20$ features. Figure 5.3 shows the results of this analysis. Based on these results, I suggest to use the sampling frequency of 15 minutes with 20 features. Indeed, a periodic assessment of the nature of usage traces/logs sensor logs would also improve the efficiency of the deployed system. For example, once the user model has been



(a)



(b)

Figure 5.2: Visual inspection of SOM clusters after embedding sequence information of behaviour: (a) Highly correlated neuronal responses from inclass $user_2$ SOM model, to the outclass $user_1$ data; (b) Neuronal responses very strongly correlated for BMUs only, intensity of hits vary in less active neurons from inclass $user_2$ model to outclass $user_1$ data.

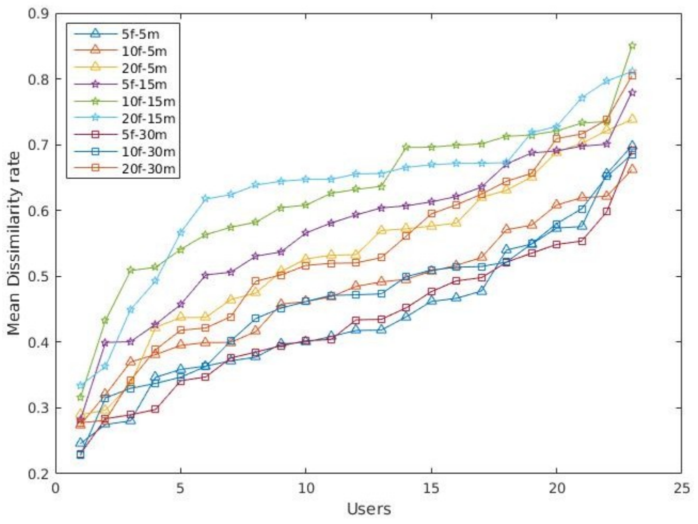


Figure 5.3: Empirical study with different sampling intervals and top $\mathcal{N}$ features.

learned by the system, reducing the sampling frequency for model updates could be a possible way to taper down the consumption of smart phone resources.

Sequential feature selection (SFS) can be employed to answer either of the following questions:

- Leave one out analysis: Which sensor is not important to be profiles for each user?
- Combinatorial analysis: Which combination of sensors best model each user?

In particular, forward SFS starts with an empty sensor set. It continues to add a sensor to the study if and only if, the addition improves the overall ER of an user. Although this increases the initial training time, it reduces the computational cost of the proposed framework in the deployment stage. This approch to user representation is particularly beneficial in the case of insider attacks wherefore, subtle changes in user behaviour have to be identified.

The limitation of this thesis arises when multiple users interact with a single mobile device. Such is the case when faculty use a registered mobile device to take classroom attendance, or when a waitress at a resturant uses the same iPad device to take food orders. When the AESOM framwork is deployed on a mutli-user platform, it can be configured to detect abnormal usage and quantify similariites in the 'communities' of users interacting with this device. Futhermore, cyber security policies can be used as guidance in order to establish ‘normal’ usage thresholds. An upper limit on the amount of data transmitted over the internet through the day is a good example.

Lastly, the discussion on security of smart devices is not complete without the use of such techniques in resolving security breaches on the *Internet of Things*. Consider the vulnerability posed by APIs used in the IoT devices for data transmission (for instance). Within this context, the proposed AESOM framework can be adapted to monitor high risk application behaviour. This is possible by logging the sequence and frequency of APIs and other system level function calls invoked within the firmware of the smart device. Wherefore, inclass translates to respresent the baseline application behaviour. On the contrary, a bizzare sequence of API calls may result in a new ’pattern’ in the application’s behaviour. Thereby increasing the amount of dissimilarity from the inclass patterns, modelled under the one-class learning constraint. This flexibility in modelling behaviour, either user, device or application, allows the proposed AESOM framework to be used in improving various security mechanisms in IoT.

Bibliography

[1] Emelie Andersson and Frida Frost. The use values of smartphone apps-a qualitative study. 2013.

[2] Gunes Kayacik Mike Just Lynne Baillie and David Aspinall Nicholas Micallef. Data driven authentication: On the effectiveness of user behaviour modelling with mobile device sensors.

[3] David Barrera, H Güneş Kayacik, Paul C van Oorschot, and Anil Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 73–84. ACM, 2010.

[4] Attaullah Buriro, Bruno Crispo, Filippo Del Frari, and Konrad Wrona. Touchstroke: smartphone user authentication based on touch-typing biometrics. In *International Conference on Image Analysis and Processing*, pages 27–34. Springer, 2015.

[5] Senaka Buthpitiya, Ying Zhang, Anind K Dey, and Martin Griss. N-gram geotrace modeling. In *International Conference on Pervasive Computing*, pages 97–114. Springer, 2011.

[6] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Personalization and user verification in wearable systems using biometric walking patterns. *Personal and Ubiquitous Computing*, 16(5):563–580, 2012.

[7] Nathan L Clarke and Steven M Furnell. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6(1):1–14, 2007.

[8] Wayne W Daniel et al. Applied nonparametric statistics. 1990.

[9] Mohammad Omar Derawi, Claudia Nickel, Patrick Bours, and Christoph Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pages 306–311. IEEE, 2010.

[10] Trinh-Minh-Tri Do and Daniel Gatica-Perez. By their apps you shall understand them: mining large-scale patterns of mobile phone usage. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, page 27. ACM, 2010.

[11] Anders Drachen, Alessandro Canossa, and Georgios N Yannakakis. Player modeling using self-organization in tomb raider: Underworld. In *2009 IEEE symposium on computational intelligence and games*, pages 1–8. IEEE, 2009.

[12] Nathan Eagle and Alex Sandy Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268, 2006.

[13] Nathan Eagle and Alex Sandy Pentland. Eigenbehaviors: Identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, 2009.

[14] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[15] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14. ACM, 2011.

[16] Peter Gilbert, Byung-Gon Chun, Landon P Cox, and Jaeyeon Jung. Vision: automated security validation of mobile apps at app markets. In *Proceedings of the second international workshop on Mobile cloud computing and services*, pages 21–26. ACM, 2011.

[17] Nathaniel Husted, Hassen Saïdi, and Ashish Gehani. Smartphone security limitations: conflicting traditions. In *Proceedings of the 2011 Workshop on Governance of Technology, Information, and Policies*, pages 5–12. ACM, 2011.

[18] Ankita Jain and Vivek Kanhangad. Exploring orientation and accelerometer sensor data for personal authentication in smartphones using touchscreen gestures. *Pattern Recognition Letters*, 68:351–360, 2015.

[19] N. Japkowicz, C. Myers, and M. Gluck. A novelty detection approach to classification. In *Proceeding of the Fourteenth International Conference On Artificial Intelligence*, pages 518–523, 1995.

[20] Woongryul Jeon, Jeeyeon Kim, Youngsook Lee, and Dongho Won. A practical analysis of smartphone security. In *Symposium on Human Interface*, pages 311–320. Springer, 2011.

[21] RC Johnson, Walter J Scheirer, and Terrance E Boult. Secure voice-based authentication for mobile devices: vaulted voice verification. In *SPIE Defense, Security, and Sensing*, pages 87120P–87120P. International Society for Optics and Photonics, 2013.

[22] H Gunes Kayacik, A Nur Zincir-Heywood, and Malcolm I Heywood. A hierarchical som-based intrusion detection system. *Engineering Applications of Artificial Intelligence*, 20(4):439–451, 2007.

[23] Teuvo Kohonen. Essentials of the self-organizing map. *Neural Networks*, 37:52–65, 2013.

[24] Max Landman. Managing smart phone security risks. In *2010 Information Security Curriculum Development Conference*, pages 145–155. ACM, 2010.

[25] Wei-Han Lee and Ruby B Lee. Multi-sensor authentication to improve smartphone security. In *Conference on Information Systems Security and Privacy*, pages 1–11, 2015.

[26] Lingjun Li, Xinxin Zhao, and Guoliang Xue. Unobservable re-authentication for smartphones. In *NDSS*, 2013.

[27] Peter Lichodzijewski, A Nur Zincir-Heywood, and Malcolm I Heywood. Host-based intrusion detection using self-organizing maps. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1714–1719. IEEE, 2002.

[28] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. Can your smartphone infer your mood. In *PhoneSense workshop*, pages 1–5, 2011.

[29] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. Moodscope: Building a mood sensor from smartphone usage patterns. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 389–402. ACM, 2013.

[30] L. M. Manevitz and M. Yousef. One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

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

[32] M. Markou and S. Singh. Novelty detection: a review – part 2: neural network approaches. *Signal Processing*, 83:2499–2521, 2003.

[33] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[34] F Murtagh and M Hernández-Pajares. The kohonen self-organizing map method: an assessment. *Journal of Classification*, 12(2):165–190.

[35] Alexios Mylonas, Stelios Dritsas, Bill Tsoumas, and Dimitris Gritzalis. Smartphone security evaluation the malware attack case. In *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*, pages 25–36. IEEE, 2011.

[36] Alexios Mylonas, Dimitris Gritzalis, Bill Tsoumas, and Theodore Apostolopoulos. A qualitative metrics vector for the awareness of smartphone security users. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 173–184. Springer, 2013.

[37] James Newsome and Dawn Song. Dynamic taint analysis: Automatic detection, analysis, and signature generation of exploit attacks on commodity software. In *In In Proceedings of the 12th Network and Distributed Systems Security Symposium*. Citeseer, 2005.

[38] Claudia Nickel, Tobias Wirtl, and Christoph Busch. Authentication of smartphone users based on the way they walk using k-nn algorithm. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2012 Eighth International Conference on*, pages 16–20. IEEE, 2012.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] Shari Lawrence Pfleeger and Deanna D Caputo. Leveraging behavioral science to mitigate cyber security risk. *Computers & security*, 31(4):597–611, 2012.

[41] Bahman Rashidi, Carol Fung, and Tam Vu. Dude, ask the experts!: Android resource access permission recommendation with recdroid. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 296–304. IEEE, 2015.

[42] Vaibhav Rastogi, Yan Chen, and William Enck. Appsplayground: automatic security analysis of smartphone applications. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 209–220. ACM, 2013.

[43] Partha Pratim Ray. Rays scheme: graphical password based hybrid authentication system for smart hand held devices. *International journal of computer trends and technology*, 3:235–241, 2012.

[44] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[45] Clayton Shepard, Ahmad Rahmati, Chad Tossell, Lin Zhong, and Phillip Kortum. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2011.

[46] Anastasia Skovoroda and Dennis Gamayunov. Review of the mobile malware detection approaches. In *Parallel, Distributed and Network-Based Processing*

(PDP), 2015 23rd Euromicro International Conference on, pages 600–603. IEEE, 2015.

[47] Ralf Steuer, Jürgen Kurths, Carsten O Daub, Janko Weise, and Joachim Selbig. The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics*, 18(suppl 2):S231–S240, 2002.

[48] Dusan Stevanovic, Natalija Vlajic, and Aijun An. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Applied Soft Computing*, 13(1):698–708, 2013.

[49] Tim Stöber, Mario Frank, Jens Schmitt, and Ivan Martinovic. Who do you sync you are?: smartphone fingerprinting via application behaviour. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 7–12. ACM, 2013.

[50] Matthias Trojahn and Frank Ortmeier. Toward mobile authentication with keystroke dynamics on mobile phones and tablets. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 697–702. IEEE, 2013.

[51] Tommi Vatanen, Maria Osmala, Tapani Raiko, Krista Lagus, Marko Sysi-Aho, M Orešič, Timo Honkela, and Harri Lähdesmäki. Self-organization and missing values in som and gtm. *Neurocomputing*, 147:60–70, 2015.

[52] Sergio Valero Verdú, Mario Ortiz Garcia, Carolina Senabre, A Gabaldón Marín, and Francisco J Garcıa Franco. Classification, filtering, and identification of electrical customer load patterns through the use of self-organizing maps. *IEEE Transactions on Power Systems*, 21(4):1672–1682, 2006.

[53] Juha Vesanto, Johan Himberg, Esa Alhoniemi, Juha Parhankangas, et al. Self-organizing map in matlab: the som toolbox. In *Proceedings of the Matlab DSP conference*, volume 99, pages 16–17, 1999.

[54] Ye Wei, Li Liu, Jun Zhong, Yonggang Lu, and Letian Sun. Unsupervised race walking recognition using smartphone accelerometers. In *International Conference on Knowledge Science, Engineering and Management*, pages 691–702. Springer, 2015.

[55] Jiang Zhu, Pang Wu, Xiao Wang, and Joy Zhang. Sensec: Mobile security through passive sensing. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 1128–1133. IEEE, 2013.

Appendix A

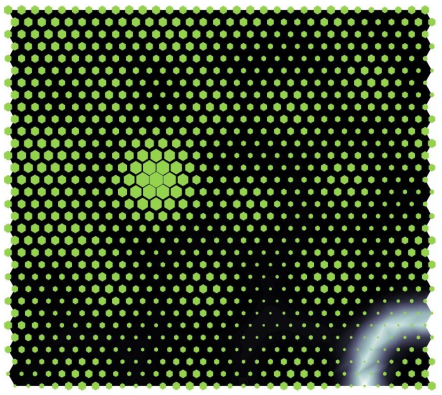
SOMs from other datasets

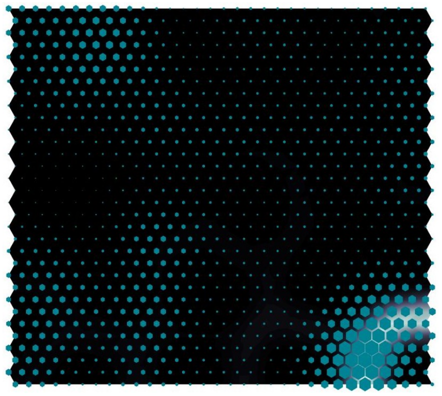
A.1 Rice LiveLab

Following are the visualizations (one per user) from evaluation of the proposed framework on the LiveLab dataset.

Discrete

In total, 529 SOMs have been obtained. For purposes of illustration, the most exclusive and least exclusive user models are shown below, with their corresponding outclass responses.


(a)


(b)

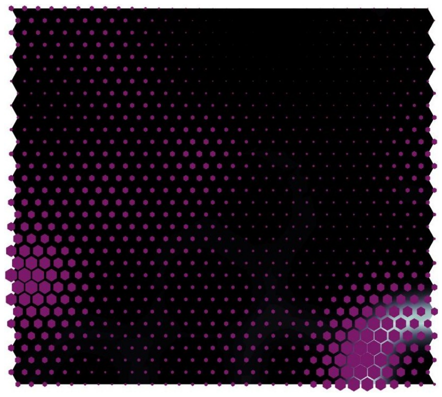
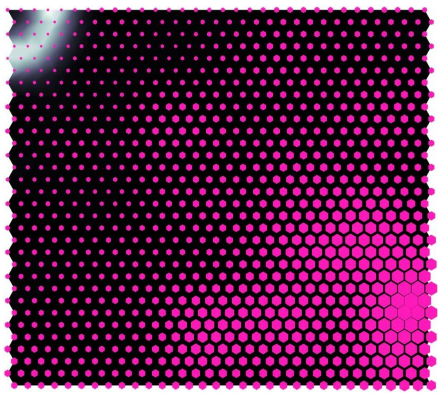
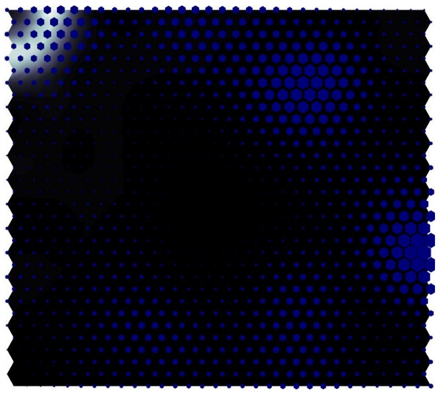

(c)

Figure A.1: Most exclusive user in LiveLab dataset (discrete): (a) inclass (b-c) sample outclass response


(a)


(b)

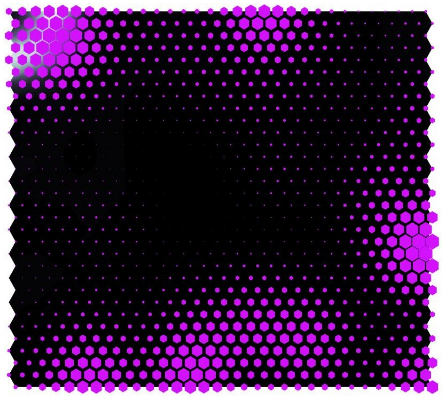
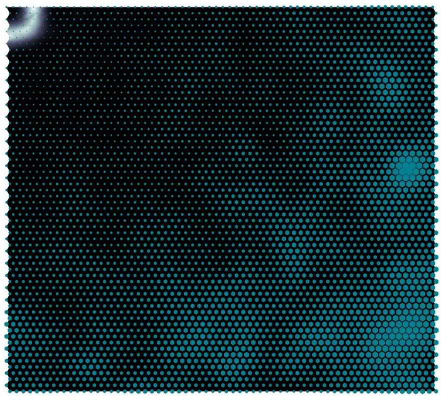
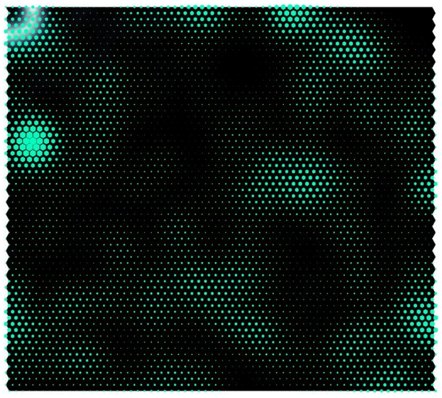

(c)

Figure A.2: Least exclusive user in LiveLab dataset (discrete): (a) inclass (b-c) sample outclass response

Continuous


(a)


(b)

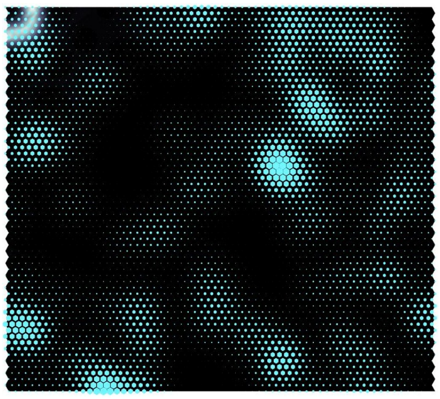
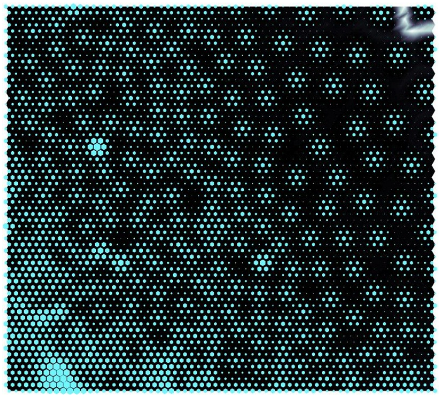
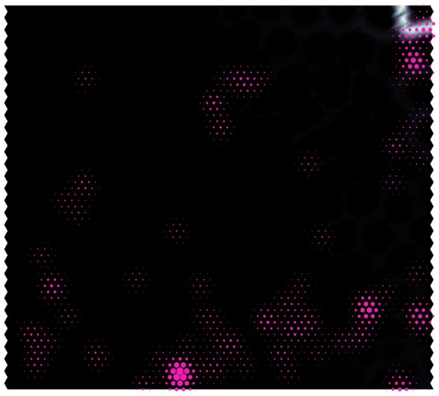

(c)

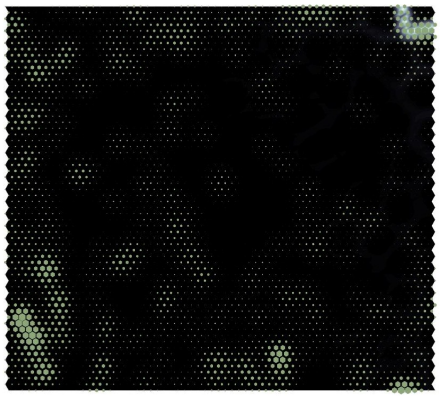
Figure A.3: Most exclusive user in LiveLab dataset (continuous): (a) inclass (b-c) sample outclass response



(a)



(b)

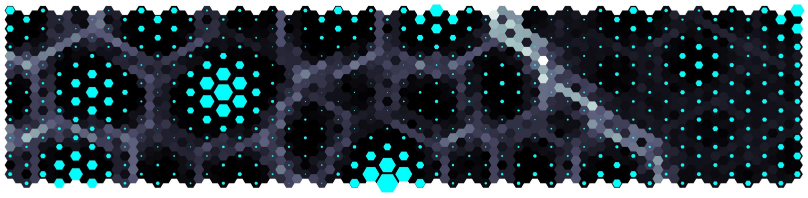


(c)

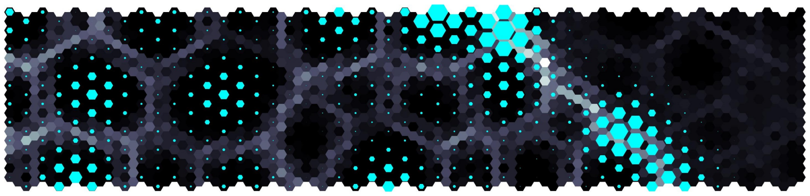
Figure A.4: Least exclusive user in LiveLab dataset (continuous): (a) inclass (b-c) sample outclass response

A.2 GCU version 1

Following are the visualizations (one per user) from evaluation of the proposed framework on the GCU version 1 dataset.

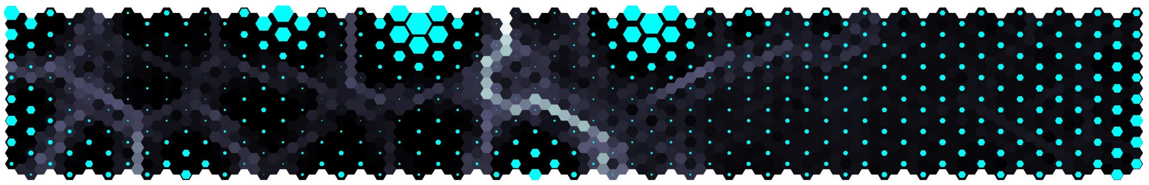


(a)

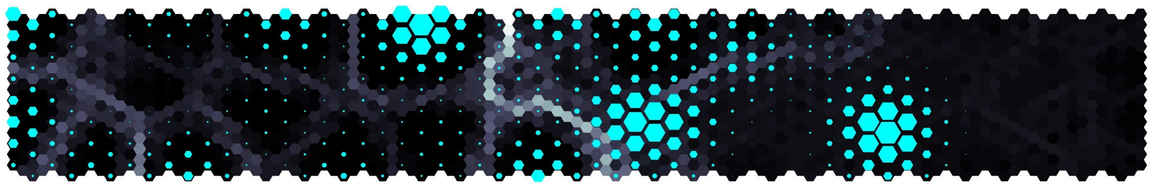


(b)

Figure A.5: U1: (a) inclass (b) outclass

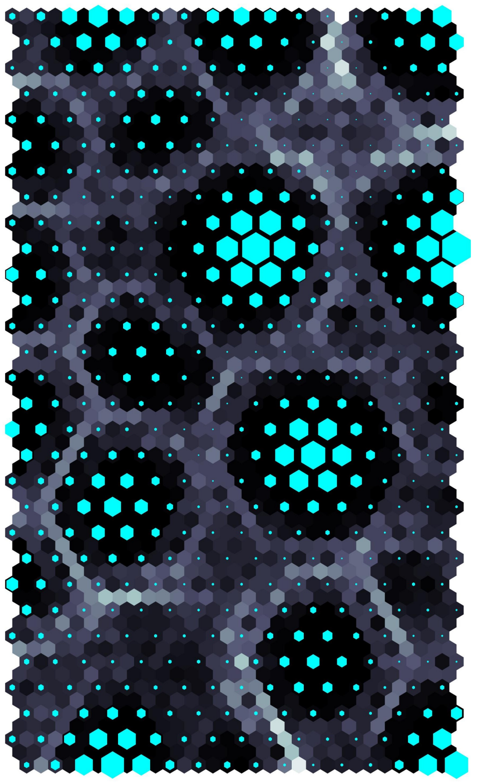
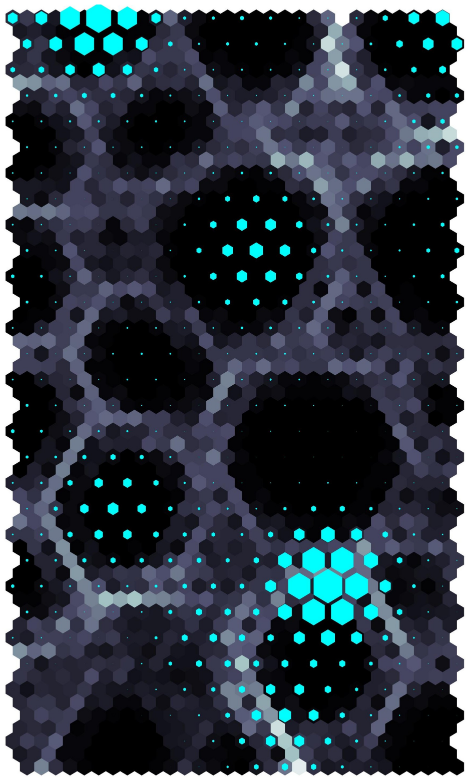


(a)



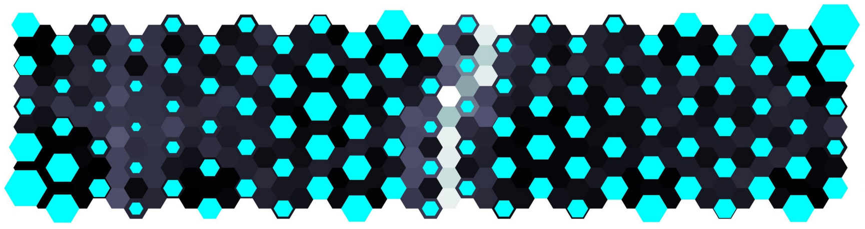
(b)

Figure A.6: U4: (a) inclass (b) outclass

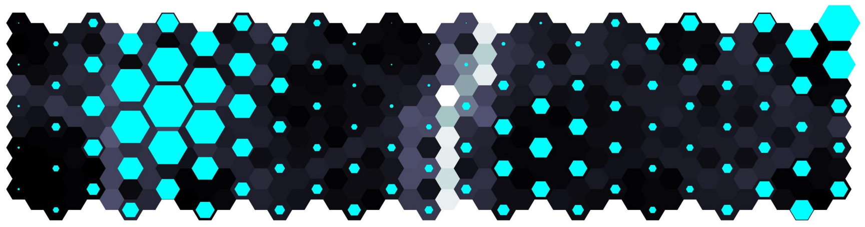



(a) (b)

Figure A.7: U2: (a) inclass (b) outclass

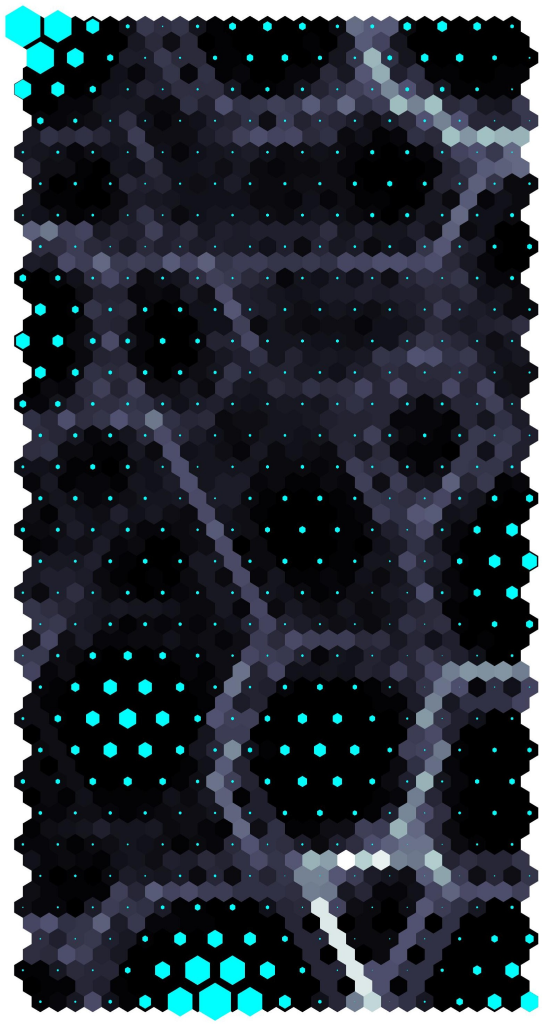


(a)



(b)

Figure A.8: U7: (a) inclass (b) outclass


(a)

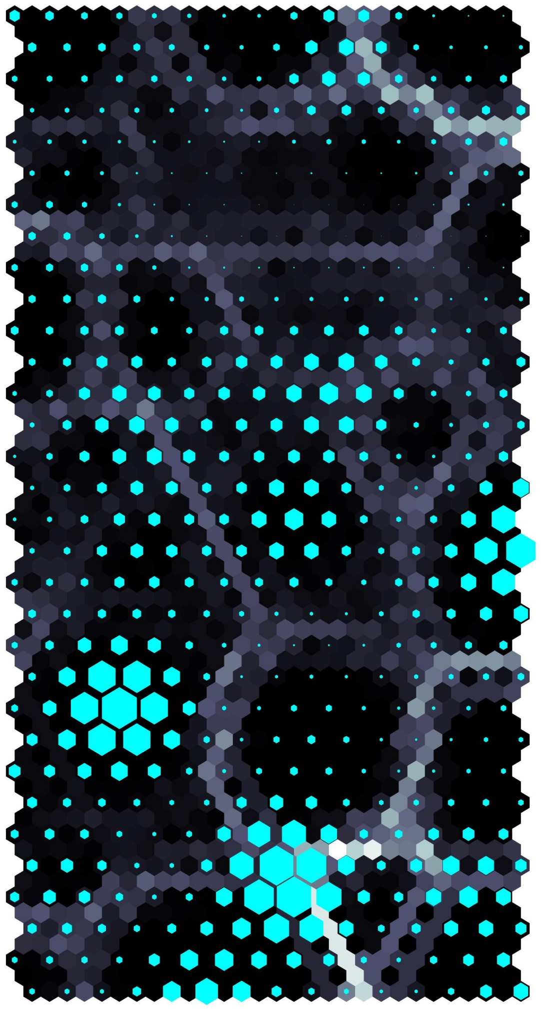
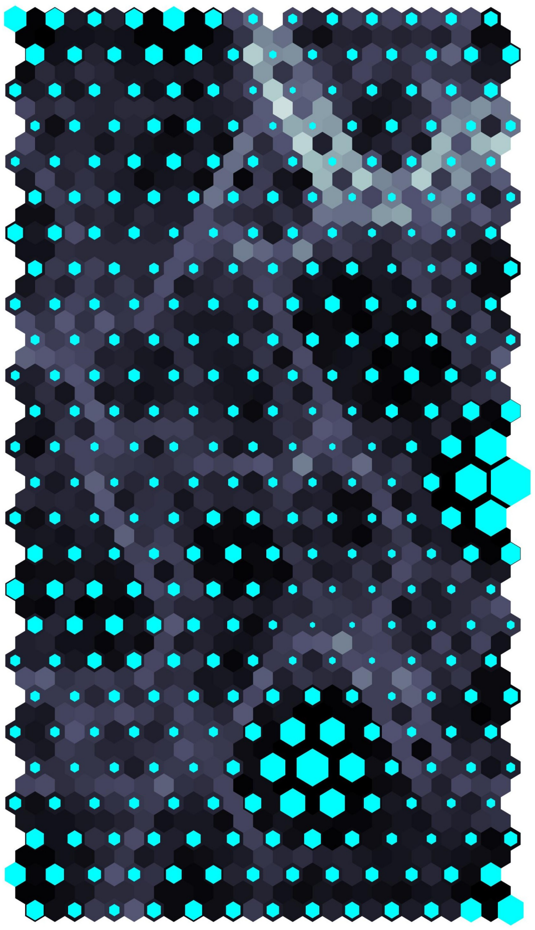

(b)

Figure A.9: U3: (a) inclass (b) outclass


(a)

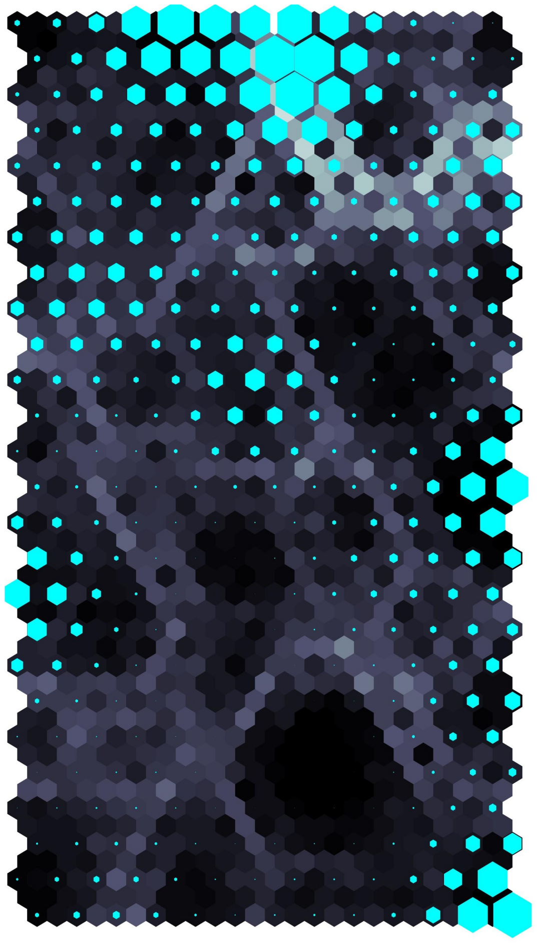
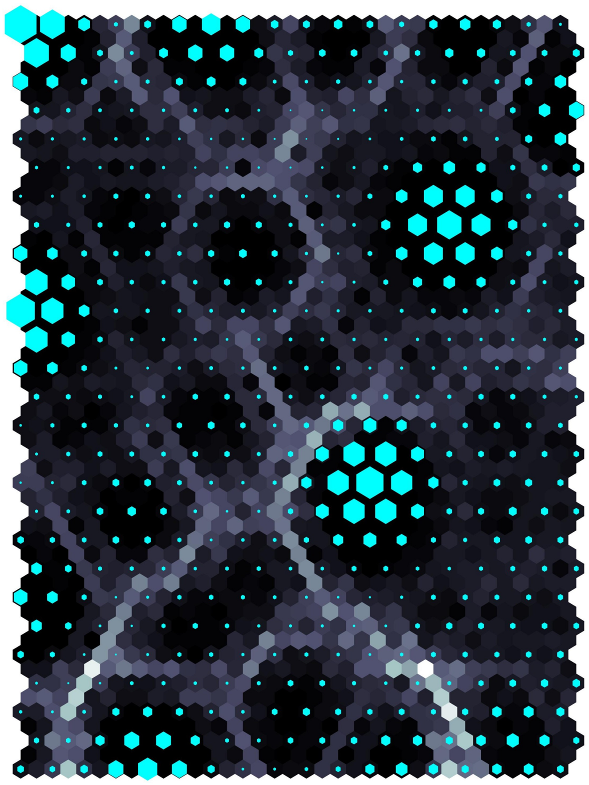

(b)

Figure A.10: U5: (a) inclass (b) outclass


(a)

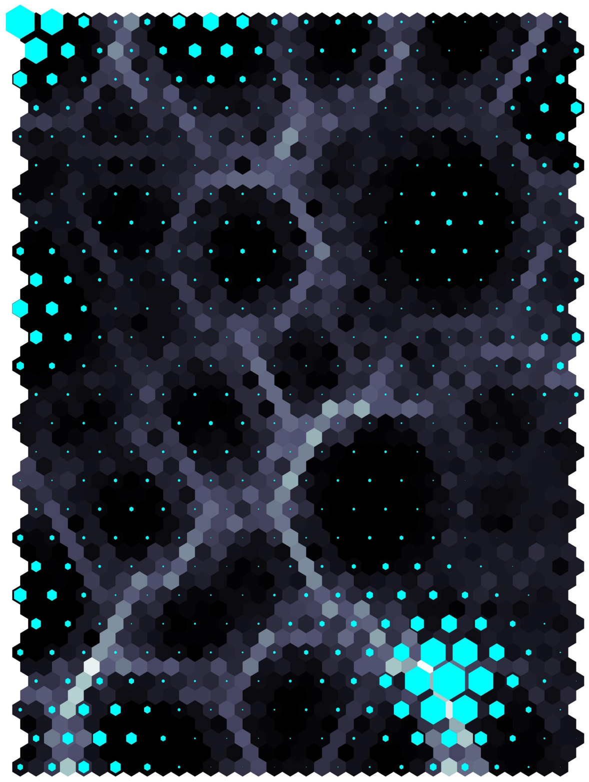

(b)

Figure A.11: U6: (a) inclass (b) outclass