Applying Machine Learning to enhance payments systems security

Mario Parreño Centeno

School of Computing Science
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

November 2020

I would like to dedicate this thesis to my loving parents

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Mario Parreño Centeno
November 2020

Acknowledgements

I want to thank many people I met during this marvellous trip to achieve my PhD degree, and without whom I would not have completed this thesis.

First, I want to express my immense gratitude to my supervisor Professor Aad van Moorsel. He has provided me with invaluable support, guidance, and crucial insights throughout these four years.

Further, I want to offer my special thanks to Dr Yu Guan for his technical support and suggestions. I also wish to thank Professor Rajiv Ranjan and Dr Stephen McGough for their comments and advice.

My gratitude extends to the Centre for Doctoral Training in Cloud Computing for Big Data. As a student in this centre, I have enjoyed working alongside numerous brilliant and inspirational people. I want to express my enormous gratitude to the centre's directors, Professor Paul Watson and Professor Darren Wilkinson. Furthermore, I wish to thank the following people for their support: Profesor Stefano Castruccio, Dr Mathew Forshaw, Dr Sarah Heaps, Barry Hodgson, Steve Caughey, Oonagh McGee, Jennifer Wood, and all the rest of the PhD students, especially to the first cohort.

My PhD has taken place at the Secure and Resilient Systems research group, where I met extraordinary friends and colleagues. Some of them are Maher Alharby, Luca Arnaboldi, Amjad Aldweesh, Uchechi Nwadike, Roberto Metere, Dr Mohammed Aamir, Dr Martin Emms and Dr Ioannis Sfyrakis.

Also, I want to express my sincere thanks to the students and staff I met at The Alan Turing Institute during my internship, which was a fantastic experience.

I also wish to acknowledge my internal examiner Jaume Bacardit and my external Noura Al-Moubayed for their relevant comments and feedback on this thesis.

Finally, I wish to mention my family and friends for their kindness, understanding and motivation. I want to express my sincere gratitude to Marino Parreno-Lujan, Sagrario Centeno-Jara, Gemma Parreno, Xavier Cardus and Emma Cardus. And an extraordinary mention of gratitude to Laura Sanchez for her help and invaluable support.

Abstract

During the last two decades, the economic losses because fraudulent card payment transactions have tripled. The significant percentage of losses is because of fraud on e-commerce transactions. Nowadays, there is a clear trend to use more and more mobile devices to make electronic purchases, and it is estimated that this trend will continue in the coming years.

In the card payment scheme, big financial institutions process millions of transactions every day; thus, they can model the processed transactions to predict fraud. On the other hand, merchants process a much lower number of transactions, but they have access to valuable information that they can collect from the devices that users utilise during the transaction.

In this thesis, we propose a series of measures to enhance the security of these two scenarios based on past transactional data and information collected from the users' device. Most of the approaches proposed so far to model processed transactions were based on supervised Machine Learning techniques. We propose a fraud detection system for card payments based on an unsupervised machine learning technique; thus, the system may be able to recognise new patterns of fraud.

On the other hand, we are looking far ahead, and because of the increment of use of mobile devices to conduct payments, we propose a series of measures to enhance the security of the mobile payment system. We have proposed a user identification and verification systems for smartphones. We base the identification and verification systems on motion data, so the systems will not require any explicit action from users.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the 1970s, a variety of schemes have been proposed to allow card payments. Typically, card payment has been categorised into card present and card, not present (CNP). In card present payment, the cardholder is physically present at the merchant store, and payment is performed by swiping (magnetic stripe), inserting (CHIP & PIN) or tapping (in case of contactless) a payment card to the merchant provider's point of sale (PoS) terminal/reader. On the other hand, CNP involves online, telephone and mail transactions where the cardholder is identified by the card details but is not physically present with the card. In the early years of the development of the payment technologies, card-present was the most popular transactions. However, with the arrival of the Internet, the use of Card Not Present transactions drastically increased [111]. We are using more and more electronic devices to carry on activities we were used to conduct in other ways. The United Kingdom had the fourth largest e-commerce market in the world, with a value of 78,903 million dollars [39]. It is estimated that in 2021 more than half of these purchases take place on mobile devices [177].

During the first years of the development of card payment transactions technology, fraudsters focused on making profit mainly with counterfeit and lost and stolen card frauds. However, with the introduction of CHIP & PIN technology in 2003, this landscape changed drastically. The introduction of this measure significantly enhanced the security of card-present payment, and fraudsters decided to focus their activity on the CNP scheme. Since then, economic losses from fraudulent CNP transactions have fluctuated because of the constant development of anti-fraud measures and campaigns, but nowadays, the economic losses because of fraudulent CNP payments in the UK are three times higher than 20 years ago (exclude the effect of inflation). Most of these losses are because of fraud in e-commerce.

Currently, in the UK, the two major contributions to fraud losses in card payments are the theft of personal information through social engineering and data breaches [64]. Our fast adoption of the use of new technologies in our daily routine helps fraudsters to steel information. The popularity of social media has opened other channels for fraudsters to contact and obtain information from their victims [155]. Furthermore, the increment of use of mobile devices makes it more difficult for users to spot fake sites, i.e., the size of the mobile devices screens is considerable smaller what hindered users' ability to account for anomalies [94]. On the other hand, data breaches occur more often, and their impact is larger. Nowadays, double data breaches have been reported in the United Kingdom than a decade ago (1244 in 2018 compared to 656 in 2008) [47].

To fight fraud in card payments, initially, financial institutions implemented some form of transaction classification systems, often a set of rules, which raised alerts on individual transactions, which are considered suspicious [44]. The first Fraud Detection Systems for card payments were based on rules, i.e., a set of thresholds established by experts trigger an alarm [73]. In 1993, a Canadian bank generated a report each night [68] with accounts that break specific rules, e.g., too many items with high value or too many purchases in a short period. Managers review the reports to flag potential fraud and contact the cardholders. Because the process was time-consuming and demanding, and in an attempt to make the rule-based systems more reliable and avoid the difficulty of setting up the rules, data mining techniques were introduced to set up the association [17].

Since the 90s, Machine Learning (ML) techniques were used to learn fraudulent patterns from past data. ML techniques in fraud detection systems operated on two different levels [166]: transaction level (individual or aggregation) and account level (behavioural models). In this thesis, we focus on enhancing the security of electronic payments systems. We propose approaches for both operating levels.

Financial institutions, such as banks, process millions of transactions every day. They can analyse the characteristics of past transactions to model normal and fraudulent payments. On the other hand, merchants will only process transactions from their customers, which is a much smaller number. However, they have access to other sources of data that they can use to model their clients' behaviours to build a fraud detection system. For example, modelling the web browsing user behaviour of customers when they make purchases online takes into account data sources such as cadence when clicking.

Because of the increment in the trend of using mobile devices for online purchases, we propose an authentication system for smartphones at the account level.

The three available factors for authentication are:

1. Something the user knows, e.g., a PIN
2. Something, the user, posses, e.g., card, smartphone
3. Something the user is, e.g., fingerprint

Traditionally, the pin has been the most popular method for authentication, although their weakness has been proven multiple times [179]. Some disadvantages of token-based methods are the discomfort for users to carry on the token and that it can be stolen [137].

With the incorporation of new sensors in mobile devices, biometric authentication methods such as fingerprint and Facial Recognition has become very popular in recent years [50]. However, these types of authentication systems require explicit action from users. We investigate authentication approaches based on motion data, i.e., accelerometer, gyroscope, and magnetometer data, which do not require any explicit action from the user. We propose different approaches for identification and verification based on Machine Learning techniques.

For specific scenarios, the merchant can be interested in identifying the user between a group of them, e.g., identifying the member of a family who is using a sharing tablet. We compare the performance of several supervised ML methods. On the other hand, we propose a verification approach that uses a deep learning technique for feature extraction and bases the classification phase in an unsupervised method.

Furthermore, we investigate the goodness of the approach for continuous authentication. Because biometrics methods such as face recognition require an explicit action from the user, they are not convenient for continuous authentication, i.e., the user experience will be deeply affected, continually asking for a new picture. However, since motion authentication does not require any explicit action from the user, it could be implemented as a continuous authentication system where the user is verified continuously without requiring any specific action from the user.

Moreover, EU Payment Services Directive [58] is arranged to make essential changes to the payment industry by March 2021. From that date, electronic payments (except for some exemptions) must be authenticated with multi-factor authentication. The new directive

called Strong Customer Authentication will require at least two independent factors in the authentication process; thus, our approach could support this new directive.

On the other hand, we have proposed several approaches that operate at the transaction level. Many of the approaches that model the characteristics of individual transactions proposed so far are supervised detection systems. The models have been trained to learn the patterns of normal and fraudulent transactions. This approach presents the main disadvantage, i.e., the system may only detect known fraudulent patterns. If the model is trained to recognise specific patterns of fraud, it may not recognise new fraudulent patterns when they occur. Thus, we have investigated the performance of several card payment fraud detection systems based on unsupervised Machine Learning methods. Moreover, we have investigated cost-sensitive models. Many of the models are trained to minimise the number of false positives and false negatives. We have investigated if models can be more effective in minimising financial losses.

1.1 Research Problem

To enhance the security of electronic payments systems, we investigate the following research problems:

Card payment fraud. Despite the many different measures to avoid fraud proposed since the inception of the card payments technology, losses because the fraudulent activity has tripled in the last 20 years. It is essential to understand the current card payment system and fraud related to them to propose efficient fight fraud measures.

Supervised identification of smartphone users. It is common to share mobile devices in a group of people between the members of a family. Here, it will be useful to identify different users within a group to grant them different privileges.

Most of the identification approaches proposed so far require explicit action from users producing less user-friendly systems.

On the other hand, it will be desirable that the system identify the user during the whole session instead of just one time at the beginning of the session.

Unsupervised verification of smartphone users. In specific scenarios, merchants will not be interested in identifying a user within a group. They will only be concerned

with verifying the user's identity utilising the device to realise the transaction, i.e., the smartphone.

Unsupervised fraud detection systems for electronic payments. Financial institutions such as banks have access to millions of transactions performed by customers. Machine learning techniques have been successful in modelling this vast amount of information to detect fraud. However, many of the approaches proposed so far are supervised approaches, limiting the system's ability to recognise new fraudulent patterns.

On the other hand, usually, these types of systems are required to detect the highest number of fraudulent transactions as possible. However, the main goal of financial institutions is to increase their benefits.

1.2 Contributions

The work presented in this thesis makes several key contributions:

(i) It provides a comprehensive survey of the economics of payment systems' frauds and identifies different types of frauds in card payment and mobile payment.

(ii) Proposal and evaluation of and smartphone user identification system based on sensor data and supervised Machine Learning techniques. Merchants can collect a large amount of information during the payment process, not necessarily related to the transaction. It can be information related to the device from the customer initiating the transaction. Nowadays, mobile devices process most of the payment transactions. Many of these devices, such as smartphones and tablets, include motion sensors, i.g. accelerometers. We could use them to authenticate users based on their movements.

(iii) Introduce a novel verification system for smartphone users based on unsupervised Machine Learning techniques. We have investigated unsupervised verification approaches for smartphones, i.e., the system learns the owner's patterns and calculates the probability that new samples belong to the same distribution.

(iv) Introduce an We demonstrate that we can increase the approach's accuracy by transforming the raw data into a more meaningful representation.

(v) Introduce an innovative fraud detection system for electronic card payment transactions based on unsupervised Machine Learning techniques. We have investigated the development of unsupervised fraud detection systems for card payments to detect new fraudulent patterns.

Moreover, we have investigated cost-sensitive models. Many of the models are trained to minimise the number of false positives and false negatives. We have investigated if models can be more effective in reducing financial losses.

1.3 Thesis Structure

This thesis is organised as follows.

Chapter 1 describes the motivations behind the work carried out as part of this thesis and highlights the main contributions of the research. We describe the related peer-reviewed publications produced throughout my PhD.

Chapter 2 introduces and provides context for the Machine Learning models we used in the research. The chapter is divided into two main sections, on supervised and unsupervised techniques, respectively.

Chapter 3 investigates credit card payment technologies, fraud for each method of card payments, attack mechanisms and economics of fraud. To propose efficient defences for card payment systems, first, we have to know the weakness and strengths of the current systems.

Chapter 4 proposes a motion-based identification approach for mobile devices based on supervised machine learning techniques. The experiments are based on motion sensor data, i.e., accelerometer, gyroscope and magnetometer data.

Chapter 5 proposes a motion-based verification approach for smartphones based on unsupervised machine learning techniques. Similar to Chapter 4, the experiments are based on motion data. We investigate the efficiency of different approaches, as well as feature importance and the re-authentication time.

Chapter 6 proposes an unsupervised fraud detection system for card payment transactions comparing several machine learning and deep learning models. It investigates transaction feature selection and cost-sensitive models.

Chapter 7 summarises the conclusions of the work presented in this thesis and motivates future directions for work in the area.

1.4 Related Publications

During my PhD. I have contributed to the following peer-reviewed publications:

> Mohammed Aamir Ali, Muhammad Ajmal Azad, Mario Parreno Centeno, Feng Hao, Aad van Moorsel. Consumer-facing technology fraud: Economics, attack methods and potential solutions, Future Generation Computer Systems, Volume 100, 2019, Pages 408-427, ISSN 0167-739X,

This paper investigates fraud over the Internet, which has increased dramatically. It presents the anatomy of frauds for different consumer-facing technologies from three broad perspectives — it discusses the Internet, mobile, and traditional telecommunication, from the perspective of losses through frauds over the technology, fraud attack mechanisms and systems used for detecting and preventing frauds. The paper also provides recommendations for securing emerging technologies from fraud and attacks. My contributions to this paper were section 4 integrally and figure 2 in section 3. These contributions to this paper form the basis of Chapter 3 of this Thesis.

> Mario Parreño Centeno, Yu Guan and Aad van Moorsel. Mobile Based Continuous Authentication Using Deep Features. In Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning (EMDL'18). 2018. ACM, New York, NY, USA, 19-24. DOI: https://doi.org/10.1145/3212725.3212732

This paper investigates continuous authentication approaches to authenticate smartphone users during a work session, i.e., for mobile banking applications. The approach is based on motion patterns. We also investigate how the authentication accuracy is affected by the sampling frequency and the re-authentication time. This paper forms the basis of Chapters 4 and 5.

> Mario Parreño Centeno, Mohammed Aamir Ali, Yu Guan and Aad van Moorsel. Unsupervised Machine Learning approaches for card payment fraud detection. In Proceedings of the 14th International Conference on Risks and Security of Internet and Systems (CRISIS'19). 2019.

This paper investigates the current fraud detection systems for card payments and proposes an unsupervised approach. This paper forms the basis of Chapter 6.

Chapter 2

Background–Machine Learning techniques

2.1 Introduction

Machine Learning (ML) is in the area of artificial intelligence (AI), and it gives computers the ability to make predictions or decisions without being explicitly programmed for it. Their criteria are based on algorithms that are built mathematically modelling past data.

Today, machine learning is so popular that one can use dozens of times a day without realising it. In the last few years, machine learning has contributed enormously to developing a massive variety of solutions in driverless vehicles, speech recognition systems, effective web search engines, and knowledge of the human genome.

Machine Learning problems are classified into these three main categories: unsupervised, supervised, and reinforcement learning.

Supervised learning is meant to find patterns in the data corresponding to a label that defines the meaning of each sample. For example, there could be millions of pictures of animals and include an explanation of what each animal is. Here, we could create a machine learning application that distinguishes an animal from another.

In unsupervised learning, there are no labels associated with the input data. It can be used to find commons patterns on a group of samples.

Reinforcement learning computer programs interact with an environment to learn a task, i.e., playing a game. As it navigates into the problem, the environment provided feedback in the form of reward that the algorithm tries to maximise.

In this thesis, we investigate the application of supervised and unsupervised algorithms.

2.2 Supervised Machine Learning methods

Supervised learning techniques use labels to identify patterns in the input data for prediction.

Predictive modelling can be divided into classification and regression problems. In classification modeling an approximation function maps the input variables (independent variables) to a discrete output variable (the dependent variable DV).

We can describe variables as categorical, ordinal, or numerical. Categorical variables have two or more categories in intrinsic ordering, i.e., eye colour. Ordinal variables are similar to categorical, but they can be ordered, i.e., high and low. Finally, numerical variables are identical to ordinal variables except that they are numerical, i.e., £10,£20, and £30.

On the other hand, in regression modelling, the output variable is continuous.

In this section, we are going to introduce the different classification techniques which we will use through this work. For it, we are going to use the following notation:

- $\mathbf{x}^{(i)}$ denotes a vector of input variables with $i = 1,...,m$ where m equal to the number of training samples.
- $x_j^{(i)}$ denotes a element of the vector of input variables with $i = 1,...,m$ where m equal to the number of training samples and with $j = 1,...,n$ where n equal to the number of features in the input vector.
- $y^{(i)}$ denotes the target variable to be predicted with $i = 1,...,m$.
- the tuple $(\mathbf{x}^{(i)}, y^{(i)})$ denotes a training sample with $i = 1,...,m$.
- a list of m training samples $\{(\mathbf{x}^{(i)}, y^{(i)})\}$ where $i = 1,...,m$ denotes the training dataset.
- $X \varepsilon \mathbb{R}^{rxc}$ is matrix of r rows and c columns. Each element can be indexed by a $x_{ij} | 0 \leq i \leq r$ and $0 \leq j \leq c$.
- $\mathscr{X}$ denotes the space of input values.

– $\mathscr{Y}$ denotes the space of output values.

So, giving a training dataset $\{(x^{(i)}, y^{(i)})\}$, the machine learning model pretends to learn a function $h : \mathscr{X} \mapsto \mathscr{Y}$,where $h(x)$ should be a good predictor of y. Probabilistic classifiers infer a probability distribution over a set of classes, while non-probabilistic classifiers infer the most likely class for a given sample. Logistic Regression, Neural Networks, Random Forest and k-Nearest Neighbours are examples of probabilistic classifiers and Support Vector Machines of non-probabilistic. Some probabilistic techniques such as Logistic Regression and Neural Network, provide a functional form f and parameter vector α to express $P(y|x)$ as:

$$P(y|x) = f(x, \alpha)$$

The parameters α are determined based on the training data [49].

k-Nearest Neighbours, determine the value of $P(y|x)$ as the ratio samples of the class y among the k nearest neighbors of x.

2.2.1 Logistic regression

Logistic Regression (also known in the literature as logit regression, maximum-entropy classification, or the log-linear classifier) is a generalised linear model for classification where the DV is categorical. In the case of a binary DV, the output can take only two values, 0 and 1. Some of the assumptions of the LR model are:

– The dependent variable should be dichotomous in nature (e.g., presence vs absent).

– There should be no outliers in the data.

– There should be no high correlations among the predictors.

– We Assume that the m training examples were generated independently.

We can interpret the parameters of the model, coefficients and intercept, to understand the relative importance of the input features.

To show the mathematical definition of the LR model, first, we are going to introduce the linear regression model, which has many similarities to it.

Linear regression is a linear approach for modelling the relationship between a scalar DV y and one or more explanatory variables. In linear regression, the relationships are modelled

using linear predictor functions whose unknown model parameters are estimated from past data. The linear regression model approximates y with the following lineal function of x:

$$h_\theta = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta_0 \sum_{i=1}^{n} \theta_i x_i,$$

where the constant θ_0 (by setting $x_0 = 1$) is call the intercept term and the n parameters θ are call the weights. The weights parametrise the space of linear functions mapping from $\mathscr{X}$ to $\mathscr{Y}$. This expression is called the decision function and can be rewritten in vector form as:

$$h_\theta = \theta^T x \tag{2.1}$$

where θ^T is a matrix with $\theta^T \varepsilon \mathbb{R}^{(n+1)x1}$ and x is a vector of $(n+1)$ elements.

In LR, we still want to solve a linear combination of features and weights, but we map this linear combination to a binary output , $y \varepsilon \{0,1\}$. A simple way to force the output of the expression 2.1 to take a value between 0 and 1 is changing the model h_θ to:

$$h_\theta = g(\theta^T x) = \frac{1}{1+\mathrm{e}^{-\theta^T x}}\,, \tag{2.2}$$

where:

$$g(z) = \frac{1}{1+\mathrm{e}^{-z}} \tag{2.3}$$

$g(z)$ is called the logistic or sigmoid function and its graph is shown in Figure 2.1

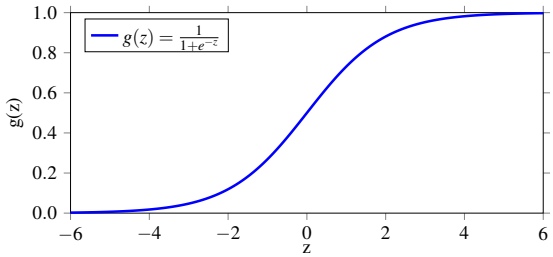


Figure 2.1 Graph of the sigmoid function.

Notice that $g(z)$, and hence also h_θ, are always bounded between 0 and 1.

Notice that $g(z)$, and hence also h_θ, are always bounded between 0 and 1.

Given the LR hypothesis in 2.2 we need to fit the parameters θ. As we told before, LR learns the parameters θ from past data. Given a training dataset consisting of a list of m training samples $\{(x_{(i)}, y_{(i)})\}$, we want to make h_θ close to y. Newtons Method can be used to solve Logistic Regression. Logistic Regression uses the concept of Log-Likelihood. We use the hypothesis function h_θ, and formulate the likelihood function to maximise and fit the weights. We can fit the parameters via maximum likelihood under a set of probabilistic assumptions. Let us assume that the output of the LR hypothesis function is the estimated

probability that $y = 1$ for a given sample x parametrised by θ:

$$P(y = 1|x;\theta) = h_\theta(x)$$

Thus, because this is a binary classification model, the estimated probability that $y = 0$ for a given sample x is:

$$P(y = 0|x;\theta) = 1 - h_\theta(x).$$

Then, the probability of y given x is:

$$p(y|x;\theta) = (h_\theta(x))^y(1 - h_\theta(x))^{1-y}$$

So, in LR we define a hypothesis function h_θ such as:

$$h_\theta = g(\theta^T x) \begin{cases} 1 & \text{if } \theta^T x \geq 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Where $\theta^T x = 0$ defines the decision boundary, which divides the data space into two areas and depending on the position respect to the decision boundary the sample $x^{(i)}$ will be classified belonging to any of the two classes $y = 1$ or $y = 0$. Fig. shows a graphical representation of the decision boundary for a scenario with two θ parameters.

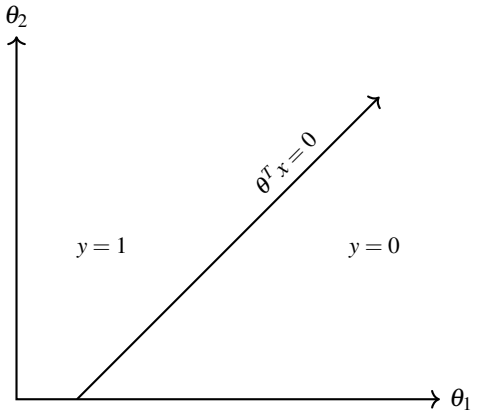


Figure 2.2 LR decision boundary for a case where the hypothesis has to θ parameters.

Because logistic regression predicts probabilities, rather than just classes, we can fit it using likelihood. We write the likelihood of the parameters as:

$$L(\theta) = p(y|x;\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}}(1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

However, taking the *log* of the expression make the problem more tractable (because we multiply m likelihoods together, all of them less than 1, the program may run out of precision. On the other hand, taking the log we guarantee that the objective function is strictly concave,

meaning there is just one local maximum). This new expression is known as the log-likelihood of the hypothesis function:

$$\mathscr{L}(\theta) = \log L(\theta) = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \tag{2.5}$$

To find the θ's parameters that define the decision boundary, we want to maximise the log-likelihood $\mathscr{L}$ of the hypotheses $h_\theta(x)$. There are many methods to solve optimisation problems (minimising or maximising some function $f(x)$) such as gradient descent, which we will introduce in 2.2.2. In this section, we are going to use Newton's method to find $w*$, which are the values for the vector w, which maximise our objective function.

Newton's method uses a second-order Taylor series expansion to approximate $f(x)$ near some point $x_{(0)}$ [74]:

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^T \Delta_x f(x^{(0)} + \frac{1}{2})(x - x^{(0)})^T H(f)(x^{(0)})(x - x^{(0)})$$

And solving for the critical point of the function, we obtain:

$$x^* = x^{(0)} - H(f)(x^{(0)})^{-1} \Delta_x f(x^{(0)}) \tag{2.6}$$

Substituting in 2.6 our objective function $\mathscr{L}(\theta)$, we can use the same algorithm to update each θ_n, and we obtain the update rule:

$$\theta_n := \theta_n - H^{-1}_{\mathscr{L}(\theta)} \nabla_\theta \mathscr{L}(\theta) \tag{2.7}$$

where $\nabla_\theta \mathscr{L}(\theta)$ is the vector of partial derivatives of $\mathscr{L}(\theta)$ with respect to the θ_i (it is call the gradient) and $H_\theta(\mathscr{L}(\theta))$ is an $n+1$ by $n+1$ matrix (assuming that it includes the intercept term) called the Hessian matrix, whose entries are the second derivatives of our objective function (note if a function has n input dimensions, there are n^2 second derivatives), which can be represented in a matrix with entries:

$$H_{\mathscr{L}(\theta)ij} = \frac{\delta_2 \mathscr{L}(\theta)}{\delta\theta_i \delta\theta_j}$$

So, applying Newton's methods to maxime the log-likelihood $\mathscr{L}(\theta)$ of our hypothesis function $h_\theta(x)$ consists in applying the follow steps:

1. Calculate the gradient of $\mathscr{L}(\theta)$
2. Calculate the Hessian of $\mathscr{L}(\theta)$
3. Update the values of the vector θ using 2.7
4. Repeat steps 1,2 and 3, until the result converges

When the $\mathscr{L}(\theta)$ is a positive definite quadratic function, the Newton method will find the maximum directly, so it will find w^* applying equation 2.7 once. When $\mathscr{L}(\theta)$ is not truly quadratic but can be locally approximated as a positive definite quadratic, we will find the solution throw the iterative algorithm much faster than gradient descent would. Although one iteration of Newton's method can be more computational expensive since it requires to calculate the Hessian, as long as n is not too large, it is much faster overall.

Regularisation

To improve the generalisation of our model, i.e., the performance on unseen data, we can use regularisation. Instinctively, It can be seen as a penalty to prevent that the model picks up the peculiarities or noise.

Technically, adding regularisation to the model means adding bias if the model suffers from high variance, i.e., avoiding overfitting. However, high bias will lead to underfitting, i.e., low performance for the training and testing datasets.

Variance is the variability of the model prediction, which shows the spread of the data. Model with high variance performance very well on the training data but does not generalise well on unseen data.

On the other hand, bias is the difference between the mean prediction of the model and the true label. Models with high bias do not pay attention to the training data and oversimplify the resulting model. It leads to a high error in training and test data.

Underfitting in supervised learning happens when the model is not able to capture the underlying patterns of the data. In this case, usually, the model has high bias and low variance. It occurs when we do not have enough data to train the model or when we try to train a linear model with nonlinear data. On the other hand, overfitting occurs when the model captures the noise along with the underlying patterns in the data. It usually happens when the training data is very noisy. As a result, these models have low bias and high variance.

To build a proper model, we need to balance the bias and variance, i.e., to minimises the total error, to generalise well and to prevent overfitting. regularisation can help with it.

Our goal in an unregularised model is to minimise the cost function, i.e., we want to find the feature weights that correspond to the global cost minimum (note that the logistic cost function is convex).

In equation 2.5 we have written the expression of the log-likelihood. Instead of minimising this cost function, in regularised logistic regression, we minimise the following cost function:

$$\mathscr{L}(\theta) = \sum_{i=1}^{m} [y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))] + \lambda R(\theta) \tag{2.8}$$

This approach penalises high coefficients by adding a regularisation term $R(\theta)$ multiplied by a parameter λ. We penalise high coefficients because LR assigns a high weight if a feature appears only in one class.

The two most common regularisation terms to penalise high coefficients are the l1 norm or the square of the norm l2 multiplied by. Thus, they are called L1 and L2 regularisation:

The l1 norm term:

$$R(\theta) = ||\theta||^1 = \sum_{i=0}^{n} |\theta_i|, \tag{2.9}$$

and the l2 regularisation term is defined as:

$$R(\theta) = \frac{1}{2}||\theta||_2^2 = \frac{1}{2}\sum_{i=0}^{n} \theta_i^2. \tag{2.10}$$

The parameter λ controls the effect of the regularisation term. Higher values of λ lead to smaller coefficients, but when λ is very high can lead to underfitting.

2.2.2 Multi Layer Perceptron

A multilayer perceptron (MLP) is a feedforward artificial neural network. The term 'neural network' was proposed when looking for mathematical representations of biological systems [40, 157].

The perceptron

Given a list of training samples $\{x^{(i)}, y^{(i)}\}$, the Perceptron Learning Algorithm tries to find a hypothesis function h that predicts the label $y^{(i)}$ of every $x^{(i)}$ correctly.

Figure 2.3 is the graphical representation of the simplest feedforward network, including only one neuron (the perceptron). In this example, the neuron has n inputs for each feature value of the input sample and one more input for the bias, which gives flexibility to the model.

Each of the connections between the input units and the neuron has associated a weight (w or b depending on the kind of input unit).

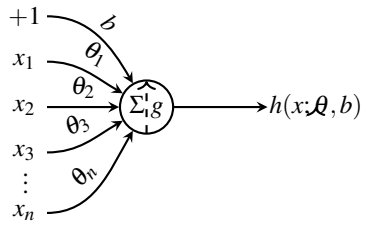


Figure 2.3 The simplest feedforward network including only one neuron.

The output of the neuron, call the decision function (the hypothesis function), is the summation of each input multiply by the weight associated to the connection.

$$h(x;\theta,b) = \theta_1 x_1 + ... + \theta_n x_n + b \tag{2.11}$$

where n is equal to the number of inputs parameters of the model and b the intercept which we called bias. Differently from 2.2, we use this notation where we treat the intercept term b separately and do not treat it such as extra parameter in the feature vector. 2.11 can also be written in vectorial form as:

$$h(x;\theta,b) = \theta^T x + b \ .$$

However, because the output could take any real value, we map it throw another function:

$$h(x;\theta,b) = g(\theta^T x + b) \ .$$

where $g(z)$ is called the activation function. Mapping the output helps to prevent the problem of vanishing/exploding gradients when training the model. So, when a neural network includes just one neuron and a sigmoid activation function 2.3, it will show the same results than a Logistic regression model shown in 2.2.

Here, as we see for the LR model, we could find the parameters θ from past data using Newton's method. However, in Neural Networks is common to use the stochastic gradient descent algorithm. Although, as we saw, Newton's method in the LR model has been proven to converge faster than gradient descent, there are two main reasons to use gradient descent to optimise the objective function in neural networks:

- Newton's method assumes convexity, and large data spaces are very much non-convex. The ratio of saddle points increases exponentially with the dimensionality of the feature space. Newton's method does not deal properly with saddle points. "While gradient descent dynamics are repelled away from a saddle point to a lower error by following

directions of the negative curvature, saddle-points instead become attractive under the Newton dynamics".

- The application of Newton's method for training large neural networks impose a significant computational burden. While gradient descent maximises the objective function using its derivative, Newton's method uses knowledge of its second derivative. The later can be faster when the second derivative is known and easy to compute. However, the analytic expression for the second derivative is often complicated or intractable when the number of parameters grown.

In the 2.2.1, we used a probabilistic approach to define the cost function of our hypotheses, i.e., maximise the log-likelihood of the data under the probabilistic model. As we saw, it was relevant to be able to apply the Newton's method because the log-likelihood is a convex function. In this function, we are going to define a different cost function which may be not convex. For each value of θ's we measure how close the $h(x^{(i)})$ are to the corresponding label $y^{(i)}$:

$$J(\theta) = \sum_{i=1}^{m} (h(x^{(i)}; \theta, b) - y^{(i)})^2 \tag{2.12}$$

We iterate through the list of training samples to minimise the above function and update the parameters θ and b in the direction of minimising each of the term of the objective function. So, we update the parameters in the following manner:

$$\theta_n := \theta_n - \alpha \Delta \theta_n \tag{2.13}$$

$$b := b - \alpha \Delta b \tag{2.14}$$

where α is a small non-negative scalar called the learning rate. The update is related to the value of α i.e. large value of α will lead large updates. This algorithm is known as the gradient descent.

Note that we will update all the model parameters, θ's and b, simultaneously.

To find the optimal values of the parameters α and b which better approximate h to y, we calculate the partial derivative of the cost function respect to each of the parameters which gives us a descent direction and it is known as the gradient. Because, the objective function is composed of functions of functions, we use the the chain rule to compute the derivatives:

$$\frac{\delta g}{\delta x} = \frac{\delta g}{\delta z} \frac{\delta z}{\delta x}$$

Therefore, at example $x_{(i)}$, we can compute the partial derivative of θ as:

$$\begin{aligned}\Delta\theta_n &= \frac{\delta}{\delta\theta_n}(h(x^{(i)};\theta,b)-y^{(i)})^2 \\ &= 2(h(x^{(i)};\theta,b)-y^{(i)})\frac{\delta}{\delta\theta_n}h(x^{(i)};\theta,b) \\ &= 2(g(\theta^T x^{(i)}+b)-y^{(i)})\frac{\delta}{\delta\theta_n}g(\theta^T x^{(i)}+b)\end{aligned} \tag{2.15}$$

Applying the change rule and noting that $\frac{\delta g}{\delta z} = [1-g(z)]g(z)$, we have:

$$\begin{aligned}\frac{\delta}{\delta\theta_n}g(\theta^T x^{(i)}+b) &= \frac{\delta g(\theta^T x^{(i)}+b)}{\delta(\theta^T x^{(i)}+b)}\frac{\delta(\theta^T x^{(i)}+b)}{\delta\theta_1} \\ &= [1-g(\theta^T x^{(i)}+b)]g(\theta^T x^{(i)}+b)\frac{\delta(\theta_1 x_1^{(i)}+...+\theta_n x_n^{(i)}+b}{\delta\theta_n}) \\ &= [1-g(\theta^T x^{(i)}+b)]g(\theta^T x^{(i)}+b)x_n^{(i)}\end{aligned} \tag{2.16}$$

And substituting 2.16 in 2.15 leads to:

$$\Delta\theta_n = 2[g(\theta^T x^{(i)}+b)-y^{(i)}][1-g(\theta^T x^{(i)}+b)]g(\theta^T x^{(i)}+b)x_n^{(i)} \tag{2.17}$$

where:

$$g(\theta^T x^{(i)}+b) = \frac{1}{1+\exp(-\theta^T x^{(i)}-b)}$$

And similar derivations lead to:

$$\Delta b = 2[g(theta^T x^(i)+b)-y^i][1-g(\theta^T x^i+b)]g(\theta^T x^{(i)}+b) \tag{2.18}$$

Here, the stochastic gradient descent algorithm to learn the decision function $h(x;\theta,b)$ is:

1. Initialise the parameters θ and b randomly.
2. Pick a random example $\{x^{(i)},y^{(i)}\}$.
3. Compute the partial derivatives of each parameter θ_n and b by equations 2.17 and 2.18.
4. Update the parameters using equations 2.13 and 2.14, then back to step two.

The process is repeated until converges the result or the number of iterations exceeds a specific value.

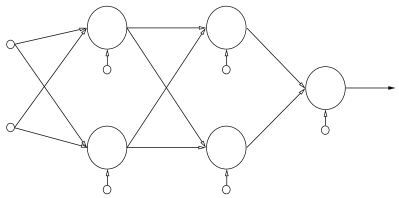


Figure 2.4 A feedforward network including three layers. The first and second layers consist in two neurons each; the third layer includes one neuron.

Regularisation

In 2.12 we have define the cost function of the perceptron.

$$J(\theta) = \sum_{i=1}^{m} (h(x^{(i)}; \theta, b) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \tag{2.19}$$

We iterate through the list of training samples to minimise the above function and update the parameters θ and b in the direction of minimising each of the term of the objective function. So, we update the parameters in the following manner:

Multi Layer Perceptron

However, using only one neuron as in the sample study so far, we can separate the data space in two areas by a linear boundary [55]. The Multi-Layer Perceptron (MLP) architecture consists of perceptrons' layers only with forward connections to successive layers. Thus, we separate the data spaces in several areas and split the data space with non-linear boundaries . It is shown in fig. 2.4. In this new neural network architecture, including several neurons, also we can find the θ parameters using the gradient descent algorithm, we need to calculate the partial derivative respect each parameter of the objective function.

Because each parameter may affect the objective functions from different ways we can use the backpropagation algorithm to calculate the derivatives of each parameter. The backpropagation algorithm is an implementation of the chain rule specifically designed for neural networks. To make reference to each of the $theta$ parameters, from now, we will use the notation $\theta_{ij}^{(l)}$ which means $theta$ parameter at the l^{th} layer connecting neuron (or input) j^{th} to the neuron i^{th} in layer $l+1^{th}$. And we are going to refer to the b parameters such as b_i^l which means the bias of neuron i. The layers are indexed from 1 for the input layer to L to the last layer. And the number of neurons in the layer l is equal to s_l. Using this new notation, we can show the recursive computation of $h(x; \theta, b)$ as:

$$h^{(1)} = x$$
$$h^{(2)} = g((\theta^{(1)})^T h^{(1)} + b^{(1)})$$

...

$$h^{(L-1)} = g((\theta^{L-2})^T h^{(L-2)} + b^{(L-2)})$$

$$h(x) = H_{(L)} = g((\theta^{(L-1)})^T h^{(L-1)} + b^{(L-1)})$$

And we can specify the steps of the backpropagation algorithm as:

1. Compute $h_{(1)}$ to $h_{(L)}$.
2. In the output layer compute the gradient as:
$$\delta_1^{(L)} = 2(h^{(L)} - y)g'(sum_{j=1}^{s_L-1}\theta_{1j}^{(L-1)}h_j^{L-1} + b_1^{(L-1)})$$
3. In each node i in layer l, compute the gradient as:
$$\delta_i^{(l)} = (\sum_{j=1}^{s_l+1}\theta_{ji}^{(l)}\delta_j^{(l+1)})g'(sum_{j=1}^{s_l-1}\theta_{ij}^{(l-1)}h_j^{l-1} + b_i^{(l-1)})$$
4. And finally the partial derivatives can be computed as:
$$\Delta\theta_{ij}^{(l)} = h_j^{(l)}\delta_i^{(l+1)}$$
$$\Delta b_i^{(l)} = \delta_i^{(l+1)}$$

2.2.3 Convolutional Neural Network

A Convolutional Neural Networks (CNN) model, as the name suggests, it is a kind of Neural Network. Similar to the Multilayer Perception Model introduce in 2.2.2, the model consists of a number of layers stacked. Similarly, for each input sample $\mathbf{x}^{(i)}$, we run the network in the forward pass, getting the prediction in the output layer L. And during the training phase, we will compare the prediction with the target $y^{(i)}$ to update the parameters of the model until the prediction error converges.

CNN also can use the Stochastic Gradient Descent and Backpropagation algorithm to learn the optimal parameters of the model. The main difference between the MLP model and the CNN model is the type of layers included. In MLP, we built the layers stacking perceptrons vertically. Although the CNN approach can include layers of perceptrons, it can include another kind of layers such as a convolutional layer, a pooling layer, a normalisation layer, etc., which we describe later.

Note that it is common to call fully connected layers to the layers of perceptrons where each perceptron is connected to all the outputs from the previous layer in the same way we introduce the MLP model.

Common types of Layers in the CNN model

This section introduces some of the more common layers in a CNN, such as the convolutional layer, the pooling layer, and the fully-connected (FC) layer. Not all of these layers applied transformations in the parameters of the model (weights and bias). While the convolutional and FC layers perform transformations that are a function of not only the activations in the input volume but also of the parameters, on the other hand, the pooling layer implement a fixed function. Thus, only the parameters in the convolutional and FC layers will be trained with gradient descent.

The convolutional layer

The first layer on CNN is always a Convolutional Layer. The input to a convolutional layer is a matrix $X \varepsilon \mathbb{R}^{rxc}$ where r is the number of rows and c the number of columns. Here, we can define the convolutional kernel (or filter) as a matrix $K \varepsilon \mathbb{R}^{uxp}$ where $u \leq r$ and $p \leq c$. The convolution operation consists in selecting a sub-matrix $C \varepsilon \mathbb{R}^{uxp}$ from the matrix X and perform the dot product between C and K. The result of the convolution is a scalar, and we repeat the same operation for all the sub-matrix C included in X, obtaining an activation map that gives the responses of that filter at every spatial position. Figure 2.5 shows the two-dimensional convolutional layer procedure. The one-dimensional and three dimensional procedures are explained in [164] and [15] respectively.

Here, we have to define the stride, which defines the way we slide the filter throw the input matrix X. When the stride is equal to one, we will move the filter one cell at the time, when the stride is two, we will move the filter two cells at the time and so on.

Another important concept when convolving is zero-padding. We could pad the input matrix X with zeros around the border. This allows us to control the dimensions of the output volume.

We can compute the spatial size of the output volume as a function of the input volume size X, the size of the filter K, the stride which we have applied S, and the zero-padding value (P), in the form:

$$\frac{(W - F + 2P)}{S} + 1$$

For example, in Fig. 2.5, for a input size $7x7$ and a filter $3x$ with stride 1 and pad 0, we get an output of $5x5$. In this case, with stride two, we would get a $3x3$ output.

It is common to apply more than one filter to the input matrix X. Here, we will stack the activation map produce for each filter, resulting in an output volume with depth dimension equal to the number of filters.

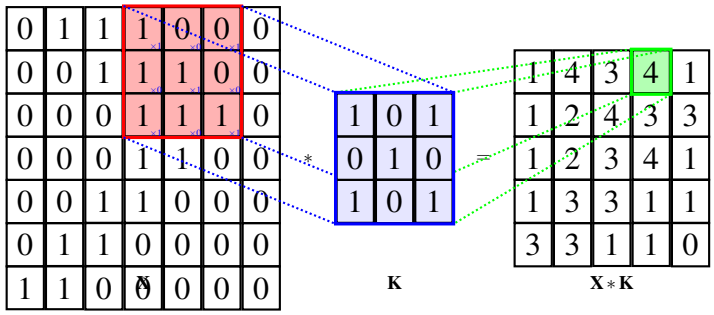


Figure 2.5 Convolution operation.

The ReLU (Rectified Linear Units) Layer It is common applying a nonlinear layer (or activation layer) immediately after each convolutional layer. The reason is to introduce nonlinearity to the system. In the past, nonlinear functions like tangential and sigmoid were used, but it has been found out that ReLU layers work better because the network canto train a lot faster without making a significant difference to the accuracy.

The RELU activation function is defined as the positive part of its argument:

$$g(z) = \mathbf{z}^+ = max(0, z),$$

and we can see its graph in Fig. 2.6.

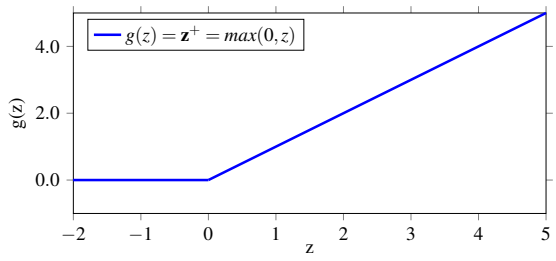


Figure 2.6 Graph of the RELU activation function.

The pooling layer A pooling operation consists in the application of a filter to reduce the dimensionality of a matrix (as well overfitting). A common pooling filter is the max-pooling. It selects each *uxp* sub-matrix of the original input matrix and selects the maximum element of the sub-matrix. It very frequently apply a max-pooling filter of dimension 2x2 and stride 2, it is shown in 2.7.

It is usually applied after a conv layer or a RELU layer.

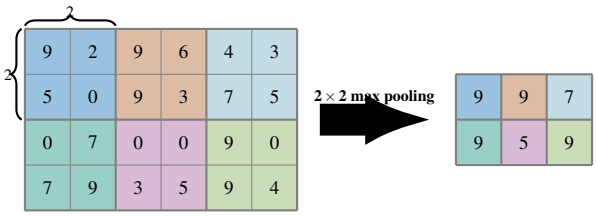


Figure 2.7 Max pooling operation in a 2D matrix.

The fully connected layer As we told before, a fully connected layer is a layer build stacking neurons vertically. The fully connected term makes reference to the fact in each perceptron of the layer l^{th} will be connected to each output from the previous layer $l^{th}-1$ (to each of the inputs if the previous layer $l^{th}-1$ was the input layer).

One difference in the CNN architectures with the MLP architectures is that being, as we saw, the hypothesis function of the perceptron:

$$h(x;\theta,b) = g(\theta^T x + b)\ .$$

It is common to use the Rectifier Linear Unit (RELU) as activation function instead of the sigmoid or the tangent functions, which are more common in the MLP approach.

2.2.4 Recurrent Neural Networks

As we said, the MLP model introduces in 2.2.2 takes an independent variable vector x and a dependent variable vector y and it learns the mapping between x and y from past data. Once the model has learned the proper parameters w and b to perform the mapping, giving a new independent variable sample vector, we predict the dependent variable (the class).

However, if the order of the elements of x matter, this model does not take it into account. A particular type of a Neural Network called Recurrent Neural Network (RNN), takes into account the order of the elements of x sharing parameters across different parts of the model. A recurrent neural network shares the same weights across several time steps.

Giving a list of m training samples such as $\{x^{(i)}, y^{(i)}\}$. Each training input vector and label will be a sequence of elements. We index each element of the input vector and label such as:

- $x^{(i)<t>} \mid 0 \leq t \leq T_x$ where T_x is equal to the length of the input sequence.
- $y^{(i)<t>} \mid 0 \leq t \leq T_y$ where T_y is equal to the length of the label sequence.

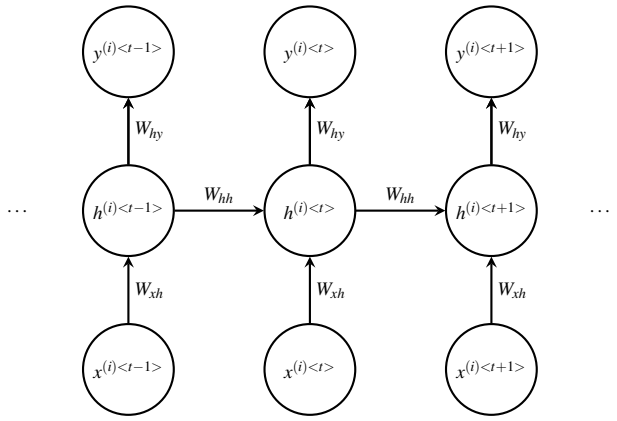


Figure 2.8 A simple Recurrent Neural Network architecture.

At time step t the input of the network will be $x^{(i)<t>}$ and the model will try to predict $y^{(i)<t>}$ taking into account the value of $x^{(i)<t>}$ and the activation value of the previous time step $h^{(i)<t-1>}$. Note that with this procedure, at time step t the neuron will take into account the value of $x^{(i)<t>}$ and all the elements with smaller t (the previous elements in the sequence) throw $h^{(i)<t-1>}$. This procedure is repeated for all the T_x elements in the sequence. The procedure is shown in Fig. 2.8. Note that in this example T_x is equal to T_y, but it is not a requirement.

Because in the time step $t-1$ there is not a previous time step, in some approaches, the neuron will take as the previous activation value a vector of 0's. It is known as the Zero activation approach, but there are others such as used a random vector.

If we call the weight parameters governing the connection from $x^{<t>}$ to the hidden layer as W_{xh}, to the parameters weights governing the connection between hidden layers as W_{hh}, and the parameters governing the connection between the hidden layer and $y^{<t>}$ as W_{hy} (these are the same of parameters for each time step)., we can describe mathematically the forward propagation of the RNN model as:

$$h^{<t>} = q(W_{xh}x^{<t>} + W_{hh}h^{<t-1>} + b_h) \tag{2.20}$$

$$\hat{y}^{<t>} = r(W_{hy}h^{<t>} + b_y), \tag{2.21}$$

where $q(z)$ and $r(z)$ are activation functions such as the sigmoid shown in 2.9, the hyperbolic tangent (tanh) or the RELU shown in 2.6. For $q(z)$,it is common use the tanh activation function. Tanh activation function is define such as:

$$g(z) = \frac{\exp^{2z} - 1}{\exp^{2z} + 11},$$

and its graph is shown in Fig. 2.9.

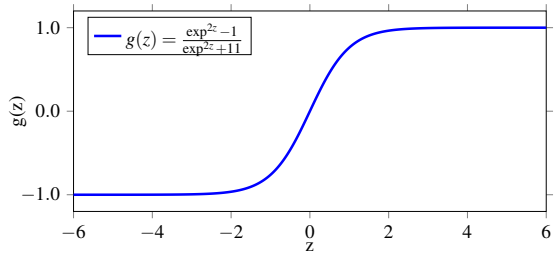


Figure 2.9 Graph of the hyperbolic tangent function.

In binary classification problems, for $r(z)$, it is a common choice to use the sigmoid or the softmax activation functions.

The softmax function constraint the outputs of each unit to be between 0 and 1, the same as the sigmoid function. But, the softmax function also divides each output such that the total sum of the outputs is equal to 1. Thus, the output of the softmax function corresponds to a categorical probability distribution specifying the probability of the sample to belong to any of the classes.

Mathematically the softmax function is:

$$g(z)_j = \frac{\exp^{z_j}}{\sum_{k=1}^{k} \exp^{z_k}}$$

We can simplify the notation in 2.20 stacking the parameter matrices W_{ah} and W_{hh} together in W_h; and the matrices $h^{<t-1>}$ and $x^{<t>}$ in $[x^{<t>}, h^{<t-1>}]$, in the way:

$$\begin{bmatrix} W_{ah} & | & W_{hh} \end{bmatrix} \begin{bmatrix} x^{<t>} \\ h^{<t-1>} \end{bmatrix} = W_{xh}x^{<t>} + W_{hh}h^{<t-1>}$$

And, we can rewrite 2.20 such as:

$$h^{<t>} = q(W_h[x^{<t>}, h^{<t-1>}] + b_h) \tag{2.22}$$

And in similar way, we can rewrite 2.21 as:

$$\hat{y}^{<t>} = r(W_y h^{<t>} + b_y), \tag{2.23}$$

Backpropagation through time algorithm

The Backpropagation Through Time (BPTT) algorithm is an expansion of the standard backpropagation algorithm that performs gradient descent on a complete unfolded network. In RNN we can define the cross-entropy loss of a single element of a sequence such as:

$$\mathscr{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1 - y^{<t>}) \log(1 - \hat{y}^{<t>}) \tag{2.24}$$

It is known as the element-wise loss force. The cross-entropy loss for the entery sequence will be:

$$\mathscr{L}(\hat{y},y) = \sum_{t=1}^{T_y} \mathscr{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}), \tag{2.25}$$

thus, different from the update function of the general SGD algorithm introduce in 2.13, for the RNN model the gradient descent weight updates have contributions from each time-step (from each element in the sequence:

$$\Delta w_{ij} = -\alpha \frac{\delta \mathscr{L}(\hat{y},y)}{\delta w_{i,j}} \tag{2.26}$$

where α is the learning rate. Thus, the partial derivatives $\delta \mathscr{L}(\hat{y},y)/\delta w_{i,j}$ have contributions from the multiple weights $w_{ij} \in \{W_{xh}, W_{hh}\}$ and depend on the inputs and hidden unit activations at previous time steps. So, the errors now have to be back-propagated through time as well as through the network.

RNN for classification

The architecture we have introduced in 2.8 corresponds to a kind of architecture called many to many. In this architecture, the input sequence x and the output sequence y are vectors. This kind of architecture is useful for problems such as text translation or creates a description of a giving image. Beyond this kind of architecture, it exits others such as:

- one to one
- one to many
- many to one

For binary classification problems such as ours, where we have a sequence of input data, and we want to map it to a class, 0 or 1, we could use the many to one architecture which graph is shown in 2.10.

Long Short Term Memory units

As we have told RNN, uses past elements in the sequence to predict the previous one. However,in practice , a limitation of the basic RNN architecture which we have seen so far, it is that they have difficulty to use information from past elements which are separate with a big gap from the previous one [16]. Long Short Term Memory (LSTM) networks which are a type of RNN, are able of learning long-term dependencies.

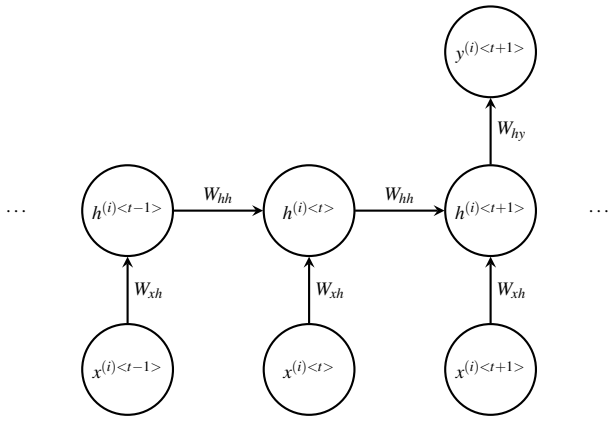


Figure 2.10 A Recurrent Neural Network with a many-to-one architecture.

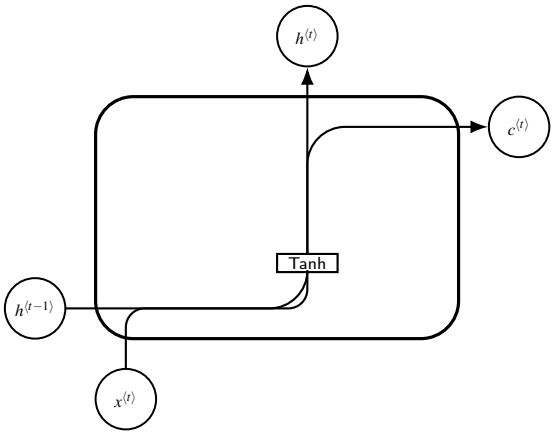


Figure 2.11 Visualisation of the RNN cell.

Figure 2.11 visualise the RNN cell, when the activation function in 2.22 is the tanh. The inputs to the cell are the element sample in the current step $x^{<t>}$ and the activation function value from the previous step $h^{<t-1>}$. The RNN cell only has one hidden neuron, in this case with the $ftanh$ activation function. This hidden neuron calculates $h^{<t>}$ in the form 2.22. The result is passed to the next RNN cell and to the output layer to calculate $y^{<t>}$.

The idea of LSTM is to include paths where the gradient can flow for long durations (these paths act as a memory). With a self-loo,p the LSTM cell included recurrence inside the cell in addition to the outer recurrence. The LSTM unit has four hidden neurons interacting between them. Some of the hidden units are used as a gate to control, based on the context, when forgot and updated the current memory state. We can see the scheme of the LSTM cell in Fig.2.12. The memory state of the LSTM unit is the $c^{(t)}$ vector, which has the same dimension as the activation function value $h^{<t>}$.

The flow in the LSTM cell is as follow (with the corresponding forward propagation equations):

- The previous value in the memory state $c^{<t-1>}$ is input to the unit. It would be deleted if the forget gate decide it.
$$\Gamma_f = \sigma(W_f[x^{<t>}, h^{<t-1>}] + b_f)$$
- The previous activation function value $h^{<t-1>}$ and the value of the current element of the sequence $x^{<t>}$ is given to the input neuron, to the update gate, to the forget gate and to the output gate.
- The input neuron (commonly with a tanh activation function) calculates the update candidate $\tilde{c}^{<t>}$ as:
$$\tilde{c}^{<t>} = \tanh(W_c[x^{<t>}, h^{<t-1>}] + b_c)$$
- The update gate decide if the update candidate value will be accumulated into the memory state:
$$\Gamma_u = \sigma(W_u[x^{<t>}, h^{<t-1>}] + b_u)$$
- At this point the new value of the memory cell is known:
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * \tilde{c}^{<t-1>}$$
which is passed to the next LSTM unit and to the output layer to calculate $y^{<t>}$.
- At the same time, the output gate has been calculating:
$$\Gamma_o = \sigma(W_o[x^{<t>}, h^{<t-1>}] + b_o)$$
And the inner product between the result of the output gate Γ_o and the known new value of the memory state, is passed to next LSTM cell as the current activation function value:
$$h^{<t>} = \Gamma_o * \tilde{c}^{<t>}$$

As we have seen, the gating units control the flow of information. All of them have a sigmoid activation function and their result has the same dimension as the activation function value.

Bidirectional Recurrent Neural Networks

There is a type of recurrent networks call Bidirectional Recurrent Neural Networks. Them connect hidden layers in both opposite directions. In this way, layers can simultaneously obtain information from the past and future states (backward and forward).

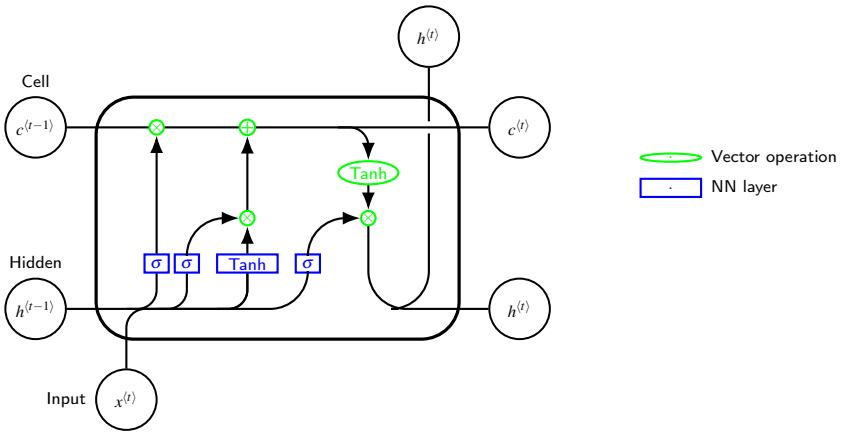


Figure 2.12 Visualisation of the LSTM cell.

2.2.5 Siamese architecture

A Siamese network is a neural network that consists of two or more identical subnetworks. The objective of this network is to find the similarity or a relationship between two comparable observations. Some examples are sentence similarity [103], where the inputs are two sentences, and the output is a score of how similar they are; or signature verification [26], where the objective is to find whether two signatures are from the same person. Another application where Siamese Networks have been used is for image similarity [11]. Here, the output indicates how similar two images are. We can see the architecture of a Siamese network in Fig. 2.13. In this example, the architecture consists of two subnetworks. Because the two subnetworks share the weights w and biases b, physically, there is a unique subnetwork which computes different observations in different time steps. We can choose the architecture of the subnetwork i.e. MLP, 1D-CNN, and 2D-CNN.

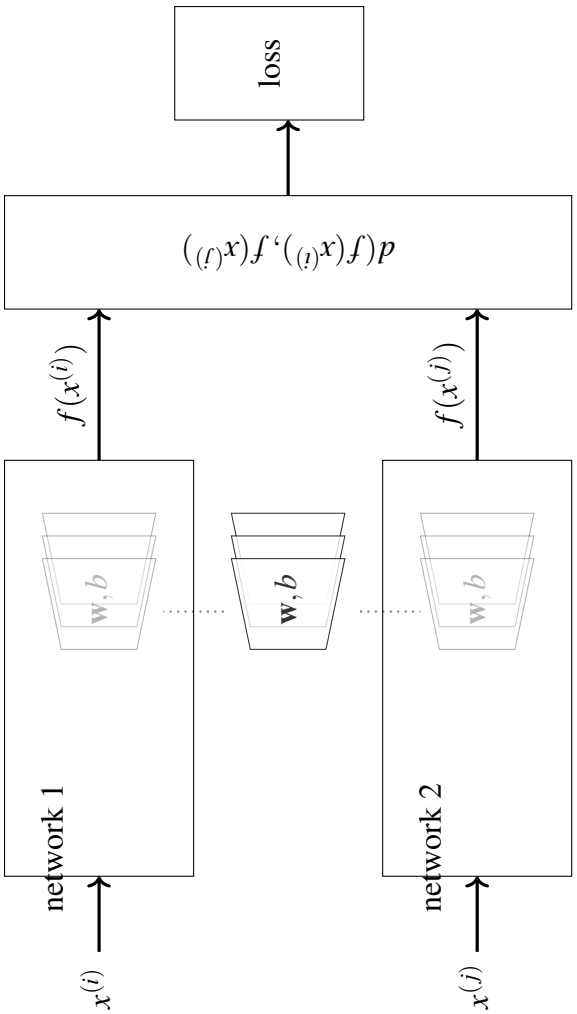


Figure 2.13 Siamese network architecture consisting in two identical subnetworks which share their parameters.

Pairwise loss function

One way to learn the parameters of the Siamese network, which gives us a good encoding for the input vector is to define the pairwise loss function and apply gradient descent on it. For each pair of samples, the distance between the output vectors of the two networks is fed into the contrastive loss function. The contrastive loss penalises small or large distances, depending on the similarity label $y^{(i)}$. We can see a Pairwise neural network architecture in Fig. 2.13 .

If we call $f_{(w,b)}(x^{(i)})$ the encoding of the input vector $x^{(i)}$ and $f_{(w,b)}(x^{(j)})$ the encoding of the input vector $x^{(j)}$. And we define the parametrised distance function to be learned $d_{(w,b)}(x^{(i)},x^{(j)})$ such as the euclidean distance between the encoding vectors $f_{(w,b)}(x^{(i)})$ and $f_{(w,b)}(x^{(j)})$ [148] [77]:

$$d_{(w,b)}(x^{(i)},x^{(j)}) = ||f_{(w,b)}(x^{(i)}) - f_{(w,b)}(x^{(j)})||^2$$

Different from other classification neural networks architectures which use the cross entropy loss function to predict the class of the observation, here, the model learns the parameters which satisfy the following conditions:

$$d_{(w,b)}(x^{(i)},x^{(j)}) = ||f_{(w,b)}(x^{(i)}) - f_{(w,b)}(x^{(j)})||^2 \begin{cases} \text{small if } f_{(w,b)}(x^{(i)}) \text{ similar to } f_{(w,b)}(x^{(j)}) \\ \text{large if } f_{(w,b)}(x^{(i)}) \text{ different to } f_{(w,b)}(x^{(j)}). \end{cases} \tag{2.27}$$

Then the loss function in its most general form is [77]:

$$\begin{aligned} \ell(w,b) &= \sum_{i,j=1}^{m} L_{(w,b)}(x^{(i)},x^{(j)},y^{(i)}) \\ L_{(w,b)}(x^{(i)},x^{(j)},y^{(i)}) &= (1-y^{(i)})\frac{1}{2}(d_{(w,b)})^2 + (y^{(i)})\frac{1}{2}\{\max(0,\alpha - d_{(w,b)})\}^2 \end{aligned} \tag{2.28}$$

where $\alpha > 0$ is called the margin.

Being objective function:

$$d(x^{(i)},x^{(j)}) \leq u\,(i,j) \in S$$

$$d(x^{(i)},x^{(j)}) \geq l\,(i,j) \in D$$

And the cost function:

$$\max(0, d(x^{(i)},x^{(j)}) - u),\ (i,j) \in S$$

$$\max(0, l - d(x^{(i)},x^{(j)})),\ (i,j) \in D$$

2.2.6 Support Vector Machines

Support Vector Machines (SVM) is a non-probabilistic binary linear algorithm that can be used for classification and regression. In binary classification problems for each class present in the training dataset, SVM defines a hyperplane in a high dimensional space. The algorithm maximises the distance between the hyperplane and the nearest training sample from any class solving a constrained quadratic optimisation problem. This distance is known as the geometric margin, and usually, a larger margin means lower generalisation error.

SVM can implement non-linear classification using a technique known as kernel trick. It will map the samples to a high-dimensional feature space to separate the different classes. Note that because being a non-probabilistic technique, the result will specify the category assigned, but it will not show the probability of belonging to the class. However, it is possible to produce probabilities from an SVM method e.g., fitting a sigmoid function that maps the SVM outputs to posterior probabilities [120].

Differently from LR where we considered h_θ equal to 1 when $\theta^T x \geq 0$, and 0 otherwise, in SVM we define a hypothesis function such as:

$$h_{w,b} = g(w^T x + b) \begin{cases} 1 & \text{if } w^T x + b \geq 0, \\ -1 & \text{otherwise} \end{cases} \tag{2.29}$$

Note that we use different notation, we refer to the parameters θ as w to follow the common notation in the literature and similarly to 2.2.2, we treat the intercept term b separately.

Separable data: Hard Margin

Given a linearly separable dataset $\{(x^{(i)}, y^{(i)} \,|\, x^{(i)} \,\varepsilon\, \mathbb{R}^p, y^{(i)} \,\varepsilon\, \{-1,1\}\}_{i=1}^m$, we can use a hyperplane to conduct binary classification. We use the relative position of each sample $x^{(i)}$ with respect to the hyperplane to predict the belonging class as shown in fig. 2.14.

Then, defining the hypothesis function 2.29 of the SVM model as $h_{w,b} = g(w^T x + b)$, the decision hyperplane can be defined by an intercept term b and a decision hyperplane normal vector w which is perpendicular to the hyperplane. Being the training data set linearly separable, we can find infinite values of w, which define different hyperplanes that could classify all the training samples correctly, as we can see in fig. 2.15.

The perceptron model introduces in 2.2.2, updates the parameters w's until they converge, defining a hyperplane that correctly classifies all training samples. This model will not take into account any consideration to choose a specific hyperplane between all of those, which

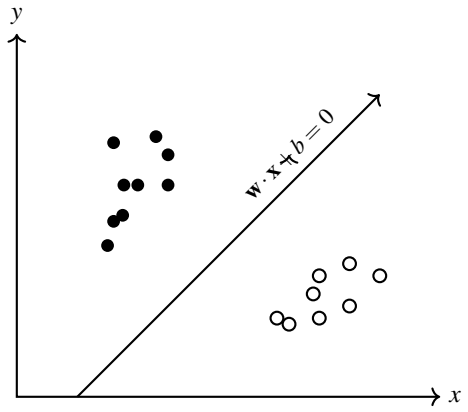


Figure 2.14 Binary classification using a hyperplane on a linearly separable data.

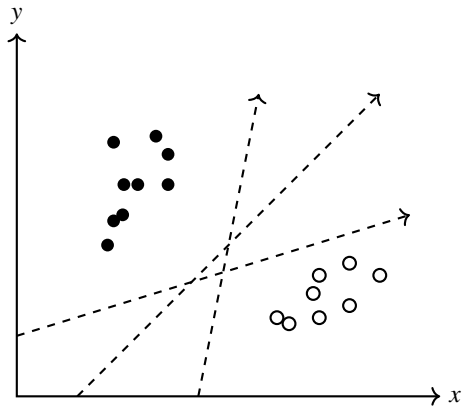


Figure 2.15 Binary classification using different hyperplanes on a linearly separable data.

can classify the training samples correctly. Thus, the perceptron model can define a different hyperplane every time we run the algorithm (modeling the same training samples) because when using the stochastic gradient descent algorithm to minimise the cost function 2.12, the parameters w and b, are initialised randomly and the training samples $\{x^{(i)}, y^{(i)}\}$are randomly chosen. Choosing a different plane each time we run the algorithm means that the hyperplane sometimes will be closer to the training samples than others. When the hyperplane is closer to the training samples, it could mean that the algorithm does not generalise properly.

In the SVM algorithm, we impose a constraint to choose the hyperplane between all of them that could classify the training samples correctly. We want to find the hyperplane $H_0 = w^T x + b = 0$ that is farther from the training samples. For that, we maximise the geometric margin. The geometric margin is the maximum width of the band that can be drawn separating the support vectors of the two classes.

Geometrically, we can find H_0 such as the hyperplane which is equidistant to two others hyperplanes H_1 and H_2 which are define by:

$$H_1 = w^T x + b = 1$$
$$H_2 = w^T x + b = -1$$

These two plains must meet the condition that cannot be any sample between them, so, for each sample $x^{(i)}$:

$$\begin{cases} w^T x^{(i)} + b \geq 1 & \text{for } x^{(i)} \text{ belonging to the class } 1, \\ w^T x^{(i)} + b \leq -1 & \text{for } x^{(i)} \text{ belonging to the class } -1. \end{cases} \tag{2.30}$$

Here, we can combine both conditions in 2.30 in a unique constraint:

$$y^{(i)}(w^T x^{(i)} + b \geq 1 \text{ where } 1 \leq i \leq m. \tag{2.31}$$

Giving the equation of the geometric margin as:

$$\gamma = \frac{2}{\|w\|},$$

where $\|w\|$ is the L_2-norm of $w \varepsilon \mathbb{R}^n$ and is given by $\|w\|^2 = w^T w$, we want to find the hyperplane with w and b values which maximise the geometric margin $\gamma = \frac{2}{\|w\|}$ among all possible hyperplanes meeting the constraints 2.31, as shown in Fig. 2.16.

In this manner, we will choose the optimal hyperplane for the giving dataset among all possible hyperplanes such as that which has a larger geometric margin. However, we can rewrite it such as minimising problem since maximising $2/\|w\|$ is the same as minimising

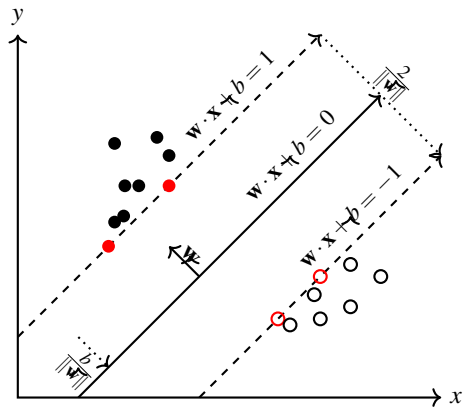


Figure 2.16 Maximum-margin hyperplane and margins for an SVM trained with samples from two classes.

$\|w\|^2/2$ [105]. So, we can rewrite the optimisation problem as:

$$\begin{aligned} &\underset{\gamma,w,b}{\text{minimise}} && \frac{1}{2}\|w\|^2 \\ &\text{subject to} && y^{(i)}(w^T x^{(i)} + b \geq 1,\ i = 1,\ldots,m. \end{aligned} \tag{2.32}$$

where we have the objective function
$frac12\|w\|^2$ and m functions $y^{(i)}(w^T x^{(i)} + b \geq 1$ which define inequality constraints. We want to find the w^* for which the objective function is at its minimum and the value meet the constraints.

Because our objective function is quadratic, it is a convex function with just a single global minimum (thus avoiding the problem with local minimums in the perceptron algorithm).

There are many algorithms to solve our optimisation problem with a quadratic function subject to linear constraints [105] implemented in quadratic programming (QP) libraries. However, we are going to use a method called Lagrange duality to solve the problem. The Dual form solution usually does better than QP and will allow us to use kernels to get the optimal margin classifiers to compute problems efficiently in very high dimensional spaces.

Lagrange Multipliers is a mathematical method used to solve constrained optimisation problems of differentiable functions.

Giving the optimisation problem:

$$\begin{aligned} &\underset{x}{\text{minimise}} && f(x) \\ &\text{subject to} && g(x) = 0, \end{aligned}$$

we find the minimum of f when its gradient point in the same direction as the gradient of g, so:

$$\nabla f(x) = \lambda \nabla g(x),$$

where λ is call the Lagrange multipliers. To find the minimum of f under the constraint g, we can solve:

$$\nabla f(x) - \lambda \nabla g(x) = 0.$$

Here, we define the Lagrangian such as:

$$\mathscr{L}(x, \lambda) = f(x) - \lambda g(x),$$

and its gradient as:

$$\nabla \mathscr{L}(x, \lambda) = \nabla f(x) - \lambda \nabla g(x). \tag{2.33}$$

Then, giving our optimisation problem with the objective function to minimise:

$$f(w) = \frac{1}{2}\|w\|^2$$

and m constraints functions:

$$g_{(i)}(w,b) = y^{(i)}(wx^{(i)}) + b) - 1, \textit{where } i = 1, \ldots, m,$$

we construct the Lagrangian function such as:

$$\begin{aligned} \mathscr{L}(w,b,\lambda) &= f(w) - \sum_{i=1}^{m} \lambda_{(i)} g_{(i)}(w,b) \\ &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \lambda_{(i)} \left(y^{(i)}(wx^{(i)} + b) - 1 \right) \end{aligned} \tag{2.34}$$

We could try to find a minimum of f solving:

$$\nabla \mathscr{L}(x, \lambda) = 0.$$

but the problem can only be solved analytically when the number of examples is small [141]. So, we rewrite the problem using the duality principle. The duality principle treats the problem from two perspectives. The first one is the primal problem, which is the minimisation problem in our case, and the other one is the dual problem, which will be a maximisation problem (the maximum of the dual problem will always be less than or equal to the minimum of the primal problem, it provides a lower bound to the solution of the primal problem). Solving the dual will lead to the same result than solving the primal and the calculations are more tractable.

Being the Lagrangian function 2.34, the primal optimisation problem is:

$$\begin{aligned} &\underset{\gamma,w,b}{\text{minimise}} \quad \frac{1}{2}\|w\|^2 \\ &\text{subject to} \quad y^{(i)}(w^T x^{(i)} + b \geq 1, \; i = 1, \ldots, m. \end{aligned} \tag{2.35}$$

To solve the minimisation problem, we take the partial derivatives of $\mathscr{L}$ with respect to w and b:

$$\nabla w\mathscr{L} = w - \sum_{i=1}^{m} \lambda_i y^{(i)} x^{(i)} = 0 \tag{2.36}$$

$$\frac{\delta \mathscr{L}}{\delta b} = -\sum_{i=1}^{m} \lambda_i y^{(i)} = 0 \tag{2.37}$$

From 2.16, we have that:

$$w = \sum_{i=1}^{m} \lambda_i y^{(i)} x^{(i)} \tag{2.38}$$

And plugging 2.38 in 2.34:

$$\begin{aligned} w(\lambda, b) &= \frac{1}{2}\left(\sum_{i=1}^{m} \lambda_i y^{(i)} x^{(i)}\right)\left(\sum_{j=1}^{m} \lambda_j y^{(j)} x^{(j)}\right) - \sum_{i=1}^{m} \lambda_i \left(y^{(i)}\left(\left(\sum_{j=1}^{m} \lambda_j y^{(j)} x^{(j)}\right) x^{(i)} + b\right) - 1\right) \\ &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} - sum_{i=1}^{m} \lambda_i y^{(i)}\left(\left(\sum_{j=1}^{m} \lambda_j y^{(j)} x^{(j)}\right) x^{(i)} + b\right) + \sum_{i=1}^{m} \lambda_i \\ &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} - \sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} - b\sum_{i=1}^{m} \lambda_i y^{(i)} + \sum_{i=1}^{m} \lambda_i \\ &= \sum_{i=1}^{m} \lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} - b\sum_{i=1}^{m} \lambda_i y^{(i)} \end{aligned} \tag{2.39}$$

Furthermore, plugging 2.37 in 2.39 we eliminated the b parameter and we obtain the Wolfe dual Lagrangian function:

$$w(\lambda) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} \tag{2.40}$$

And we can rewrite our optimisation problem in dual form such as:

$$\begin{aligned} \underset{\lambda}{\text{maximise}} \quad & w\lambda = \sum_{i=1}^{m} \lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} \\ \text{subject to} \quad & \lambda_i \geq 0, i = 1, \ldots, m \\ & \sum_{i=1}^{m} \lambda_i y^{(i)} = 0 \end{aligned} \tag{2.41}$$

Generally, when dealing with optimisation problems which involve inequality constraints, the solution must also satisfy the Karush-Kuhn-Tucker (KKT) conditions and some regularity conditions. In our specific problem, the only requirement for the solution to be optimal it is satisfying he KKT conditions [75]. The Karush-Kuhn-Tucker conditions are:

- Stationary condition:

$$\nabla w \mathscr{L} = w - \sum_{i=1}^{m} \lambda_i y^{(i)} x^{(i)} = 0$$

$$\frac{\delta \mathscr{L}}{\delta b} = -\sum_{i=1}^{m} \lambda_i y^{(i)} x^{(i)} = 0$$

- Primal feasibility condition:

$$y^{(i)}(wx^{(i)} + b) - 1 \geq 0,\ i = 1, \ldots, m$$

- Dual feasibility condition:

$$\lambda_i \geq 0,\ i = 1, \ldots, m$$

- Complementary slackness condition:

$$\lambda_i(y^{(i)}(wx^{(i)} + b) - 1) = 0, |\, i = 1, \ldots, m$$

To sum up, to solve the optimisation dual problem in 2.41 we may:

- First, we put the equations into the form of a Lagrangian give in 2.34.
- We solve for the gradient of the Lagrangian give in 2.33 which gives us a set of partial derivatives respect x, y and λ's parameters. We solve for the λ's parameters taking into account the inequality constraints and we obtain we obtain the vector λ which contain all the Lagrange multipliers. (note that only the parameters λ from the support vectors will be different from 0).
- Once we know the λ' parameters from the previous step, we obtain w^* from 2.38.
- We obtain b^* from:

$$b^* = -\frac{\max_{i:y(i)=-1} w*^T x^{(i)} + \min_{i:y(i)=1} w*^T x^{(i)}}{2}$$

- If the solution meet the KKT constraint conditions, we will have the optimal hyperplane which classify the training dataset correctly.

Using 2.29 (the same than in the perceptron model 2.2.2), we can predict the class a new sample calculating $w^t x^{(i)} + b$ and predict $y = 1$ if and only if the result is higher than one. With the new Dual formulation, using 2.38, we can rewrite $w^t x^{(i)} + b$ as:

$$w^t x^{(i)} + b = \left(\sum_{i=1}^{m} \lambda_i y^{(i)} (x^{(i)} \right)^T x + b, \tag{2.42}$$

so the prediction will be based only in the support vectors since only the parameter λ from the support vector are different from 0.

Non separable data: Soft Margin

So far, we have dealt with a linearly separable training dataset, but real datasets can include noisy that will stack the method seen so far to find the optimal hyperplane. Furthermore, this method is very sensitive to outliers — the classification hyperplane in Fig. 2.17 it is very close to the samples of the negative class (empty circles) because the positive class (full circles) includes one outlier. Intuitively, we can see that this hyperplane might not generalise correctly.

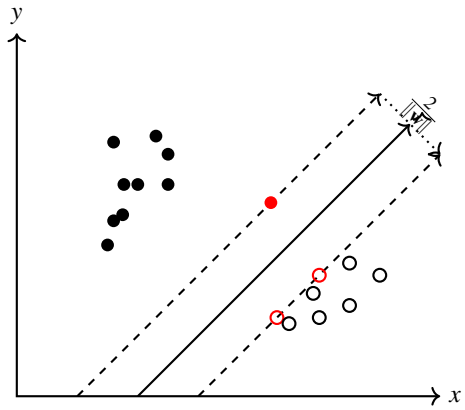


Figure 2.17 Maximum-margin hyperplane and margins for an SVM trained with samples from two classes when including outliers.

To be able to deal with non-linear and noisy datasets, we can allow the algorithm to make some classification mistakes. We can achieve this reformulating our optimisation problem 2.35 to add a regularisation term [36]:

$$\begin{aligned} \underset{\gamma,w,b}{\text{minimise}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \zeta_i \\ \text{subject to} \quad & y^{(i)}(w^T x^{(i)} + b \geq 1 - \zeta_i,\ i = 1,\ldots,m \\ & \zeta_i \geq 0,\ i = 1,\ldots,m. \end{aligned} \tag{2.43}$$

This formulation is known as the L1 soft-margin SVM. The changes we have introduced are:

- we change the constraint 2.31 adding a parameter ζ to allow the geometric margin to be less than one:

  $$y^{(i)}(w^T x^{(i)} + b \geq 1 \text{ where } 1 \leq i \leq m.$$

  So, some samples could be misclassified or be closer to the hyperplane.

- We add the term $\sum_{i=1}^{m} \zeta_i$ to the objective function to penalise the choice of a high value of ζ (if the value of ζ was very high, the constraint always would be meet. It is called the regularisation term.

- We multiply the regularisation parameter by a constant C to control the trade-off between making the $\|w\|^2$ small (making the margin larger) and ensuring that most training samples have functional margin at least 1.

- We avoid minimise the function using negative values of ζ adding a constraint that:

  $$\zeta_i \geq 0, i = 1, \ldots, m$$

Here, we can rewrite the soft-margin optimal problem in dual form such as:

$$\begin{aligned} \underset{\lambda}{\text{maximise}} \quad & w\lambda = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} \\ \text{subject to} \quad & 0 \leq \lambda_i \leq C, i = 1, \ldots, m \\ & \sum_{i=1}^{m} \lambda_i y^{(i)} = 0 \end{aligned} \tag{2.44}$$

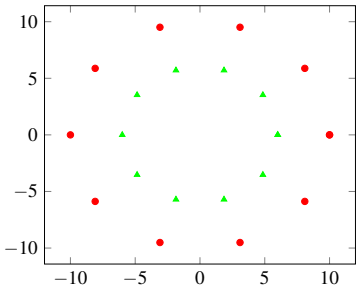
So, we only change the constraint from the dual form of the hard case 2.41.

We could reformulate the quadratic optimisation problem in 2.43 to allow multiclass classification. Binary classification is performed by the plane defined by the quadratic optimisation problem. Multiclass classification is done by combining the output of all classifiers

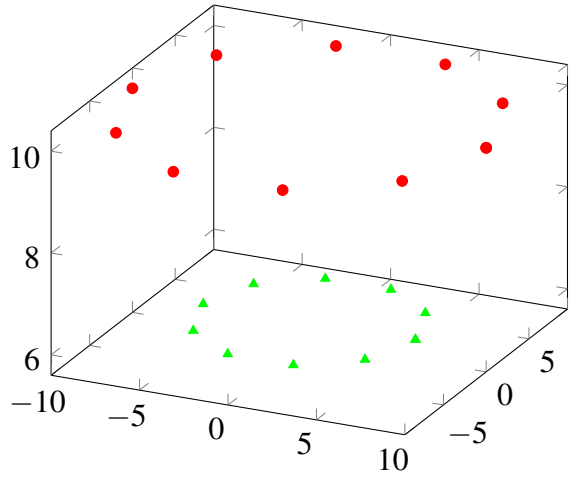
Kernels

Giving a dataset that is not linearly separable such that shown in 2.18a, we can map the original features to a new set of features that can be linearly separable using a feature mapping denote by ϕ. So, rather than applying SVMs using the original input attributes x, we may instead want to learn using some features $\phi(x)$. Thus, giving a feature mapping ϕ, we define the corresponding kernel as:

$$k(x,z) = \phi(x)^T \phi(z)$$



(a) Two dimensional dataset non linearly separable.



(b) Radial transformation of two dimensional dataset.

Figure 2.18 Example of radial transformation of two dimensional dataset

Kernel trick: the transformation from the original feature space to the new can be an expensive calculation when our dataset is big. However, we do not need explicitly transform the data since the SVM model is entirely written with inner products $\langle x, z\rangle$, so we can replace all these inner products with $\langle \phi(x), \phi(z)\rangle$. Thus, we can rewrite the soft-margin dual optimisation problem as:

$$\begin{aligned} \underset{\lambda}{\text{maximise}} \quad & w\lambda = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} \\ \text{subject to} \quad & 0 \leq \lambda_i \leq C, i = 1, \ldots, m \\ & \sum_{i=1}^{m} \lambda_i y^{(i)} = 0 \end{aligned} \tag{2.45}$$

There are some common types of kernels such as:

- Linear kernel: It is the most simple kind of kernel where:

$$k(x, x') = xx'$$

- Polynomial Kernel: It has the parameter c which is a constant term and d which is the degree of the kernel:
$$k(x,x') = (xx' + c)^d$$

- Radial Basis Function Kernel: it projects vectors into an infinite dimensional space. The parameter γ defines how far the influence of a single training samples reaches.
$$k(x,x') = \exp(-\gamma \|xx'\|^2)$$

Choosing which transformation to apply depends on the specific dataset, and the decision should be made via trial and error.

2.2.7 Decision Tree

Decision trees are decision support methods that use tree-like models to map a series of inputs to possible related outcomes. They are non-parametric techniques, so they model the relations between the input and output data spaces without any prior assumption. Some of the advantages of this kind of algorithm are:

- They are interpretable.
- They are robust to noise and outliers because they intrinsically implement feature selection.
- They can learn efficiently from small training sets.
- They can deal with high-dimensional feature spaces and complex structures.
- They can handle qualitative and quantitative attributes.

Tree-based algorithms approximate the Bayes model's partition by recursively partitioning the input space $\mathscr{X}$ into subspaces R_j. Then, a prediction value $\hat{y}$ is assigned to each terminal subspace $R_j \,|\, i = 1,\ldots,J$. In regression trees, usually, the prediction value assigned is the mean or mode of the training samples belonging to the particular sub-space.

We can summarise the set of splitting rules in a tree. Trees are usually drawn upside down, being the leaves at the bottom of the tree. We connect the vertices of the tree by one unique path (an edge). The root of the tree has all the edges away from itself. When there is an edge from the node a to the node b, the node a is the parent node of the node b, which is the child. We call a node internal if it has one or more children and terminal if it has no children (the

leaves). Fig. 2.19 shows a graphical representation of a decision tree. We can interpret the importance of each feature by its hierarchical position in the tree.

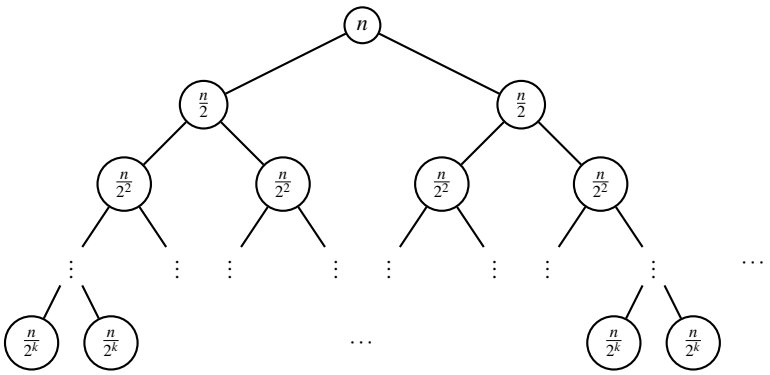


Figure 2.19 Graph of a decision tree.

Classification trees

We use classification trees to predict categorical classes.

To build the tree, we perform recursive binary splitting. First, we select an attribute $x_i \,|\, i = 1,\ldots,n$ and the cut-point $sin[\min\{x_i..\max\{x_i\}$, splitting the attribute space in two regions $\{x|x_i \leq s\}$ and $\{x|x_i > s\}$. We repeat the same operation iteratively, dividing the attribute space x into J non-overlapping regions $R_j \,|\, j = 1,\ldots J$. Then, the algorithm will predict the same class for each new sample $x^{(i)}$ which fall in the same region R_j. Classification trees assigns the class of the training samples which most occurring frequency in the region. So, if a node t, representing a region R_t with M_t observations, the proportions of samples with class k in the node is:

$$\hat{p}_{tk} = \frac{1}{M_t} \sum_{x^{(i)} in R_t} I(y^{(i)} = k),$$

then the class predicted in the node t is that of the majority of observations in the region:

$$k(t) = arg \max_k \hat{p}_{tk}$$

To grown the tree, for each split a greedy algorithm is used to decide which attribute x_i and which cut-point s should be choose. From all the observations M_t in the node t, considering a splitting feature x_i and split point s, for each candidate $\theta(i,s)$ partition the data in:

$$R_{left}(i,s) = \{x|x_i \leq s\} \text{ and } R_{rigth}(i,s) = \{x|x_i > s\}. \tag{2.46}$$

and being the number of observations in R_{left} equal to M_{left} and the number of observations in $R_{rigth}(i,s)$ equal to M_{right}, we define the impurity in the node t using an impurity function

$I()$ as:

$$G(R,\theta) = \frac{M_{left}}{M_{left}+M_{rigth}} H(R_{left}(\theta)) + \frac{M_{right}}{M_{left}+M_{rigth}} H(R_{right}(\theta))$$

We will select the attribute and which cut-point $\theta(i,s)$ which minimise $G(Q,\theta)$:

$$\theta^* = argmin_\theta G(R,\theta)$$

For the case of classification trees common choices of the impurity function $I()$ are:

- Misclassification error:

$$I_{misclassification} = 1 - \hat{p}_{tk} \tag{2.47}$$

- Gini impurity :

$$I_{Gini} = \sum_{k=1}^{K} \hat{p}_{tk}(1-\hat{p}_{tk}) \tag{2.48}$$

- Shannon's Entropy:

$$I_{Entropy} = -\sum_{k=1}^{K} \hat{p}_{tk} \log \hat{p}_{tk} \tag{2.49}$$

Once we have found the best split for the node t, we partition the data into the two resulting regions, and we repeat the splitting process iteratively on each of the new two regions.

Here, we have to decide how long to split the tree. Very large trees may overfit the training data, while small trees may not capture the underneath data relations [78]. A common method to decide the tree size is grown a large tree T_0 ending the splitting process using a predefined stopping criterion, such as a minimum number of observations in each region R_j. Then this large tree is pruned using a method called cost-complexity pruning. This process consists in, for all the $T \subset T_0$ which can be obtain by collapsing any number of the internal nodes of T_0 calculate the complexity criterion define such as:

$$C_\alpha(T) = \sum_{t}^{|T|} M_{left} + M_{right} I(T) + \alpha|T|,$$

where $|T|$ is the number of nodes of T, $I()$ is one of the impurity functions seen in 2.47,2.48 and 2.49; and the scalar $\alpha \geq 0$ defines the trade-off between the size and the goodness of the tree to capture the underneath data relations. To find the appropriate tree size, we should find the value of α and T, which minimises $C_\alpha(T)$.

Ensemble methods

Decision trees are known to suffer from bias and variance i.e. high bias with simplest trees and high variance with complex ones. Ensemble methods, combines several decision trees to improve the performance of the single decision tree. techniques to perform ensemble decision trees:

A common way to construct several learners is using methods such as Bagging (Bootstrap Aggregating) and boosting. The procedure of these two methods is:

1. Producing distributions of simple models on subsets of the training dataset.
2. Combine all distributions into one aggregated solution.

Bagging and Boosting generate additional data in the training phase by random sampling with replacement from the original dataset. By sampling with replacement, some observations may be repeated in each new training data set. The difference between them is that bagging samples each observation with the same probability while boosting weight the observations, giving to the misclassified samples more preference. Thus, while Bagging decreases the variance of the prediction and avoids overfitting, boosting decreases the variance and bias. Furthermore, both methods give stability to the predictions.

To predict the class of $y^{(i)}$, we will use the prediction of each of the learners. In Bagging, we obtain the result by averaging the responses of the learners (or majority vote on classification problems). Boosting takes the weighted average (or average voting), giving more importance to the prediction from learners with good classification results in the training phase.

Random Forest

Random Forest (RF) is a learning method for classification and regression. RF constructs multiple decision trees to predict to the class of an unseen sample $y^{(i)}$ based on the prediction of all of them.

1.
2.

RF algorithm tries to improve the variance reduction of Bagging by reducing the correlation between the trees. For this, when growing the three, RF chooses the splitting attribute between a random selection of all of them.

Thus, RF improves the prediction accuracy of other tree-based methods such that seen in 2.2.7 (at expenses of losing some interpretability) [78].

The steps which follow RF are summarised in:

1. For each tree e of the desired number of trees E of the forest:
   (a) Draw a bootstrap sample from the training dataset
   (b) Grown an RF decision tree until reaching the minimum node size
      i. Randomly select a subset of the attributes
      ii. Pick the best split attribute/split point
      iii. Split the data space in to regions
2. Combine the ensemble of trees $\{T_e\}_1^E$

And to make a prediction of an unseen sample $y^{(i)}$:

- For regression:
$$\hat{f}_{rf}^{B}(x) = \frac{1}{E}\sum_{e=1}^{E} T_e(x)$$
- For classification:
$$\hat{C_{rf}^{E}}(x) = \text{ majority vote } \{\hat{C}_e(x)\}_1^E,$$
where $\hat{C}_e(x)$ is the class prediction of the e^{th} RF decision tree.

We should tune the parameters of the model, such as the size of the data subset, number of trees, and minimum node size, to choose the optimal.

2.2.8 k-nearest neighbors

k-nearest neighbors (K-NN) is a simple non-parametric method which can be used for classification. It assumes that samples with similar patterns are close to each other in the feature space. It is useful to perform discriminant analysis when parametric estimates of probability densities are unknown or difficult to determine.

It classifies samples based on the majority vote of the k neighbors i.e., the predicted class will be that of the most common class among the k nearest neighbors.

There are different metrics to calculate the distance between the sample and their neighbors, and one way might be preferable depending on the specific problem [123]. However, Euclidean distance is a popular choice.

The K-NN Algorithm steps are:

- Load the data and chose the number of neighbors k
- For each of the points in the feature space:
    1. Calculate the distance between the current sample and the rest of them
    2. Sort the collection of distances in ascending order
    3. return the most common class of the K first entries

2.2.9 Hidden Markov Model

Hidden Markov Model (HMM) is a probabilistic approach for modeling sequences to capture hidden information, i.e., that cannot be observed, in markovian processes.

Markov chain is a stochastic model which describe a sequence by:

- possible stages
- the probabilities of moving from one stage to other i.e. the transiction probabilities.
- the initial state of probabilities

And the characteristics of the model are:

- The number of stages is finite
- The probability of the next stage depends only on the current stage and it does not in the previous ones i.e. memoryless property.
- The transiction probabilities are constant over time.

We can discriminate between discrete-time Markov chain when events occur in discrete time steps and continuous-time Markov chain, when time is continuous.

We define S as the set of possible steps $S = S_1, S_2, ...S_m$. If at current time n the sytem is in the state i the probability to be in the state j at the next step $n+1$ is $P(X_{n+1} = j | X_n = i)$ with $p_{ij} \geq 0, \forall i, j, \in S \sum_{j \in S} p_{ij} = 1, \forall i \in S$.

We can represent the Markov chain by a trasiction graph such in 2.20.

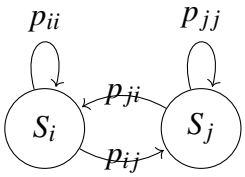


Figure 2.20 Markov chain graph.

Hidden Markov Model sequences are generated from two coexistent stochastic processes: a process defining the movements between stages and a process that define an output.

The process which defines the movements between stages is a Markov chain. And the process which defines the output is characterised by the emission of one character of a given alphabet from each state, with a probability distribution that only depends on the state. So, it is defined by emission probabilities and initial probabilities.

We cannot observe the sequence of transitions directly, but we can guess there observing the sequence of emitted symbols, i.e., it is because of the name of the Hidden Markov Model.

2.3 Unsupervised Machine Learning methods

In this thesis, we will use unsupervised machine learning methods for classification and dimensionality reduction. In this section, we introduce the approaches we have used.

2.3.1 Deep Learning Autoencoders

In Chapter 6, we use a particular deep learning method called an autoencoder, which consists of an input layer, an output layer of equal size, and one or more hidden layers connecting them. Autoencoders have been used for data representation [171] and, more recently, for authentication [31].

In this context, the input is the input vector $\mathbf{t}_i = (a_1, ..., a_n)$ and the output is:

$$\mathbf{u}(\mathbf{t}) \quad = \quad h_u(\mathbf{W}_u\mathbf{t}+\mathbf{b}_u)\,, \tag{2.50}$$

where $\mathbf{W}_u \in \mathbb{R}^{d\times s}$ is a weight matrix, $\mathbf{b}_u \in \mathbb{R}^s$ is the bias vector, $a_1, a_2, ..., a_n \in \mathbb{R}^d$ are the input features of the sample i and h_u is called the activation function, which in this approach we define such the hyperbolic tangent function [88]. The process of the approach is performed in two stages: the encoding and decoding steps. In the encoding step, the input a is mapped to the abstract representation $\mathbf{u}(\mathbf{t})$ according to Eq. 2.50, and in the decoding step, the latent space is reconstructed to the output representation $\hat{\mathbf{t}}$, which is an approximation of the input vector, according to the decoder function:

$$\hat{\mathbf{t}} \quad = \quad h_d[\mathbf{W}_d\{\mathbf{u}(\mathbf{t})\}+\mathbf{b}_d]\,,$$

where $\mathbf{W}_d \in \mathbb{R}^{s\times d}$ is the weights decoding matrix, $\mathbf{b}_d \in \mathbb{R}^s$ is the decoding bias vectors, and h_d the decoding activation function. We restrict the degrees of freedom using a tied architecture, where the encoding matrix is the transpose of the decoding matrix, i.e. $\mathbf{W}_d = \mathbf{W}_u^T$ [160].

More than one hidden layer can be applied to achieve higher flexibility (and abstraction) in the model. In a multiple layers architecture, encoders and decoders are stacked symmetrically, where the output from the k^{th} encoder, is the input of the $k+1^{th}$ encoder.

We train the model using back propagation to minimise the loss. After, we compute the mean squared error (MSE) between the input vector t and its representation $\hat{t}$ on the output of the autoencoder.

When the goal is to reconstruct the input as accurately as possible, a loss function is frequently used, i.e., Mean squared error (MSE). MSE is the mean of the squared difference between the input vector t and the network output $\hat{t}$.

Two of the most popular applications of autoencoders are data denoising and dimensionality reduction i.e., with dimensionality and sparsity constraints, autoencoders learn new data projections.

Less often, autoencoders are used for novelty detection. The model is trained with a unique class of samples. If the MSE of a new sample is higher than a threshold, this sample is considered an anomaly.

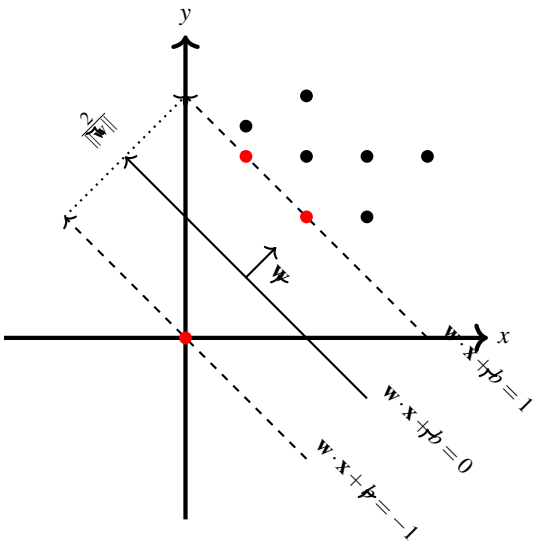


Figure 2.21 Maximum-margin hyperplane and margins for an one-class SVM trained with samples from one class.

Variatonal Autoencoder

The difference between Variational autoencoders and vanilla autoencoders is that in the former, we add regularisation during the training to avoid overfitting and ensure characteristics of the latent space-specific of generative models.

Variational autoencoder as well is trained through gradient descent in an encoder-decoder manner, but instead of encoding an input sample as a single vector, we encode it as a distribution over the latent space.

Thus, the Variational autoencoders training steps are:

- encode the input as a distribution over the latent space
- sample point from the latent space distribution
- decode the sample point and calculate the reconstruction
- propagate the reconstruction through the network

Then, the loss function that is minimised to training the VAE consists of the reconstruction term and a regularisation term that makes the distribution of the latent space close to a normal distribution.

That regularisation term is formulated as the Kulback-Leibler divergence between the returned distribution and a Gaussian distribution [109].

Minimising the KL divergence, we optimise the probability distribution parameters (μ and σ) to closely resemble that of the target distribution. For VAEs, the KL loss is equivalent to the sum of all the KL divergences between the component X_i $\mathcal{N}(\mu_i, \sigma_i^2)$ in X, and the standard normal distribution i.e it is minimised when $\mu_i = 0$, $\sigma_i = 1$. This stochastic generation means that the encoding will have a random component and will vary on each pass.

VAEs have proof better performance for some applications, i.e., generating variations on an input image.

2.3.2 One-class SVM

As we saw in the previous section, SVM learns how to discriminate between two data categories. From past data, it finds the line/plane/hyperplane that best separates the samples from both categories.

One-class SVM is an extension of the binary SVM learning algorithm to enable the training of the classifier only with samples of one class. So, one-class SVM tries to identify if new samples belong to the same distribution of observable samples from one specific class. Training can be achieved by treating the origin (of the coordinate system) as the only member of the second class 2.21, separating a certain number of samples from the rest of them. Thus, It forms the decision boundary around the learned data domain without knowledge of the samples outside the boundary.

Mathematically, it can be formulated by providing a measure $f(z)$ of the distance $d(z)$ to the positive class and a threshold θ to distinguish between the positive class and the outliers:

$$f(z) = I(d(z) < \theta_d \tag{2.51}$$

where I is the class (negative or positive).

We can apply kernels to one-class SVM in the same way we did for the binary SVM in the previous section. It will allow us to model more complicated datasets. Same that in the SVM case, data points are mapped by mean of the kernel to a high dimensional feature space. In the new feature space, we will search for the maximal separation between classes.

2.3.3 Multivariate Gaussian Model

Given the dataset $x_j^{(i)}$, we will take into account only those samples labeled as a normal. We assume that each attribute is normally distributed and we calculate the Gaussian parameters i.e. the mean μ_i and variance σ^2 for each of the features as follow:

$$\mu_i = \frac{1}{m}\Sigma_{j=1}^{m} a_i^{(j)} \tag{2.52}$$

$$\sigma_i^2 = \frac{1}{m}\Sigma_{j=1}^{m} (a_i^{(j)} - \mu_i)^2 \tag{2.53}$$

where $i \in \{1,2,...,d\}$ and d equal to the number of features.

Given a new sample, we will calculate the probability to belong to the same distribution as follow:

$$\begin{aligned} P(\mathbf{t}) &= P(a_1;\mu_1,\sigma_1^2)P(a_2;\mu_2,\sigma_2^2)...P(a_d;\mu_d,\sigma_d^2) = \\ &= \prod_{j=1}^{d} P(\mathbf{a}_j;\mu_j,\sigma_j^2) = \prod_{j=1}^{d} \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\left(a_j-\mu_j\right)^2 \big/ 2\sigma_j^2} \end{aligned} \tag{2.54}$$

And we will consider the transaction as fraudulent if $P(\mathbf{t}) < \varepsilon$ where ε is the probability threshold.

Chapter 3

Card Payment systems

3.1 Introduction

The value of global non-cash transactions is growing every year, and it is estimated to reach beyond 720 billions of dollars in 2020 [29]. Fig. 3.1 breaks down the total transactions value by global growth regions, i.e., North America, Europe, Mature Asia-Pacific (APAC), Emerging Asia, Central Europe Middle East and Africa (CEMEA) and Latin America (LATAM) between 2012 and 2021 (Note that values between 2019 and 2021 are estimated). We can see an increasing trending, which is significantly stronger in emerging Asia and CEMEA. In those regions, where the card network development is immature, the proliferation of card use is mainly due to the increase in mobile payments and wallets. On the other hand, in mature markets such as North America, Europe and mature APAC, the adoption of NFC/contactless technology has powered the increment of the card operations.

At the same time that the value of the global card payments transactions grown, fraud activities and losses related to them have increased too. During the last few years, most of the losses are related to Card Not Present (CNP) interfaces [151]. CNP involves online, telephone and mail transactions, being most of the losses because of online transactions.

The European Central Bank, in its last report on card payments fraud, informed that CNP fraud increased 66% for five years with an approximate 1000 EUR millions of losses in 2016 in Europe [57].

Fighting fraud is a difficult task, and merchants prefer that security measures have little or no impact on the customer experience. On the other hand, security procedures against

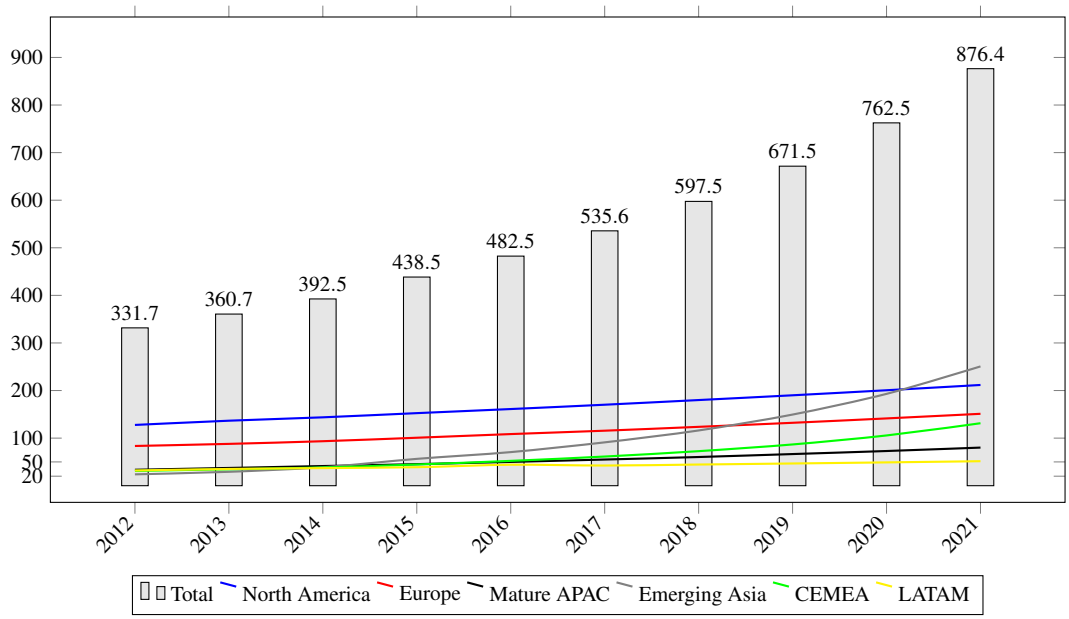


Figure 3.1 Estimated value of world wide non-cash transactions from 2012 to 2021 [167]

online fraud which require extensive personal information are also a source of vulnerability. This information can be exposed after data breaches, and once stolen, it can be used in fraudulent activities. Furthermore, customers used the same access information for different sites, allowing fraudsters to make a profit in many different ways.

3.2 Card payments

Card payment is a financial system where an institution, i.e., a bank, issues a card to clients, which enables its owner to access the funds in the customer's designated account, pay with electronic funds transfer, and access automated teller machines.

3.2.1 Actors involved

For a better understanding of the different scenarios of card payments, it is essential to know who are the actors involved. Figure 3.2 shows those playing a role in one way or other in card transactions which are:

1. The holder or applicant: The cardholder is the natural or legal person in whose name the card is issued. The credit institution authorised him to use it under the contract signed between the financial institution and the applicant. The applicant and the holder can be a different actor or the same. On the other hand, some financial institutions allow

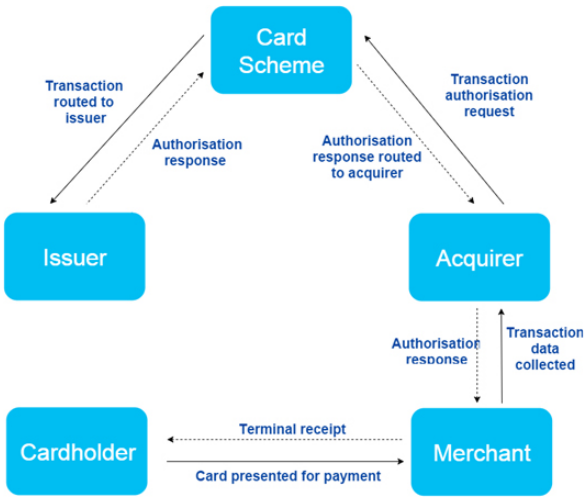


Figure 3.2 Card payments scheme.

different holders for the same card, where a single contract results in the emission of several cards. Here, we distinguished between the principal holder and the authorised holders. When the applicant is the holder too, he will be in charge of the liabilities of the contract. On the other hand, when the applicant is not a holder, the authorised holders will be liable.

2. The merchant: they are the shops that accept a card as a means of payments for goods or services which they offer. The shop will accept card payments from one or more card providers. There is a contract between the shop and the different providers (when there are more than one). The store guarantees the acceptance of the payment enabling a terminal point of sale (POS) compatible with the system of the care provider in question.

3. The Issuer: it is the entity which, as part of his business, makes available to a customer a payment card, under a contract concluded between them.

4. The acquirer: it is the financial institution or bank which is responsible for authorising and processing payments by card through the payment gateway system.

5. Card scheme: This is the brand owner of the credit card, which grants the license to financial institutions that are members of its system for issuing this payment to individuals and companies, and conclude contracts with businesses that want to join the system of payments, i.e., Visa international.

3.2.2 Card payments types and procedure

Commonly, we can categorise card payment systems into either card-present and card-not-present (CNP) systems.

In *card-present* payment systems, the cardholder is physically present at the merchant store, and he/she perform the payment by swiping (magnetic stripe), inserting (chip and PIN) or tapping (contactless) payment card to the merchant provider point of sale (PoS), i.e., a terminal or reader. Here, in the payment process, the identity of the cardholder is validated either by requesting a card's PIN or by the cardholder signature. However, note that in the case of contactless payments, which usually only accept transactions of a low amount of money, i.e., a maximum of 30 in the UK [1], the identity of the cardholders is not validated.

On the other hand, in *Card Not Present (CNP)* transactions, the process takes place remotely, i.e., the cardholders type their payment card details on the checkout page provided by a merchant website. The security of the CNP payment system relies upon the cardholders correctly providing their payment card details. A CNP payment system has two grievous security limitations. Firstly, the identity of the actual cardholder cannot be established by the merchant or by the card issuer, and secondly, the card details are static and remain the same until the card service is expired. To overcome these limitations, the payment industry proposes a user-authentication scheme which requires the cardholder to establish their identity with the card issuer before the transaction is approved. During many years, Fraud detection systems (FDS's) for CNP transactions has been based on rules established by experts. The main limitation of these systems is the inability of the approach to recognise new types of fraud that have not been previously target. Modern approaches establish dynamically the ruling criteria for detection using Machine Learning techniques, which learn from past data.

3.2.3 Authentication method on card payments

To make a transaction using a card, it is required to follow the authentication steps. The authentication method varies according to the type of transaction. We are going to describe some of the most popular authentication methods for the two basic types of transactions: Authentication methods for card-present transactions

- Signature stripe: A signature strip is an area on the back of the card that can hold ink. Owners of the card have to sign there for identification purposes. This method

of authentication is used to support a more secure method such as Magnetic Stripe or Chip & Signature authentication method.

- Magnetic stripe: The magnetic stripe is a black or brown band. This band is made adding to a resin small magnetic particles such as iron oxide or barium ferrite. Thanks to the ferromagnetic activity of these compounds, the stripe will be able to be magnetised, creating a kind of bar code consists of magnetised areas. When the card is swiped at a certain speed in a specific reader, the magnetic induction creates a voltage which will be translated into a binary code. The first swipe cards were used in the early sixties on public transport, London Transit Authority installed a magnetic stripe card in the London Underground train system in London. In the late sixties, financial institutions implemented the plastic card with a magnetic strip. Some of the information stored in the magnetic strip is the card number, the holder's name, and the expiration date. A problem with this authentication method is that the information is not encrypted. Anyone with the card can access the information and duplicate that with a simple device to clone magnetic stripes. Sometimes as part of the process, the cardholder will be required to provide a signature on the terminal receipt to authorise the transaction. Furthermore, many times, the user will be asked for an id proof too.

- EMV card: The EMV (Europay MasterCard VISA) system is named after the three companies that developed the project. Later in 2004, Japan Credit Bureau (JCB) joined the project and American Express in 2009, so most of the cards issued worldwide to implement this technology. EMV validates the operations through information stored in a chip instead of in a magnetic stripe. There are two types of EMV authentication methods to consider: Chip & Pin and Chip & signature.

- Chip & PIN: A Chip and PIN card requires the cardholder to enter a four-digit password. The majority of UK issued cards will be processed as chip & PIN transactions [The UK Cards Association].

- Chip & signature: they require that the cardholder sign on a piece of paper that comes up after a machine reads the card's chip. The merchant will compare in situ the signature of the user with the signature of the signature stripe in the back of the card.

- Contactless transaction: the payment is made bringing the card near the terminal. This method, which was announced during the London Olympics 2012 by Visa, is configured not to have to enter the PIN code to pay an amount smaller than £20. Undoubtedly, these cards allow us to pay faster, but they have against you in case of

theft, payments can be made without knowing the PIN code. Data such as cardholder name, date of issuance, card number, or expiration date, can be obtained by a smart card reader. This data can be read by anyone because it is not encrypted. Overall the information that is printed on the card, according to the EMV standard [4], the only thing that cannot be obtained is the Card Verification Value (CVV) number. What makes different the contactless cards is that they can be read without having to maintain contact with it, thanks to the Near Field Communication (NFC) technology. Nowadays, almost all mobile phones support this technology. Placing an NFC smartphone near a card, it can be read. Note that this communication can be established even if the card is into our wallet. In September 2015, "£758.6m was spent in the UK in the month using a contactless card. This is an increase of 19.7% on the previous month and 19.8% over the year" [The UK association] .From 1st September 2015, the higher contactless limit in the UK has been moved from £20 to £30. This could bring more retailers' average shop or individual items into range for contactless cards and should drive higher acceptance rates [The UK association] .

Authentication methods for card no present transactions

When a user wants to make a card not present transaction by online, mail, or telephone, the merchant asks the customer about some information to identify the owner of the card and validate the purchase. There is no regulation about which data the merchant have to request, and they decide what to ask [89]. Some of the information given by the user can be compared with the information stored by the card provider, but once more, the merchant is free to decide which information check. Some of the information asked the customer by the merchant it is not printed the card, such as the card holder's postcode and some information are printed on both sides of the card such as:

- Card number: It is a number between 13 and 18 digits. Often it is a 16-digit number. Typical exceptions are American Express which has 15 digits, Diners Club which is between 14 and 15 digits. This number is unique for each issued card. Since 1989 the numbering of all credit or debit cards must be adapted to the ISO/IEC 7812 standard [81] . This standard identifies the positions and meanings of the numbers of the card number. Holder's name: Name of the owner of the card.
- Expiration date: date until when the card will be useful. The format is mm/yy.

- Verification number: it was designed to increase security in electronic transactions. It is used to confirm that the client has physical access to the card being used in a transaction. This system takes different names CVV2 (Visa), CVC2 (MasterCard), or CID (American Express). Visa, Maestro, MasterCard and Eurocard, the number is on the back of the card, near the signature. The number consists of three digits. For American Express cards, the number is located in the front of the card, right above the number of the credit card. The number consists of four digits. The verification code is printed flat, not embossed like the card number or the expiration date, this makes it impossible to obtain copies by coal or similar techniques. Usually, the customer must provide the card holder's name, the card number, the expiration date and the CVV. On many websites, it is mandatory to enter the CVV, but it is not the case for all, for example, in Amazon. Here only the name of the holder, the card number, and the expiration date is necessary. That means that someone with a mobile NFC can read a contactless card and then make a purchase on Amazon [45] . Moreover, if we will not manage to get the card holder's name by this method, anyway we could make the purchase only with the card number and the expiration date because Amazon only checks the credit card number and expiration day with the card provider [5] .

3.3 Frauds Over Internet Technologies

3.4 Frauds Over Internet Technologies

Commonly, cybercriminals use Internet technology for committing fraud activities in two different ways:

1. spreading malicious content, e.g., malware, Trojans, or viruses that, in turn, leak private information of the victims

2. convincing victims to disclose their private information via social engineering attacks

Internet-based applications have brought uncountable new opportunities for businesses, but at the same time, it has smooth the way for fraudsters to use the latest technology to commit fraud against users and businesses. Every year, a large number of people lose their money to different types of fraud over the Internet applications such as e-commerce, online dating, online gaming, credit card frauds, telephone frauds, mobile payment frauds, etc. The e-commerce frauds can be of a different type: e.g., the merchant does not deliver the product

or has sent a product of lower quality than the advertisement. Some of the most common online frauds are buyers not receiving goods that they have ordered, receiving products that have inferior value, or are significantly different from the original description. The statistics by Experian show that e-commerce frauds (online auctions, buying products) have increased by 33% since 2015 [65]. Frauds over the online marketplaces have resulted in an annual loss of billions of dollars to customers all over the world.

Over the Internet, fraudsters have created a large number of fake pages for two purposes: 1) they convince end-users to click on some links through social engineering or phishing attacks or exploit some browser system vulnerabilities to silently download malicious software on the user's computer whenever the user visits the malicious web page. These fraudsters also convince users to call a premium telephone number, which not only results in a financial loss but also disclosure of their financial information to fraudsters. Another type of fraud that is popular over the Internet is Advanced Fee Fraud (AFF). This fraud is committed by asking victims to pay some amount to process their incentive, which can be in the form of leftover money of a deceased Nigerian rich person, an offer of a job with high pay, and the lucky win of being selected for a holiday vacation. The common attribute of these frauds is that the victim must pay a small amount of money first before they can get a larger amount from the attacker later. The Internet of Things devices has also been used to target the end-users for malicious activities, e.g. DDos and massive spamming.

3.4.1 Economics of Card Payments Frauds

Payment card fraud is an international issue that spans across nations, states, and borders. Fraud from overpayment cards has amounted to a total of $22.80 billion globally in the year 2016 [2]. This is 4.4% increase in the global card payment fraud rate as compared to the year 2015, where it was recorded $21.79 billion [2]. The United States (US) alone accounts for an overall of two-fifths (38.7%) of the global card payment fraud totalling to $8.45bn for the year 2016, and it is estimated that by the year 2020 the US card payment fraud could surpass $28*bn* [3]. On the other side of the Atlantic, card fraud losses for Europe in 2016 reached $2.12bn coming most of them (73%) from the United Kingdom (UK) and France [62].

Recently, it has been established that the costs associated with the losses on financial systems constituted the largest single category of fraud across the globe and over the Internet [62]. Thus, fighting international and organised card payment fraud has become part of the list

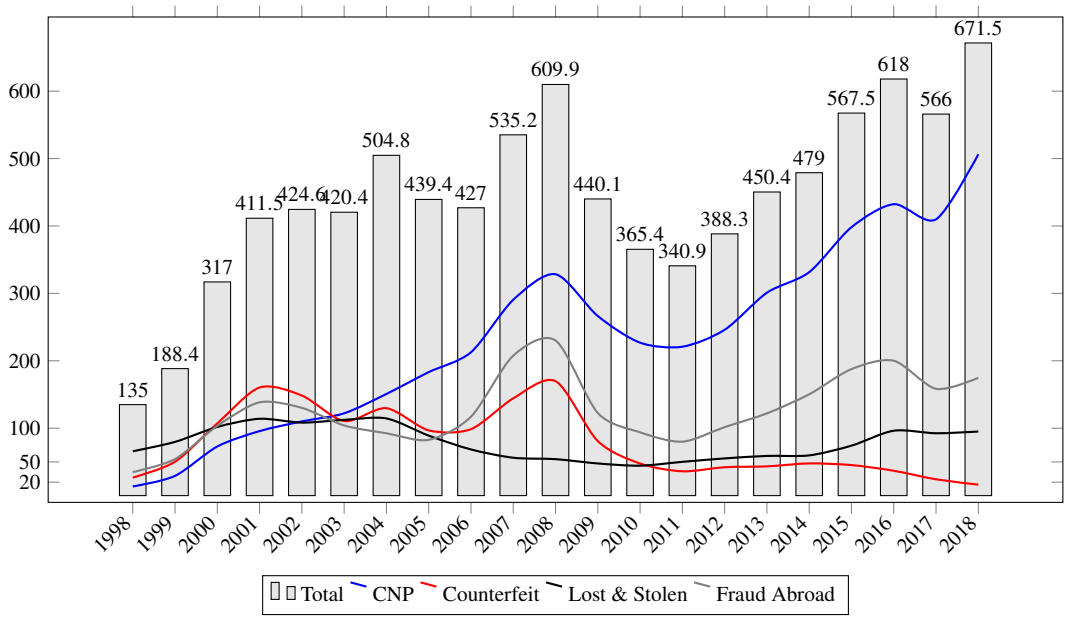


Figure 3.3 UK Card fraud by type from 1998 to 2018.

of the most serious priorities for Europe, and it is under Europol's priority crime areas (2018-2021 EU Policy Cycle [37]).

In most of the cases, fraudsters make a profit over electronic payment systems targeting the weakness in the system technologies. The methods vary depending on the type of the system (among card-present and CNP) being targeted, and they can be better understood by mapping the payment card fraud patterns over the evolution/improvements of card payment technologies. We are going to focus on card payment fraud patterns in the UK.

Figure 3.3 shows UK card fraud statistics from 1998 to 2018 [65]. In the figure, red and black lines represent losses on the two main card-present payment types of fraud, and the blue line signifies the fraud that occurred over CNP payment interfaces. The grey line represents the percentage of fraud that takes place abroad.

After the implementation of Chip and PIN in 2004 replacing earlier magnetic stripe cards for card-present transactions, fraudsters moved their activities from card-present to CNP fraud. Thus, tosses because of counterfeit fraud, after a peak in 2008 because of the delay in the deployment of CHIN and PIN technology abroad, decrease year by year until nowadays. On the other hand, fraud because lost and stolen cards after ann initial decrement has kept almost the same losses over all the period.

We can see losses on CNP interfaces double in only four years, from 150 million pounds in 2004 to 328 million pounds in 2008. In 2008, the deployment of preventive measures such as transaction risk profiling and authentication processes such as MasterCard SecureCode and Verified by Visa [63], lead to a decrement of 30% in fraud losses between 2008 and 2011.

However, nowadays, the situation is very different. During the last few years, a new scenario has been created, and it has brought new opportunities to fraudsters. Two facts have a lot to do with the new circumstances. The use of mobile devices for online purchases such as mobile phones and tablets; and the massive use of social media [96]. Mobile devices present new vulnerabilities, and the widespread use of social media is another channel where fraudsters can obtain private information to commit fraud. As a result, CNP fraud losses grew drastically from 2011 to nowadays, and the amount was beyond 500 million pounds in 2018.

3.4.2 CNP Payments - Technology, Limitations and Attacker Methods

The ease and convenience with which a customer can make purchases over the Internet benefited both the customer and the merchants alike. Within the CNP payment system we have *authorisation-only* and *user-authentication* enabled CNP payment protocols. Authorisation-only CNP protocols provided convenience to the shopping process where customers were only required to fill and submit their payment card details which include 16 digit card number, card's expiry date, three-digit card security code (CVV2) and cardholder address information to the Internet-based merchants. For fraudsters, however, this convenience came as an opportunity to steal customer's card details and misuse them.

The first attempt to combat growing CNP payment fraud came in the year 2001, where payment networks (Visa, MasterCard, American Express, et al.) introduced the 3 Domain Secure (3DS) protocol. It introduced the concept of user-authentication for payment transactions over the Internet. For every CNP payment transaction, 3DS required the customers to provide a password, thus combating the growing CNP payment fraud. However, the 3DS protocol exhibited two design flaws: activation during shopping and the use of static passwords. Activation during shopping required the cardholders to register with 3DS during the time of purchase. This enabled even attackers with stolen card details to register the victim's card over the 3DS. Additionally, attackers were still able to trick victims into giving away their static 3DS password. Because of these reasons, most merchants still stay hesitant to adopt the 3DS and prefer using the authorisation-only CNP payment protocol. This freedom of

choice for the merchants (i.e., the use of user authentication and/or authorisation-only), in the ways to accept online payments, even left pathways for the attacker to exploit loopholes and practice fraud over the CNP payment system.

The most common techniques employed by fraudsters to abuse the CNP payment include *phishing* and targeting the victim's machines with specially crafted *malware*, which is designed to steal payment card details. Stolen card details are either used by adversaries or are traded on online portals. Since phishing [46][54], card details stealing malware[149][56][72] and trading of card details in underground forums [175][174] have been comprehensively studied, we do not expand these attacker techniques in this thesis.

3.5 M-commerce Fraud

Mobile commerce is the activity of buying and selling products or services through mobile devices connected wirelessly, such as cell phones and personal digital assistants (PDAs). M-commerce allows the user to shop from any place.

Currently, consumers are increasingly turning to the Internet and mobile devices in their buying habits. M-commerce is fast growing. Mobile devices are predicted to account for more than 40% of all online retail sales in 2019, [85] . It has been predicted that the global mobile-payments market will grow by more than 33% through the year 2022 to reach $\$3,388$ billion [142].

Mobile payments have been adopted in different ways. We are using mobile devices for online shopping and paying for digital services. Moreover, they have become popular for contactless payments instead of paying with debit or credit cards. The main models for mobile payments usually are relayed in one of the following technologies [38]:

- Stored value account systems: Usually, the method is integrated into an app on the mobile device, i.e., payment wallet. Apple Pay, Samsung Pay, Android Pay, Microsoft Wallet, and PayPal are the most widely used wallets, with Paypal being the only one that works across different operating systems. They allow a customer to make faster online payments (when the merchant accepts them) and contactless transactions using the Near Field Communication technology included in many mobile devices. Other popular wallet apps are brand specific, such as Boost Mobile and the Starbucks Wallet app, which usually include loyalty programs.

- Account-based systems: A mobile web payment system can store card details that can be remembered for future purchases turning the payment into a simple click-to-buy. Commonly, strong authentication is required to commit large-value payments. Banks have taken advantage of this technology, and they have developed applications that allow customers to operate in their accounts in real-time, i.e., direct transfers.
- Mobile billing systems: The customer uses a premium SMS or direct carrier billing during the checkout. The success of earlier mobile content services, such as logos and ringtones introduced consumers to use this type of payment. An advantage of this type of payment is that existing telecom operator billing systems are suitable for handling micropayments transactions.

As we have seen in the previous section, the use of mobile devices for online purchases such as mobile phones and tablets is one of the main reasons why CNP fraud stands out to be the single largest category of fraud in electronic transactions, amounting to a total of 70% of the total card fraud for the year 2016 [65]. Mobile web fraud strategies are quite similar to those used on traditional online fraud, making the adaptation of cyber frauds to the mobile situation often straightforward. Furthermore, fraudsters have found new chances to make a profit specifically for mobile applications.

Some of the characteristics that have helped to increase the popularity of mobile devices such as ease of use and mobility create new security risks not associated with computers. It is common that mobile devices are often shared with friends and family, and it is potentially easier to leave them unattended in public spaces where they can be used for a second person or easily stolen.

A common way to steal payment information is through malware, which has been installed previously in the device. Other ways are social engineering and fake apps [12, 80]. Because of the low prices of mobile devices, fraudsters can afford using many different devices to commit the attack, and most of the observed fraudulent e-commerce transactions are originated from new devices[28]. By using new devices, fraudsters can avoid some of the traditional anti-fraud measures such as those based on a persistent identification which, i.e., the merchant can identify that is the same device trying to get access to a different account.

On the other hand, because of the low prices of mobile devices, fraudsters can afford using many different devices to commit the attack, and most of the observed fraudulent e-commerce transactions are originated from devices that are ‘new’ [28]. By operating in this manner,

fraudsters can avoid some of the traditional anti-fraud measures, such as those based on persistent identification.

As we have mentioned, one step authentication processes are easier to hijack, especially because users many times utilise the same access info for different services. Combining several types of measures makes the device less likely to be compromised.

In this scene, Machine Learning (ML) techniques begin to play an important role. The investment in ML-based methods for identifying and mitigating fraud has grown 13% since 2015 [24]. ML algorithms are employed to analyse user behavioural and transactional data, helping to detect anomaly patterns which can be correlated with fraud. ML algorithms have proved to be very efficient, but their use of mobile devices is limited to the computational resources available. For this reason, some authors propose a distributed approach in which some of the computational burdens are offloaded to the cloud [115].

Supervised techniques such as Support Vector Machine (SVM) and Hidden Markov Model (HMM) [169, 176] have been actively studied. In [106], the authors compared both techniques to identify users based on the way they walk. The authors showed that SVM was slightly superior to HMM with Equal Error Rate (ERR) of around 10%. In the same context, researchers in [107] showed that the K-Nearest Neighbours technique achieved slightly better results than HMM and SVM. These techniques as well have been studied in other contexts such as touch recognition [131, 34, 97], use of the software [41] and malware detection [147, 98, 168].

On supervised approaches, the model is trained with normal and fraudulent samples, which limit the operation of the system when the fraudulent patterns change. One-class SVM is a semi-supervised algorithm that learns a decision function to classify new data as similar or different to the training set, which only includes normal samples. In [21], the authors introduced a multi-modal approach that employed accelerometer and gyroscope data together with touch biometrics. First, the authors use a one-class SVM model to classify the samples as either belonging to the owner or another person. The decision is made based on a group of samples instead of a single observation. Subsequently, they build a dataset based on the previous classification process to train a two-class SVM i.e., the owner or the intruder. With some similarities, more recently, a semi-supervised approach was proposed [139]. The authors use the same sources of data and a classification algorithm based on the one-class SVM method. They test the approach with a dataset collected in a controlled environment where users were asked to type text during sitting and walking. They obtained an ERR

of 7.16% when the user was walking and 10.05% when the user was sitting. The authors deferred the evaluation of the approach to real-world scenarios for future studies.

More recently, other semi-supervised approaches have been proposed based on Deep Learning techniques such as Convolutional and Recurrent Neural Networks. In [104, 116], the authors used CNN to identify smartphone users in the way they hold the device.

3.5.1 Attack Methods in Mobile Payments

Many types of fraud affect the end-user of mobile devices. We describe the group with a higher incidence.

Account Takeover

Account takeover is the most frequent type of fraud [66]. After fraudsters have found out about the access information of a user, they utilise it to sign up for an expensive service or purchase a product. Bad actors manage to access personally identifiable information in many different ways, such as data breaches, which become more and more frequent, i.e., between May and July of 2017, Equifax, a large credit bureau in the U.S, was a victim of a data breach. In this case, personal information of almost 150 million of customers was exposed, including in a few cases credit card data [125].

Once an account has been taken over, it is difficult to fight because both legitimate and fraudulent users use the correct login credentials. Customers are particularly vulnerable when they do not use strong passwords, and they re-use them for several accounts [76]—increasing the exposure of customers when providers utilise one-factor authentication methods. For example, after the coffee chain, Starbucks launched an app which allowed customers to pay for their coffee, several customers reported that money was withdrawn from their accounts without authorisation [114]. After fraudsters managed to log into the app, they top up the Account using the stored credit card, and then they purchased gift cards that can be sold in the black market. The company said that criminals were obtaining login credentials from hacked websites and trying them out in the Starbucks app.

Phishing

Phishing is a well-known cyberattack where fraudsters steal personal information from users under false pretence by email, phone call, or social media sites. It is one of the oldest types of cyber fraud attacks, but still, it is frequently used in mobile channels, i.e., mobile users are 18

times more likely to be exposed to a phishing attempt than to malware [86], and three times more vulnerable to a phishing attack than to computers. While many users have learned to be suspicious of links and attachments in emails, however, today 66% of emails are checked on the mobile devices [145]. Similarly, the popularity of mobile applications like SMS and WhatsApp also attracted the fraudsters to utilise the medium for getting personal information of the victim via the phishing attack.

Fake Applications

Scammers develop fake apps that may include malware or are designed to steal personal info. Sometimes phantom applications use an organisation brand without permission to trick the user easily. Financial Trojan horse malware is one of the most popular cases because of the increasing availability of malware-as-a-service kits available in the cyber underground [28]. In some cases, the fake application sends premium SMS messages where an amount of money charged to the phone bill of the user goes directly to the fraudsters. Several researchers demonstrated the use of fake Near Field Communication (NFC) reader application on android enabled platforms. NFC enabled mobile phones to use ISO 14443 Identification cards – Contactless integrated circuit cards – Proximity cards (part 1-4) communication standards, and these are the same standards as used by contactless payment cards and readers to facilitate payments.

Mehrnezad et al. in [102] demonstrated the practicality of fake NFC applications initiating a fraudulent transaction with contactless payment cards. In that, the researchers were able to design a fake NFC app on an Android phone which can interact with the contactless cards kept in a mobile phone wallet and make fraudulent transactions, read user locations and upload these to an attacker-controlled server. In fact, in a single Google play search, we found 38 such NFC android mobile applications capable of reading contactless payment cards.

Fraudulent Website

A large number of fake websites will use a domain name that impersonates or refer to a well-known brand. But this would not represent the official website. For example, you apply for the job online, and they ask to deposit funds to process your application. This type of fraud shows the same characteristics of the computer case. However, in mobile devices, it is more successful because users are less likely to notice that a website is slightly different from the original [23]. This is because the size of the screen of mobile devices is relatively

small, constraining the user interface. This makes it considerably more difficult for users to recognise which mobile application or website they are interacting with [60].

3.5.2 Anti-fraud card payments measures on m-commerce

The trade-off of usability-security is the main concern when implementing anti-fraud measures. Users do not want their online experience to be affected by security steps. At the same time, merchants know that if they are not able to provide a smooth setting, they will have to deal with the user disappointment and economic loss.

On the other hand, because of the specific hardware specifications of mobile devices, the adopted measures must meet certain intrinsic aspects of the platform:

- lightweight: The computational demand of the system has to be low since mobile computational power is limited.
- restrict the communications: some mobile device users, i.e., smartphone users, can be charged by data rates. Additionally, bandwidth or data usage may be limited. Therefore, the amount of information sent and received for the security approach should be small.
- restrict energy consumption: battery life is nowadays a big concern in mobile technology. It is required that the energy consumption level is as much efficient as possible.

Transaction risk profiling

The card payment industry is continually seeking measures to combat fraud activities. However, where a new protective procedure is introduced, fraudsters adapt to it. One of the measures conduct by issuers and acquired during the last 20 years has been modelling transactions to develop tools to detect fraudulent activities.

However, because merchants do not process the same amount of transactions as issuers and acquire, often they opt to model the user behaviour, i.e., the way users interact with the web browser.

Control access: authentication mechanisms

An authentication mechanism is a common measure to enhance the security of mobile devices. In an authentication process, the identity of the user is verified according to different sources

of information provided, directly or indirectly, by the user. We can classify authentication methods to the following:

- Knowledge-based methods: the process is based on information that the user knows, i.e., a password or a Personal Identification Number (PIN).
- Object-based methods: the process is based on something the user possesses, i.e., a hardware token.
- Biometric-based methods: the process is based on information obtained, usually from sensors. This information describes the physical or behavioural characteristics of the users, such as the tone, cadence, and pitch of their voice.

Knowledge-based methods such as PIN and passwords have been used for mobile phone authentication since the inception of this technology in the market more than twenty years ago, and they are still widely used, despite their intrinsic weaknesses have been largely demonstrated. Furthermore, PIN/Passwords are intrusive techniques that require a specific action of the user, and they take place only once at the beginning of the session.

Biometrics based methods have been introduced more recently, and they are receiving much attention from the community. Biometric data describe the physical or behavioural characteristics of a human being. Different sources will include different attributes, such as features that describe a voice pattern and motion patterns.

A Biometric Authentication System (BAS) evaluates biometric data for verification or identification of individuals. Nowadays, many different sensors have been incorporated into the smartphone, such as environmental, location, and motion sensors. Obtaining biometric data from them is easy and straightforward, and BAS have been used in many practical applications successfully.

Biometrics-based methods are considered more reliable and secure [159], and the recent embedding of many new sensors in mobile devices has to lead to the development of many different approaches such as those based in the face, iris, periocular and fingerprint recognition [154, 126, 136].

3.5.3 Traditional authentication methods on Smartphones

Because of the easy portability of mobile devices, access control measures are necessary. An authentication mechanism is a common measure to prevent unauthorised access to the device.

In an authentication process, the identity of the user is verified according to the information provided either directly or indirectly by the user.

In [90], researchers introduced an object-based approach using a Bluetooth token. In this approach, when the user attempts to gain access to the device, a smartphone tries to communicate with a token through Bluetooth connection. If the token can be reached and the smartphone receives confirmation of the communication, it will be unlocked. However, object-based authentication methods have rarely been implemented for mobile devices. Nowadays, people bring their devices most of the time with them, and the obligation to carry on an authentication device makes these systems less practical.

Knowledge-based methods have been used for a long time on mobile authentication processes, and they are still used widely. PIN and password have been for years the most common authentication approach, despite that their inconvenience and weakness have been proved many times. This method is susceptible to simple attacks, i.e., shoulder surfing where fraudsters spy the actions of victims and smudge attacks where smudge stains on the display are used to infer the password [43]. On the other hand, because the user has to remember the code, often they either use the same memorable instance for different accounts, or they choose a weak one. A study has shown that among over $6,000,000$ passwords, 91% of all of them belong to a list of just $1,000$. The same study points out that in this list, 8.5% of the individuals use either “password” or “123456” as a password [61]. Furthermore, passwords have been reported stolen from big databases on many occasions.

Some approaches have tried to overcome some of the limitations of PINs and passwords. In [53], researchers introduce a graphical authentication system that attempts to confuse an observer with different information each time the user tries to log in. To make the system more manageable, users have to remember a group of images instead of codes, and they will have to select the images they know among those revealed. The study showed an increment of the time to log in and lower the success rate. In [43] the researchers attempted to address the threat of the shoulder surfing attack. In this case, the user should draw a shape in the back of the device, the area which should be more difficult for someone over the shoulder to watch. The system was developed in a prototype because the additional hardware required is not available in any smartphone in the market. PIN and password have been for years the predominant authentication approach on mobile phones. Although the inconvenience and weakness of these authentication methods have been pointed in numerous instances, there are still widely used on smartphones, sometimes with variations such as the grid unlock, which consists of drawing a lock pattern on the screen.

These methods are intrusive techniques that require an explicit action of the user. Since the user must remember the authentication key, a weak one is usually chosen, and often the same key is utilised for different purposes. A study showed that over 6,000,000 passwords, 91% of them belong to a list of just 1,000. The same study shows that in this list, 8.5% of the individuals use either "password" or "123456" as the password [118]. Additionally, passwords have been reported stolen from big databases in many instances. Such approaches are one-time authentication methods, which are activated once at the beginning of the session. For added security, it would be desirable to have a continuous validation of the user identity. Furthermore, traditional methods are susceptible to simple attacks like shoulder surfing, where a fraudster spies the actions of victims, or smudge attacks, where smudge stains on the display are used to infer the password pattern [43].

3.6 Biometric authentication

Biometric data describe the physical and/or behavioural characteristics of a smartphone user, and each acquisition consists of a group of features that describe a biometric pattern such as voice or motion. A Biometric Authentication System (BAS) captures and processes data for identification of allowed individuals on a smartphone. Differently from a knowledge-based method, which requires active actions from the users, BAS is an automatic identification process.

Nowadays, many different sensors that can capture environmental, locational, and user-specific motion information have been incorporated in the smartphone. Hence, obtaining biometric data from a modern smartphone is easy and straightforward. BAS are believed to be trustworthy, and their applicability in smartphone platforms is receiving increased considerably over recent years.

3.6.1 General scheme of a biometric authentication system

The operation of a biometric authentication approach comprises of two phases, as shown in Fig. 3.6. In the first phase, referred to as the enrolment phase, biometric instances are captured, and a feature vector is extracted and stored. The feature vectors are meaningful representations of the data and represent individual-specific patterns. Verification systems store patterns from a unique user, while identification systems from a group of individuals. Lately, many authentication systems have been based on Machine Learning methods,

i.e., semi-supervised learning approaches for verification systems and supervised learning approaches for identification systems.

In the second phase of the BAS, referred to as the recognition phase, a new instance from an authentication request is provided to the system. After the same pre-process and feature extraction mechanisms, as in the enrolment phase, the feature vector is transferred to the matching algorithm, which performs two operations. Firstly, in the matcher module, the feature vector from the authentication request is compared with those stored in the enrolment phase. The result of the comparison will be a match score that is assigned to the authentication request instance. The second operation is performed in the decision module, which classifies the instance comparing the match score calculated to a Decision Threshold (DT). Here, it is expected that instances belonging to different individuals will have very different score value distributions, as exemplified in Fig. 3.4. If the score is higher than the DT, the instance of the authentication request will be considered legitimate.

Some BASs take into account more than one source of biometric data. Multimodal biometric authentication systems have been proposed as a means to mitigate some problems of unimodal systems such as noisy data, spoof attacks, low detection rates, and high false-positive rate. Data from different sources can be combined at different levels, and the authentication from the combined information is typically performed in a different module of the biometric scheme. However, capturing and processing more data will increase the computational burden of the approach [130].

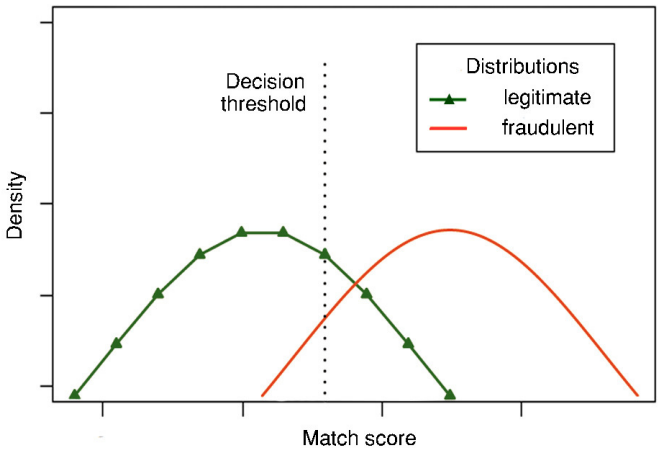


Figure 3.4 An example of the decision threshold for match score distributions from two different users, one legitimate and one fraudulent.

3.6.2 Biometric functions

Biometric systems pretend to recognise individuals. There are two main functions in the process:

- Enrolment function: it generates the biometric reference for a person from the biometric characteristics and saving it for further comparisons.
- Recognition function: it recognises users. It exits two methods: verification and identification. In the verification process, the system checks if the person is who the user claim to be. In the identification process, the system will indicate who the user is.

The norm ISO/IEC JTC 1/SC 37 Biometrics define five parts in a general biometric system [1] :

- Data Capture subsystem: it is in charge of capture the biometric sample and converts it in digital format.
- Data Storage subsystem: it stores the biometric samples.
- Signal processing subsystem: it generates a features vector from the biometric sample.
- Comparison subsystem: it compares the sample with a model.
- Decision subsystem: it takes the decision based on the decision threshold.

Figure 3.5 shows the diagram of this system. Some authors combine the Comparison subsystem and the Decision subsystem in the same module.

3.6.3 Multimodal Information Fusion

The information fusion level on multimodal systems depends on the module where the biometric information is combined. We identify four data fusion levels. We differentiate each of the four fusion levels depending on which of these four subsystems take place: in the Data Capture subsystem, in the Data Storage subsystem, in the Signal Processing subsystem, or in the Comparison subsystem.

- Sensor/Data level: we can merge the sensor data directly when sensor signals are comparable. There are three fusion paradigms:

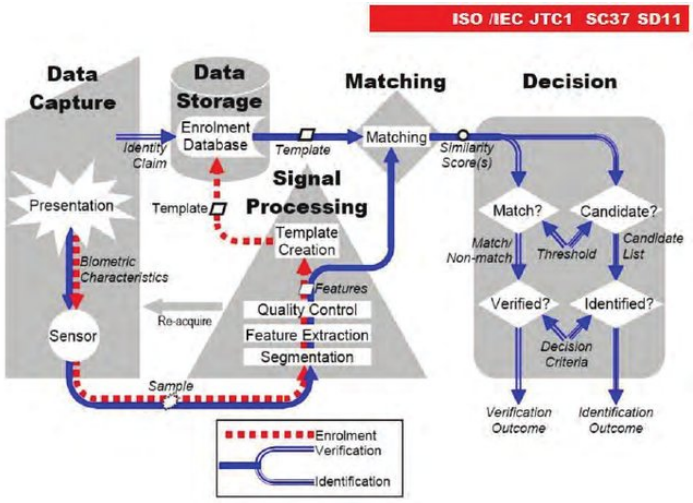


Figure 3.5 General Biometric System - ISO/IEC JTC 1/SC 37 Biometrics .

- The complementary data fusion paradigm: each sensor provides independent information about an aspect of an object. All the information is combined to obtain more detailed information about the object.
- The competitive data fusion paradigm: each sensor provides different independent information about the same aspect of an object. The system decides which sensor most correctly represents the aspect.
- The cooperative data fusion paradigm: each sensor provides different independent information about the same aspect of an object. Combining this information, we will retrieve the knowledge which could not be obtained separately from the information of the individual sensors.

3.6.4 Biometric authentication continuous systems

Nowadays, many different sensors that can capture biometric data such as environmental, locational, and user-specific motion information have been incorporated in the smartphone, and the capture process is easy and straightforward. BAS are believed to be trustworthy, and their applicability in smartphone platforms is receiving increased considerably over recent years.

The practical implementation of BASs is hampered by engineering challenges in the data acquisition process, as well as methodological challenges in the development of efficient machine learning algorithms that could achieve a satisfactory effectiveness rate.

Similarly to knowledge-based methods, BASs such as face and voice recognition are one-time authentication systems: the user is validated only at the beginning of the session. BASs based on information such as motion or location can be, however, designed as a continuous authentication system, constantly verifying the user-identity throughout the whole session. The re-authentication is performed with a given frequency depending on the system capability. Continuous BASs should not be intrusive and transparent and do not require any attention or action from users.

Continuous authentication [42], can be used either as a primary authentication method or an auxiliary fraud indicator for higher assurance [83]. In typical use cases, continuous authentication adds extra reliability to the system and improves usability. For example, when the authentication system expresses continued confidence in the identity of the user, a service provider may decide to skip further security queries, i.e., not requiring extra info to complete a new transaction.

3.6.5 Identification vs Verification

Biometrics-based access management and security systems can operate in two different modes:

- identification (1-to-n): We take into account the information from the individual, and we compare his/her biometrics to a database of possible identities to match them and determine his/her identity. For example, it is how law enforcement and border control use biometrics – running a fingerprint against a database.
- verification (1-to-1): It is the process of proving your identity. The user will be claiming the identity of someone already known to the system, and we need to verify that it is true. For example, when an individual uses his/her fingerprint to unlock a smartphone, we are verifying that it is the same fingerprint that was previously scanned.

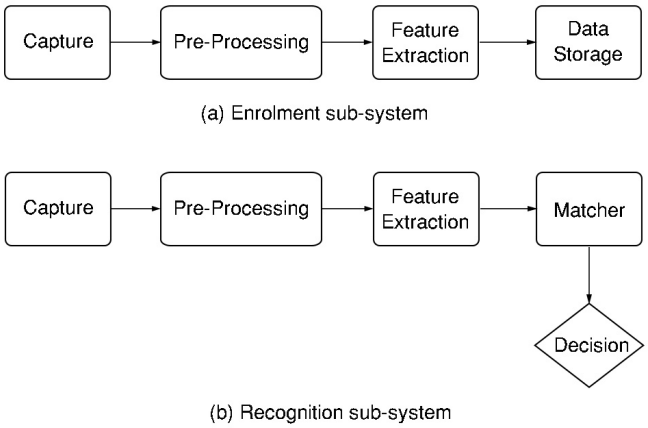


Figure 3.6 General authentication biometric scheme. **(a)** Denotes the enrolment phase i.e. the training of the algorithm. **(b)** Denotes the recognition phase, i.e. the decision rule.

Chapter 4

Motion-based identification on Smartphones

4.1 Introduction

The smartphone has become an increasingly popular device in both emerging and developed countries, and over a third of the world's population is projected to own one by this year [144]. These devices are currently used for an endless list of purposes, such as paying public transport, accessing corporate data emails, performing banking transactions and accessing social media accounts. Mobile phone apps contribute to this trend facilitating interaction with the device, and a large number of companies offer services through them [170].

While performing all these activities, critical and sensitive information is stored in the device. The characteristics which have contributed to increasing the popularity of smartphones, such as portability and ease of use, also introduce critical security vulnerabilities [93]. Fraudsters have caught the wave of opportunity, and they have been shifting their activities to this channel [122], and this prompts the community and industry to continually search for the best trade-off between security and usability to prevent unauthorised use.

Identification is a procedure where users' identity is determined. This strategy allows for granting different privileges for each user. It can be useful in many cases, i.e., avoiding that the youngest of the family buy game applications from the father's smartphone. In this chapter, we consider a scenario where a different person has physical access to a smartphone, such as family members or co-workers attempting to access the device or associated resources.

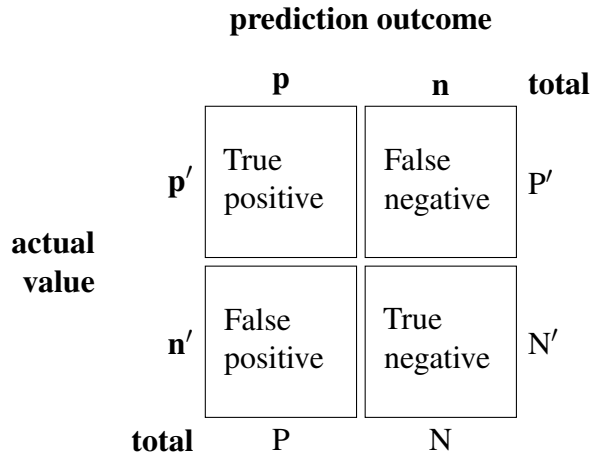


Figure 4.1 Confusion matrix

We propose a biometric identification system based on motion data, i.e., accelerometer, gyroscope and magnetometer data.

This chapter is organised as follows. Section 4.2 introduces the metrics used to show the effectiveness and to compare the performance of different approaches. Section 4.3 reviews identification approaches for electronic devices and those specific to smartphones. Section 4.4 describes the dataset used to show the results of this study and introduces several pre-processing techniques which we have applied. Section 4.8 shows the experiments results of identifying users based on motion data and Section 4.9 concludes with a discussion.

4.2 Effectiveness metrics

A common option to measure the effectiveness of binary classification models is the confusion matrix shown in Fig. 4.1 [150]. The confusion matrix shows for each class the number of samples classified correctly and the number of them misclassified.

The goal of an identification system maximises the number of true positives (TPs) when keeping the number of false negatives (FPs) low.

Some common measures derived from the confusion matrix are True Positive Rate (TPR) eq. 4.1, True Negative Rate (TNR) eq. 4.2 and False Positive Rate (FPR) eq. 4.3. TPR, as well as call sensitivity or recall, measures the proportion of positives cases (i.e., which are correctly identified. On the other hand, TNR, as well as call specificity, measures the proportion of

negative cases (legitimate transactions) that are correctly identified. Finally, FPR measure the proportion of false alarm cases.

$$TPR = \frac{TP}{TP+FN} \tag{4.1}$$

$$TNR = \frac{TN}{TN+FP} \tag{4.2}$$

$$FPR = \frac{FP}{FP+TN} = 1 - TNR \tag{4.3}$$

We are going to introduce a set of well-established performance metrics which will be used throughout this work to evaluate the accuracy and precision of the proposed biometric identification system, and to compare its performance with previous studies:

- Accuracy is the ratio of correctly predicted observation of the total observations. Accuracy is not a good measure for imbalanced datasets, i.e., the different ratios of samples from each class.
$$Accuracy(ACC) = \frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$$
- Precision (also known as specificity) is the ratio of correctly predicted positive observations of the total predicted positive observations. In our context, precision is the fraction of the fraudulent predictive instances which have been classified correctly.
$$Precision = \frac{\sum \text{True positive}}{\sum \text{Predictive condition positive}}$$
- Recall (also known as sensitivity) is the ratio of correctly predicted positive observations to all observations in the actual class. In our context, it is the fraction of fraudulent instances that have been detected.
$$Recall = \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$$
- F_1-score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account and works better than accuracy with imbalanced datasets. However, ff the cost of false positives and false negatives are very different, Precision and Recall will be more informative.
$$F_1 score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

- The false acceptance rate (FAR) is the measure of the likelihood that a biometric security system accepts an access attempt by an unauthorised user, i.e., the ratio of the number of false acceptances attempts divided by the number of identification attempts. On the other hand, the false recognition rate (FRR) is the measure of the likelihood that a biometric security system rejects an access attempt by an authorised user, i.e., the ratio of the number of false recognitions attempts divided by the number of identification attempts. In our context. FAR is the rate of fraudulent instances classify as legitimate and FRR the rate of legitimate instances classify as fraudulent.

$$FAR = \frac{\sum false \text{ negative}}{\sum false \text{ negative} + \sum true \text{ positive}}$$
$$FRR = \frac{\sum false \text{ positive}}{\sum false \text{ positive} + \sum true \text{ negative}}$$

- Equal Error Rate (ERR) is the percentage when the FAR and FRR are equal, i.e., the number of false acceptances attempts is equal to the number of false rejections attempts. The lower the ERR value, the higher the accuracy of the biometric system.

- Receiver operating characteristic curve (ROC curve) is a graphical plot that shows the goodness of a binary classifier. ROC curve plots the FAR against the true positive rate (TPR as well known as sensitivity or recall an equal to 1-FRR) when varying a discrimination threshold. In the ROC curve, the intersection between the curve with a line with intercept one and slope minus one indicates the ERR.

4.3 Identification approaches

Identification systems have been shown useful for multiple scenarios. Reference [112] developed a system for identifying people based on their footstep force profiles. They created user footstep models based on footstep profile features. They achieved an accuracy rate of 93%.

The authors of [6] proposed an identification system to distinguish between the passengers and the driver of a car. The system was proposed to offer a safer and more pleasant driven experience to the users. The approach relied on analysing the user smartphone motion data when entry to the vehicle, i.e., accelerometer, gyroscope and magnetometer data, and doors signals (if available). Experiments demonstrated the usefulness and effectiveness of the introduced probabilistic identification approach, which was based on a linear logistic

regression model. They showed an accuracy rate of over 90% when the doors signal was not available.

Authors of [91] developed an identification system for electronic devices based on fingerprint data. First, they extracted features of the image, i.e., the minutiae, and they stored them on a database. In the verification phase, they performed one-to-one matching between the input sample and the templates data previously stored in the database. The matching approach was based on the difference of the ridge orientation and the edit-distance between the template and the input samples. The approach obtained an ERR of 0.92%. Similarly, reference [162] presented a user identification system using finger-vein technology. They first conduct a feature extraction process using Radon transform and singular value decomposition (SVD). Later, they classify the feature vector using a normalised distance measure. The experimental results showed an ERR of 0.99%.

Reference [173] proposed an identification approach based on web browsing behaviour. Identifying web users is useful for a product recommendation, personalised advertising, etc. They experimented in different datasets, including a different number of users, i.e., 2, 5, 10, 20, 50, 75 and 100 and sliding windows including a different number of sessions, i.e., ranging from 1 to 100. They compared the performance of two approaches based on SVM and DT techniques. The results showed that the support-based profiling method significantly got better classification results when the size of the sliding window was larger and the number of users increased (87% accuracy rate with 100 users and sliding window size equal to 100). The DT approach performed better when the sliding window size was smaller (68% accuracy rate with 100 users, and a sliding window size equal to 1).

4.3.1 Smartphone user identification

Identification systems have been largely used in smartphones. Face recognition was widely used in banking and security access systems. With the integration of low-cost and high-quality cameras on smartphones, it has been investigated for smartphone identification [138]. In face recognition for identification, the system is trained with photos from all the users, and in the recognition phase, the approach matches the new sample with a known user. The process of image acquisition is not completely reliable as it is influenced by a number of external conditions such as the illumination or the clothes worn by the individual. One of the main threats of this method is face spoof attacks, which, in its simplest form, consist of misleading the system displays an image of the victim (print and replay attacks). Moreover,

given the widespread use of social media networks, it is now very easy to access to pictures and videos from potential victims. Measures to mitigate such attacks have been proposed, but an effective approach has not yet been developed [117].

Similarly to face recognition, iris identification uses the device's camera to capture biometric samples from the user's eye. Reference [22] introduced an approached which extracted a one-dimensional representation of the grey-level profiles of the iris from each picture.

Another BAS which has been extendedly studied on smartphones is touchscreen recognition. In this method, which has also been proposed as a continuous authentication approach, the user touch to the screen is used as a means to authentication. In the authors, [10] collected data from 32 participants when they were drawing a pattern on the screen device to unlock it. They compared the performance of six algorithms based in the Euclidean distance, manhattan distance Mahalanobis distance, SVM and RF. The approach based on the RF algorithm obtained the best accuracy results with an ERR of 10.39%.

In [165] the authors investigate the usage of app signature to identify users. They conduct experiments in a dataset, including over 46000 participants. They found that when using an app signature-based in the 60 most used apps, they are able to differentiate correctly 99.4% of the users. Furthermore, they found that the average minimum Hamming distance to a different user was 25.93. They establish as a future work the calculation of the time required to be able to identify users.

In [71], the authors used accelerometer, gyroscope, and magnetometer data to identify smartphone users. They based their approach in an SVM algorithm to classify samples from two users at the time (from an original dataset including samples from 300 users). They show the accuracy results in two different groups. For one group, the ERR was around 0.9% and for the other group 0.6%. They conclude that for some users, it was not possible to find characteristic motion patterns.

In [7], the authors used a motion dataset, including samples from 10 participants for six different physical activities, i.e., walking, sitting, standing, running, walking upstairs, and walking downstairs. They collected samples for 3-5 minutes for each activity and participant. They tested the accuracy of three different machine learning algorithms, i.e., K-Nearest Neighbour (K-NN), Bayes Net (BN), and Support Vector Machine (SVM). BN and SVM obtained an equal average accuracy result of 94% and K-NN of 90%.

Reference [108] compares an approach using the same type of data. They used a dataset of 36 subjects and compared the performance of three algorithms, i.e., HMM, SVM, and KNN. KNN obtained the best accuracy result of ERR 8.24% (HMM and SVM obtained 8.75% and 8.85% respectively).

4.4 Dataset

In the experiments of this chapter, we are going to use a publicly available dataset of behavioural data on smartphones [172]. The dataset contains samples from 100 volunteers, which encompass multiple modalities including movement, pausality, orientation, touch and gesture. Data was collected under three task scenarios (reading, writing, and map navigation) and two body motion conditions (sitting and walking).

Each session lasts about 15 minutes, and each volunteer performs 24 sessions (8 reading sessions, 8 writing sessions, and 8 map navigation sessions). In total, each volunteer contributes about 6 hours of behavioural traits.

The following 9 categories of data were recorded:

1. Accelerometer: timestamp, acceleration force along $X/Y/Z$-axis. Sampling frequency 100 Hz.
2. Gyroscope: timestamp, rotation rate along $X/Y/Z$-axis. A sampling frequency of 100 Hz.
3. Magnetometer: timestamp, ambient magnetic field along $X/Y/Z$-axis. Sampling frequency 100 Hz.
4. Raw touch event: timestamp, finger count, finger ID, raw touch-type, X/Y coordinate, contact size, screen orientation.
5. Tap gesture: timestamp, tap type, raw touch-type, X/Y coordinate, contact size, screen orientation.
6. Scale gesture: timestamp, pinch type, time delta, X/Y focus, X/Y span, scale factor, screen orientation.
7. Scroll gesture: starting and current timestamp, X/Y coordinate, and contact size; speed along X/Y coordinate; screen orientation.

8. Fling gesture: starting and ending timestamp, X/Y coordinate, and contact size; speed along X/Y coordinate; screen orientation.

9. Keypress on the virtual keyboard: timestamp, press type, key ID, screen orientation.

Figure 4.2 shows samples from the accelerometer gyroscope and magnetometer sensors from a user reading a text when sitting. Each row includes samples for each of the directional axis of each of the sensors.

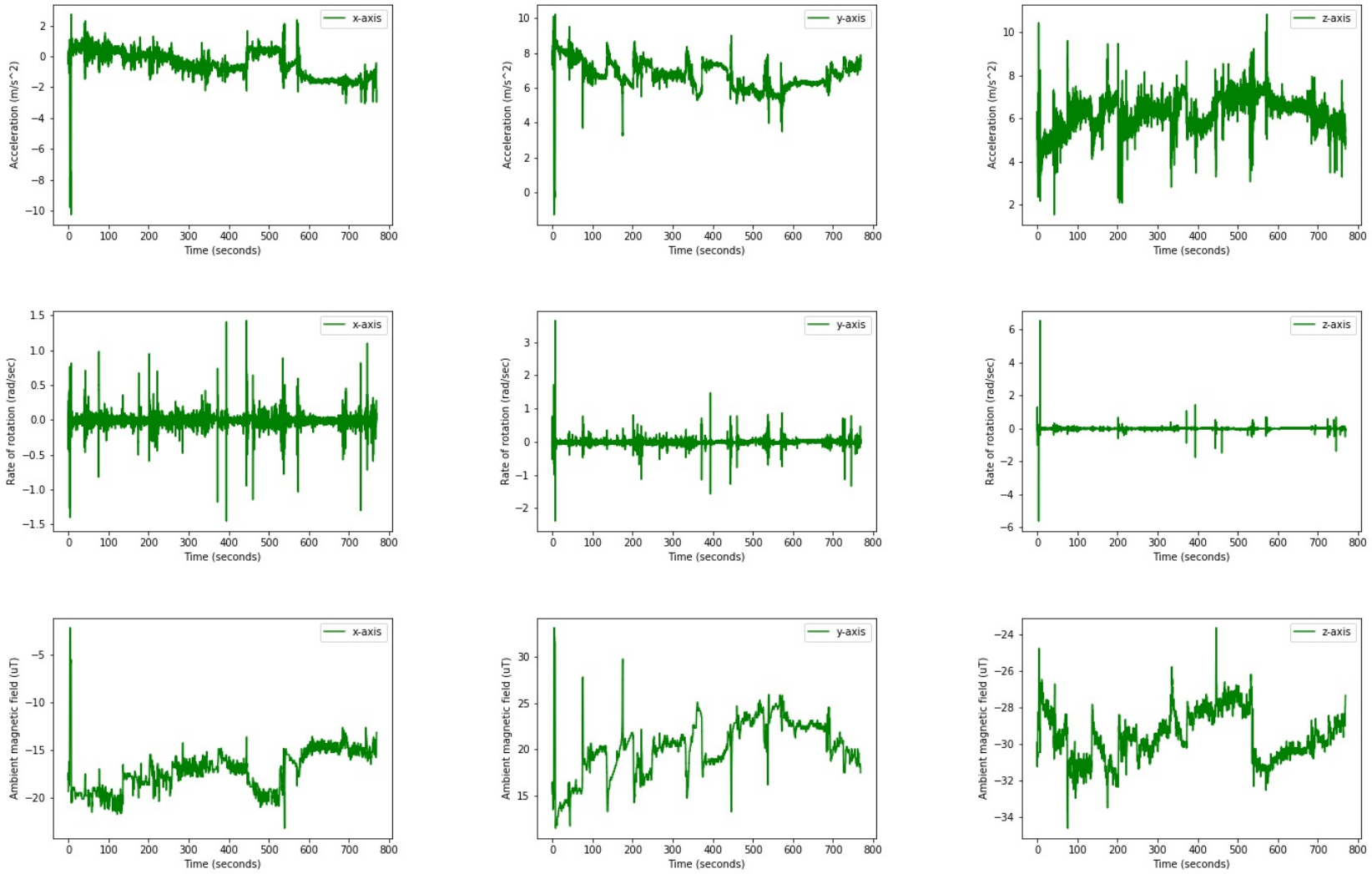


Figure 4.2 Accelerometer, gyroscope and magnetometer data along the x, y, and z axis. The data was captured when the user was reading a text when sitting.

Training and testing datasets

To show the performance of each approach, we have produced 100 datasets. Each dataset includes samples from two users, i.e., the legitimated and fraudulent user. For each dataset, both users are picked up randomly from the whole list of them. We test every identification approach in each dataset, and we show the average of the performance results.

We split each of the 100 datasets into a training and testing datasets. They include samples from two users (legitimate and fraudulent users). The training dataset is 54 minutes length, and the testing dataset 18 minutes length (with the same proportion for each of the users). Furthermore, each of the datasets includes the same proportion of samples from each of the task-condition cases defined (reading-sitting, reading-walking, writing-sitting, writing-walking, navigating-sitting, and navigating-walking) from each of the users. The training dataset includes samples from three sessions for each task-motion case (18 sessions in total) and the testing dataset from one session for each task-motion case. Note that samples in the testing dataset are from different sessions of those in the training dataset.

Thus, each of the training and testing datasets represents a scenario where known users utilise the device in different periods of time, and we want to know who is the user utilising the device on each moment. Training and testing datasets include samples for one user when performing the mentioned activities, followed by the same number of samples from another user performing the same activities.

4.5 Windowing

We split the stream of data sampled from the sensors in segments. We call instance to the vector which includes samples from the three space directional components for each of the sensors taking into account. We sort the elements of the segment concatenating the samples from each dimensional axis of each of the sensors consecutively. For example, an instance including samples from the accelerometer and gyroscope sensors, it has the form:

$$instance = \{a_{x1}, a_{x2}, ..., a_{xm}, a_{y1}, a_{y2}, ..., a_{ym}, a_{z1}, a_{z2}, ..., a_{zm}, g_{x1}, g_{x2}, ..., g_{xm}, g_{y1}, g_{y2}, ..., g_{ym}, g_{z1}, g_{z2}, ...g_{zm}\}$$

The longitude of the segment m and the sampling frequency of the sensor f_s determine the re-authentication time i.e. how often we are able to validate the user:

$$\text{reauthentication-time} = \frac{m}{f_s} \text{ seconds}$$

In the experiments, we are going to use two window sizes, i.e., a window including 500 samples and a window, including 50 samples.

4.6 Normalisation

It is common to apply normalisation before building a model. It is because:

1. Some data mining methods require the range of the input attributes lie between 0 and 1 [30].
2. In another case in certain algorithms, attributes with the highest range would contribute more to the modeling decision making process.

Two common approaches for normalisation are Standardisation and min-max scaling:

Standarisation

The result of standardisation (or Z-score normalisation) is that the features will be rescaled so that they will have the properties of a standard normal distribution with:

$$\mu = 0$$
$$\sigma = 1$$

where μ is the mean (average) and σ is the standard deviation from the mean; standard scores (also called z scores of the samples are calculated as follows:

$$z = \frac{x - \mu}{\sigma}$$

Min-max scaling

The result of min-max scaling (or normalisation) is that the features will be rescaled to a fixed range, usually 0 to 1. The cost of having this bounded range - in contrast to standardisation - is that we will end up with smaller standard deviations, which can suppress the effect of outliers. A Min-Max scaling is typically done via the following equation:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

We are going to show the performance results from each identification approach when standardising and scaling the data. To normalise each dataset, we calculate the parameters to

normalise the data, i.e., μ, σ, *min* and *max* from the training dataset, and we used them for normalising the training and testing dataset.

4.7 Sampling

Sampling is a statistical method by which one works with a subset of data significant enough to represent all of them. Sampling is a very common preprocessing process in data mining. There are several reasons for that:

1. On the one hand, usually datasets are pretty large, and the process of training in the whole dataset can be quite computationally expensive.

2. Furthermore, some algorithms load the entire training data sets into the main memory, so they cannot operate with large datasets because they cannot be allocated in the memory [146].

3. Sometimes datasets are imbalanced, i.e., there is a higher number of samples of some class. Then subsampling or oversampling can be used to balance the number of samples of each class.

We are going to conduct several experiments to show how it affects sampling when identifying users. We are going to test how the performance of the proposed models change when varying the sampling frequency of the motion sensors, i.e., accelerometer, gyroscope and magnetometer.

Practically, we are going to sample the dataset introduce in 4.4 to two different ratios of samples, i.e., 100 samples per second and 25 samples per second. Because the data originally have been recorded at 100 Hz, to get 25 Hz sampling frequency, we sample one reading every four. We will show the comparison for each of the approaches in the next section.

4.8 Experiments results

In this section, we analyse the potential of several supervised Machine methods to identify users based on sensor motion data. We train each model with data from two different users, and we test the ability of the model to identify each of the users.

Furthermore, we show the effect of using different sampling frequencies, window sizes, numbers of sensors taking into account and preprocessing techniques to scale the data (according to the values mentioned in the previous sections).

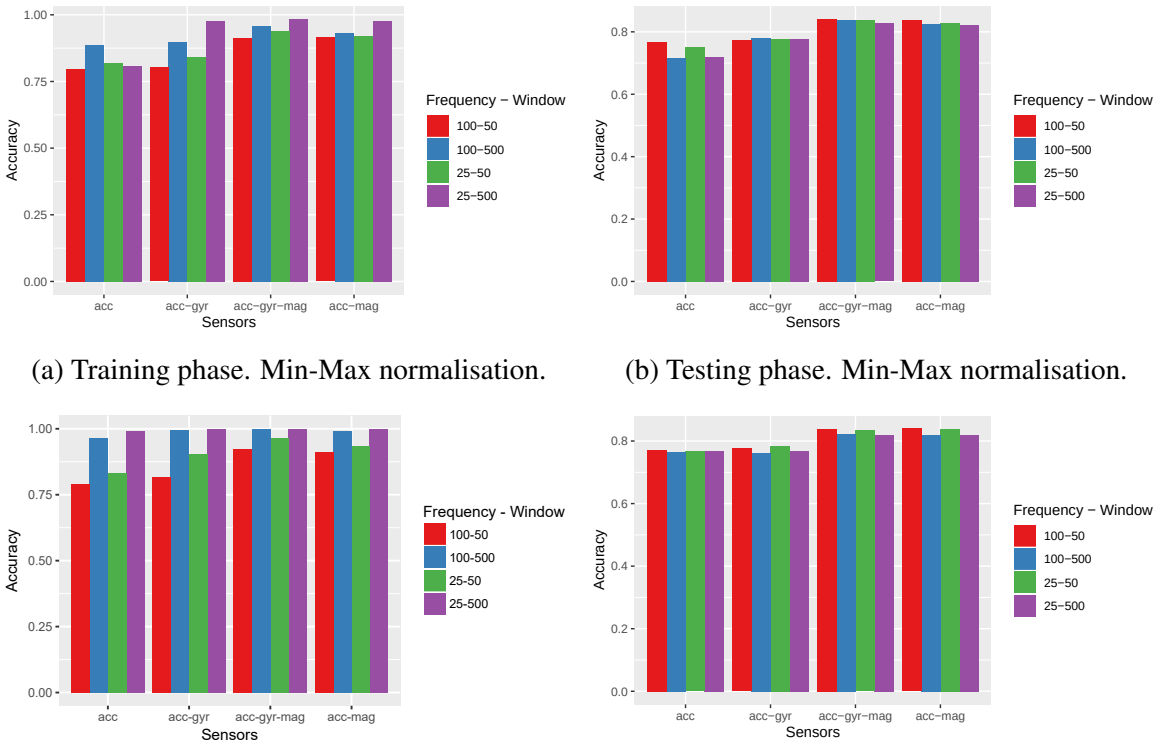
4.8.1 Logistic Regression

In this section we test the accuracy of the logistic regression model when we vary the regularisation strength parameter along $[0.1, 1, 3, 5, 10, 100, 300]$ (smaller values specify stronger regularisation).

Figure 4.3 shows the highest accuracy achieved between the models with different regularisation strength parameter, when sampling the data with different frequencies (100 and 25 Hz), using different window sizes (500 and 50 samples) and taken into account data from different sensors. Figures 4.3a and 4.3b show the results of the training and testing phases when the data has been normalised; and figures 4.3c and 4.3d, shows the prediction results of the training and testing phases when the data has been standardised.

We can observe that:

- As it was expected, accuracy results during the training are better than during the testing. We can observe a bit of overfitting during the training when the sampling frequency is 25 and window size 500, e.g., the training accuracy is much higher (97.74%) than testing accuracy 77.57% (when normalising in this example).
- We obtain better accuracy results standardising the data when we include data only from the accelerometer sensor. When we include data from more than one sensor, we obtain better accuracy results when normalising the data.
- During the testing, when standardising the data, reducing the window size from 500 to 50 improves the accuracy, whatever is the sampling frequency. This is less significant or null when normalising the data.
- Reducing the sampling frequency, decrease the accuracy. The decrement of accuracy is higher when the window size is larger.
- Taking into account data from more sensors increase the accuracy. In the testing phase, the highest accuracy result when taking into account three sensors is 84.15%, and taking into account only accelerometer data is 76.%. However, when taking into account data from the accelerometer and magnetometer sensors, the highest accuracy



(a) Training phase. Min-Max normalisation.

(b) Testing phase. Min-Max normalisation.

(c) Training phase. Standarisation.

(d) Testing phase. Standarisation.

Figure 4.3 Accuracy prediction results of the training and testing phases of the logistic regression model. Top row shows the results when the data has been normalised and the bottom row when the data has been standardised.

is 83.9%, almost the same as taking into account three sensors. When taken into account data from the accelerometer and gyroscope sensors, the highest accuracy score is 77.78%.

4.8.2 Multi Layer Perceptron

In this section, we test the accuracy of the multi-layer perceptron algorithm with $L2$ regularisation.

The model includes two layers. We have test models with a different number of input and hidden units for each layer when the window size is 500 we have test four models with the number of input units for each hidden layer of $1000-1000$, $1000-500$, $500-1000$ and $500-500$, when the window size is 50 we have test four models with the number of input and hidden units for each hidden layer of $50-50$, $50-25$, $25-50$ and $25-25$.

Figure 4.4 shows the highest accuracy achieved between the models with different architecture, when sampling the data with different frequencies (100 and 25 Hz), using different window sizes (500 and (50 samples) and taken into account data from different sensors. Figures 4.4a and 4.4b show the prediction results of the training and testing phases when the data has been normalised; and figures 4.4c and 4.4d show the prediction results of the training and testing phases when the data has been standardised.

We can observe in the testing phase results that:

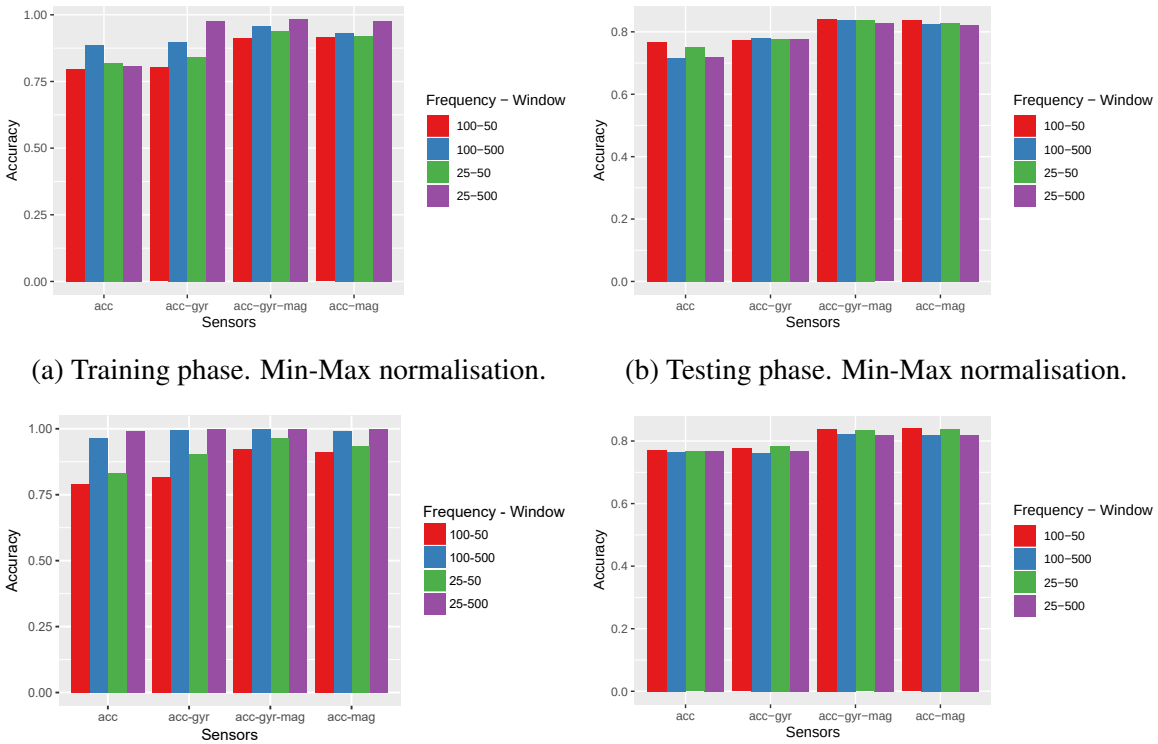
- the accuracy is higher standardising the data instead normalising it, whatever is the sampling frequency, window size and sensors taking into account
- Same as of the LR results, reducing the window size from 500 to 50 improves the accuracy, whatever is the sampling frequency and reducing the sampling frequency decreases the accuracy.
- As well, like of the LR prediction accuracy results, taking into account data from more sensors increase the accuracy. The highest accuracy result, when taking into account three sensors, is 88.37% and taking into account only accelerometer data is 83.95%. However, when taking into account data from the accelerometer and magnetometer sensor, the highest accuracy is 88.15%, almost the same as taking into account three sensors. On the other hand, when taken into account data from the accelerometer and gyroscope sensors, the highest accuracy score is 83.21%.

4.8.3 Convolutional neural network model

In this section, we test the accuracy of a convolutional neural network model, including four convolutional layers. The input to the first convolutional layer is a matrix of dimensions $(128, n_{inputs})$ where n_{inputs} it is the number of sensors taking into account. The kernel size of each layer is 2, and the numbers of filters apply in each layer are 18, 36, 72 and 144.

We choose this architecture after testing a few of them but we are not going to report the results from others

Figure 4.5 shows the accuracy achieved for the model when sampling the data with different frequencies (100 and 25 Hz) when the window size is 128 and taken into account data from different sensors. Figures 4.5a and 4.5b show the prediction results of the training and testing



(a) Training phase. Min-Max normalisation.

(b) Testing phase. Min-Max normalisation.

(c) Training phase. Standarisation.

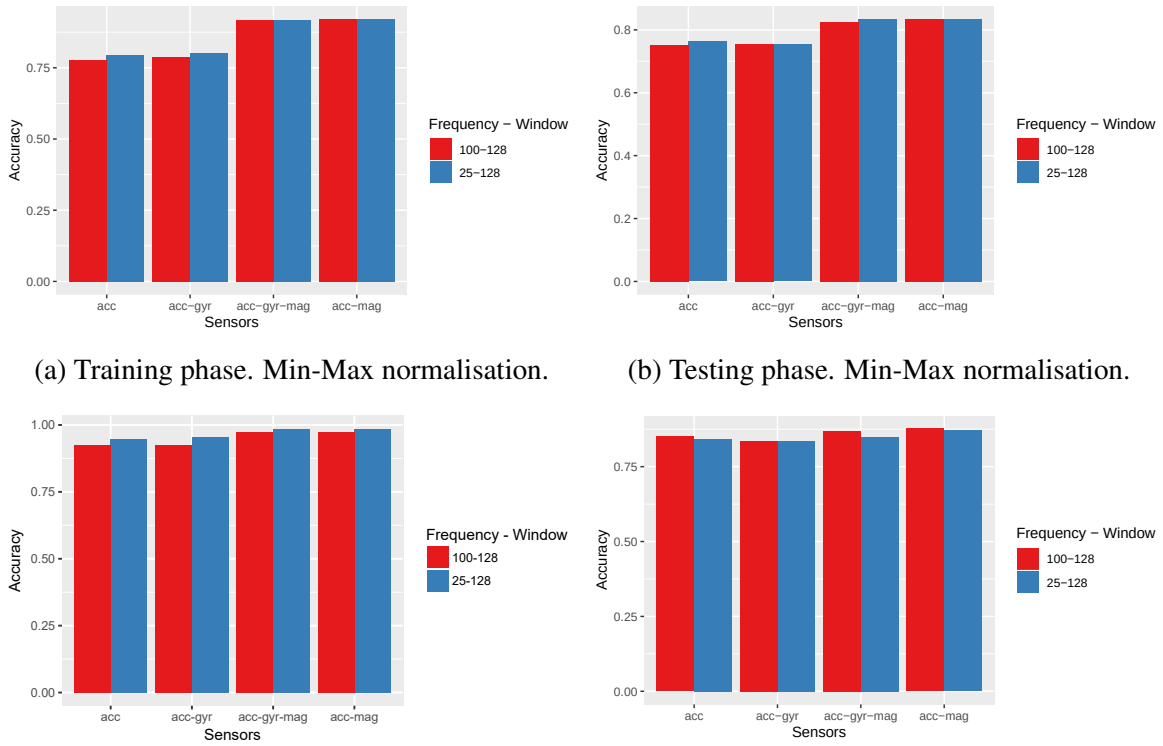
(d) Testing phase. Standarisation.

Figure 4.4 Accuracy prediction results of the training and testing phases of the MLP model. Top row shows the results when the data has been normalised and the bottom row when the data has been standardised.

phases when the data has been normalised; and figures 4.5c and 4.5d show the prediction results of the training and testing phases when the data has been standardised.

We can observe in the testing phase that:

- the accuracy higher standardising the data instead of normalising, whatever is the sampling frequency, window size and sensors taking into account, e.g., when taking accelerometer at 100 Hz, standardising the data the accuracy is 84.28% and normalising 76.28%
- When standardising the data, reducing the sampling frequency decreases the accuracy, the same as the accuracy results of the LR and MLP models. However, it does not occur the same when normalising
- Differently than the accuracy results of the LR and MLP models, when taking into account data from the three sensors does not improve the accuracy. The maximum



(a) Training phase. Min-Max normalisation.

(b) Testing phase. Min-Max normalisation.

(c) Training phase. Standarisation.

(d) Testing phase. Standarisation.

Figure 4.5 Accuracy prediction results of the training and testing phases of the CNN model. Top row shows the results when the data has been normalised and the bottom row when the data has been standardised.

accuracy occurs when taking into account data from the accelerometer and magnetometer sensors 87.67%, while the accuracy when taking data from the three sensors is 86.93%. Taking into account, only data from the accelerometer sensor and from the accelerometer and gyroscope sensors together decrease the accuracy results to 85.06% and 83.65%, respectively. Note than when taking data from the accelerometer and gyroscopes sensors together; the accuracy is lower than when taking into account only data from the accelerometer sensor.

4.8.4 Recurrent neural network

In this section, we test the accuracy of a recurrent neural network, including two LSTM layers.

Figure 4.6 shows the accuracy achieved for the model when sampling the data with different frequencies (100 and 25 Hz) when the window size is 128 and taken into account data from

different sensors. Figures 4.6a and 4.6b show the prediction results of the training and testing phases when the data has been normalised; and figures 4.6c and 4.6d show the prediction results of the training and testing phases when the data has been standardised.

We can observe in the testing phase results that:

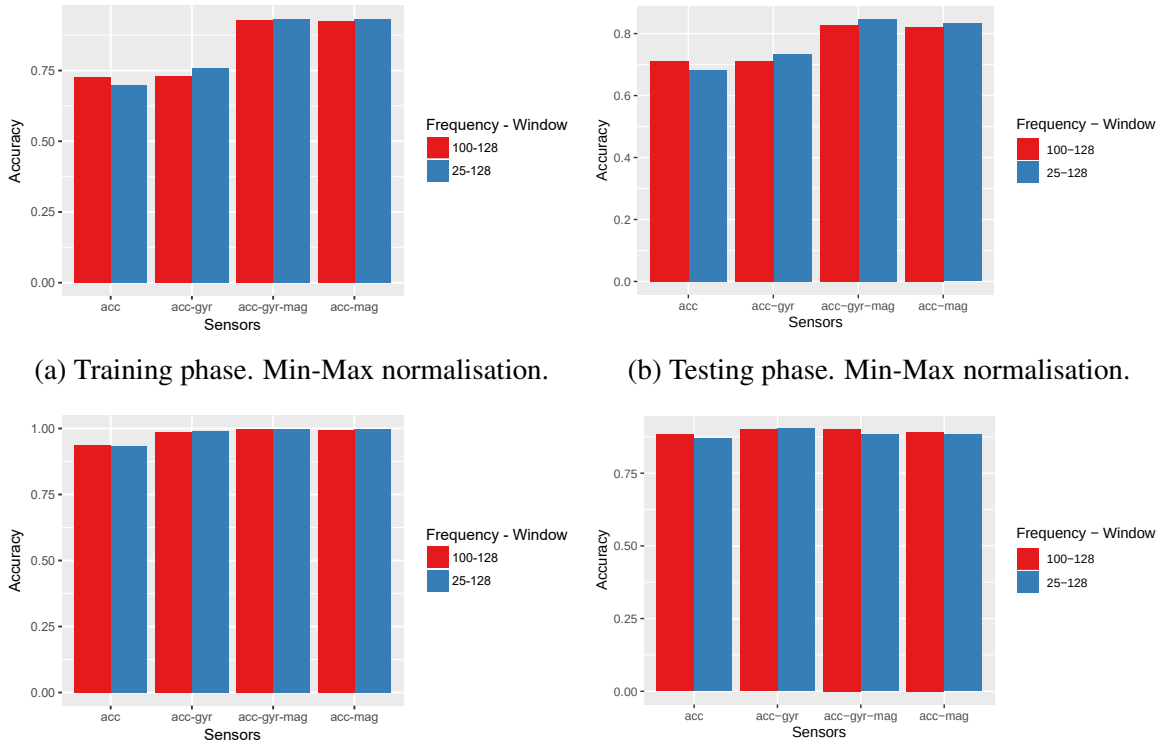
- Same to the MLP and CNN models, the accuracy is higher standardising the data instead normalising it, whatever is the sampling frequency, window size and sensors taking into account
- Same than the accuracy results of the LR, ML Pand CNN models, reducing the sampling frequency decreases the accuracy.
- Like in the LR and MLP prediction accuracy results, taking into account data from more sensors increase the accuracy. The highest accuracy result, when taking into account three sensors, is 90.23% and taking into account only accelerometer data is 88.25%. However, in this model, differently from LR, MLP and CNN, when taking into account data from the accelerometer and magnetometer sensor have approximately the same accuracy than when taking into account data from the accelerometer and gyroscope sensor, 89.09% and 90.03% respectively.

4.8.5 Support vector machines models

In this section, we show the prediction results of the Support Vector Machine model when we vary the parameters:

- Penalty parameter of the error term. It also controls the trade-off between smooth decision boundaries and classifying the training points correctly.
- We test four different kernels: linear, poly, sigmoid and radial.
- The kernel coefficient for the poly, sigmoid and radial Kernels. As we saw in section 2.2.6, the gamma parameter defines how large is the influence of a single training. Low values are meaning strong influence and high values meaning low influence.

Figure 4.7 shows the highest effectiveness result of SVM with different penalty and gamma parameters for models with four different kernels (linear, poly, sigmoid and radial) when we are pre-processing the data using the standardisation and normalisation techniques.



(a) Training phase. Min-Max normalisation.

(b) Testing phase. Min-Max normalisation.

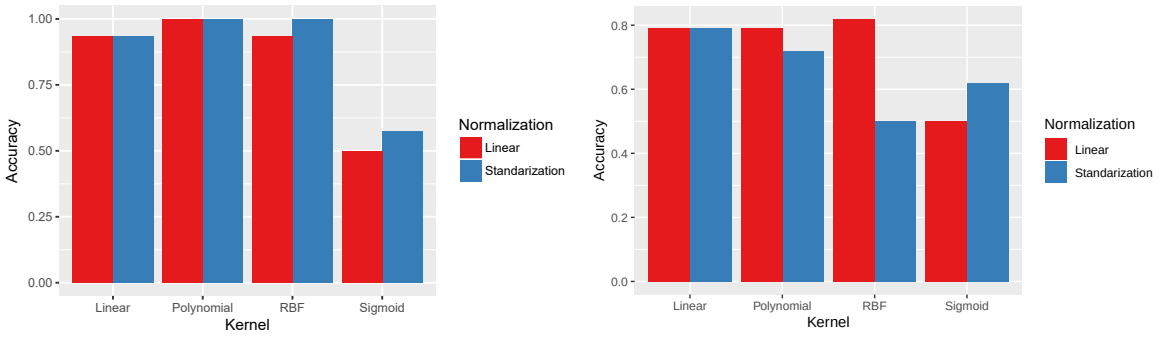
(c) Training phase. Standarisation.

(d) Testing phase. Standarisation.

Figure 4.6 Accuracy prediction results of the training and testing phases of the RNN model. Top row shows the results when the data has been normalised and the bottom row when the data has been standardised.

We can observe in the results of the testing phase, using a linear kernel, the accuracy is the same (79.21%) using both pre-processing techniques. When using the poly and sigmoid kernels, the accuracy drops whatever is the scaling technique used. The decrement is higher when using the standardisation technique. Using an RBF kernel with the standardisation technique leads to a small accuracy (0.5%). However, using the scaling technique, the model achieves the highest accuracy between that obtained from each of the different models (81.97%).

Figure 4.8 shows the effectiveness of an SVM model with a radial kernel when normalising the data (which was the model that achieved the highest accuracy in the previous experiment). We show the highest accuracy result for models with different penalty and gamma parameters when sampling the data with different frequencies (100 and 25 Hz), using different window sizes (500 and (50 samples), and taking into account data from a different number of sensors. Figure 4.8a shows the prediction results of the training phase and figure 4.8a of the testing phase.



(a) Training phase.

(b) Testing phase.

Figure 4.7 Accuracy of the SVM model in the training and testing phases when normalising and standardising the data; and using four different kernels: linear, polynomial, sigmoid and radial.

The same principles which we have seen for the LR algorithm when using different frequencies, window sizes and sources of data occur for the SVM algorithm.

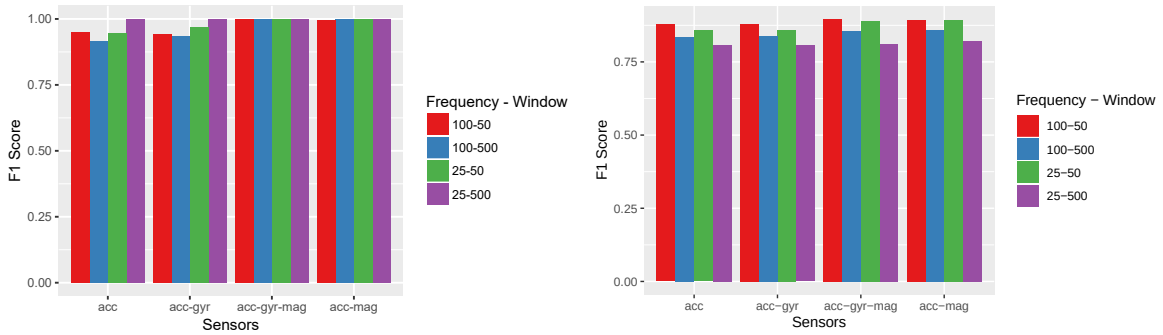
We can observe in the testing phase results:

- Reducing the window size from 500 to 50 improves the accuracy, whatever is the sampling frequency.
- Reducing the sampling frequency, decrease the accuracy. The decrement is higher when the window size is larger.
- Taking into account data from more sensors increase the accuracy. The highest accuracy result when taking into account three sensors is 89.69%, and taking into account only accelerometer data is 87.77%. However, when taking into account data from the accelerometer and magnetometer sensor, the highest accuracy is 89.31%, almost the same as taking into account three sensors. On the other hand, when taken into account data from the accelerometer and gyroscope sensors, the highest accuracy score is 87.85%.

4.8.6 Random forest model

In this section, we test the accuracy of the random forest algorithm when the number of trees in the forest takes a value of $[50, 100, 200, 300, 400, 500]$.

Figure 4.9 shows the highest accuracy achieved between the models with a different number of trees, when sampling the data with different frequencies (100 and 25 Hz), using different



(a) Training phase.

(b) Testing phase.

Figure 4.8 Accuracy results in the training and testing phases of the SVM model with a radial kernel when normalising the data .

window sizes (500 and (50 samples) and taken into account data from different sensors. Figures 4.9a and 4.9b show the prediction results of the training and testing phases when the data has been normalised; and figures 4.9c and 4.9d, shows the prediction results of the training and testing phases when the data has been standardised.

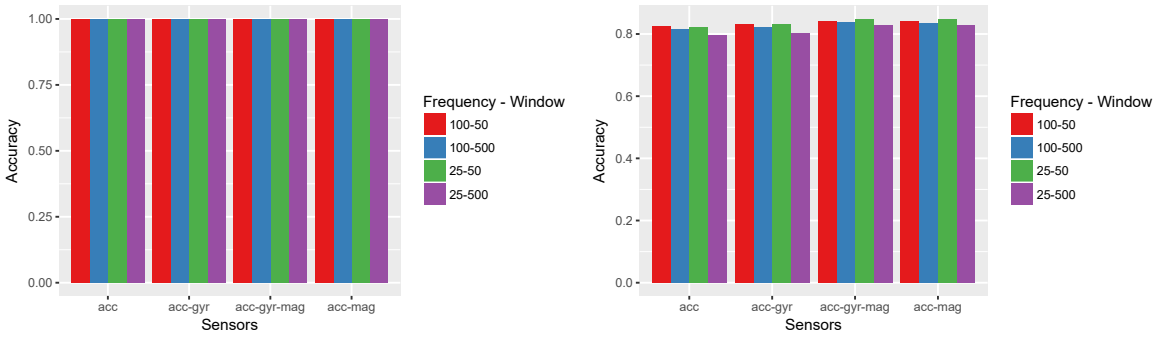
We can observe in the testing results, that there is no difference in the accuracy when normalising or standardising the data. About using different sampling frequencies, window sizes, and the number of sensors, the same principles observed in the previous models (LR, MLP, RNN and SVM) occur.

4.9 Conclusions

In this chapter, we have used the supervised machine learning techniques introduce in Chapter 2 to identify users based in motion data, and we have compared the effect of using different window sizes, sampling frequencies, number of sensors and preprocessing techniques to scale the data.

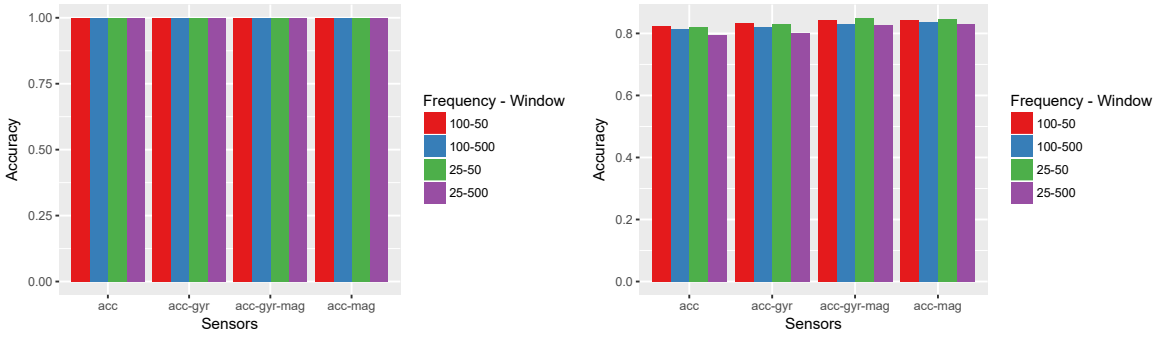
Table 4.2 shown the highest effectiveness results of each of the models analysed in section 4.8. We can observe that the RNN model achieves the highest accuracy of 0.902. However, as well, it is the method which takes more time to train the model 170.16 seconds.

Trought the experiments of section 4.8, we can extract some of the conclusions about using different sampling frequencies, window sizes, number of sensors taking into account and preprocessing techniques to scale the data.



(a) Training phase. Min-Max normalisation.

(b) Testing phase. Min-Max normalisation.



(c) Training phase. Standarisation.

(d) Testing phase. Standarisation.

Figure 4.9 Accuracy of the RF model in the training and testing phases .Top row show the results when the data has been normalised and the bottom row when the data has been standardised.

We can establish that:

- We obtain better accuracy results standardising the data for the LR and SVM models. For models based on neural networks such as MLP, CNN and RNN, it is better to scale the data. For the RF model, the preprocessing technique applied does not have a significant impact on the prediction results.
- Reducing the window size improves accuracy.
- Reducing the sampling frequency, decrease the accuracy.
- Taking into account data from more sensors increases the accuracy, but the increment is very small when taking into account data from all the sensors instead of data only from the accelerometer and magnetometer sensors.

We can conclude that user identification is feasible from motion data. However, the current experiments have some limitations which will be interesting to review in future work:

- We have conducted identification discriminating between two people. Future work could extend these experiments and complete the same investigations distinguishing between more persons.
- We have not compared the efficiency of different architectures for the CNN and RNN models. It will be interesting to compare and report them in future work.
- We use a CNN architecture with a maximum kernel size of 2. Because the input instance can have a dimension of $128 samples x 3 channels$ (note that here channels equal to the number of sensors taking into account), this kernel could not capture some patterns for these instances. In future jobs, it would be interested comparing the performance result with architecture with a filter of a bigger size.

Table 4.1 Accuracy metrics of the dummy classifier during the training and testing time.

Training dataset					Testing dataset						
Accur.	Precision	Recall	F1	AUC	Accur.	Precision	Recall	F1	AUC	FAR	FRR
0.4989	0.4992	0.4981	0.4983	0.5007	0.5009	0.4962	0.4973	0.5007	0.5007	0.5037	0.4947

Table 4.2 Highest effectiveness results of the different model used in section 4.8.

Model	Pre-Process	Sensor	Sampling Frequency	Window Size	Accuracy	F1 score	AUC	FAR	FRR	Train time
SVM	Standardisation	a-m-g	100	50	0.896	0.890	0.896	0.110	0.095	0.457
LR	Standardisation	a-m-g	100	50	0.841	0.842	0.841	0.140	0.176	0.437
RF	Normalisation	a-m-g	25	50	0.868	0.862	0.864	0.870	0.137	0.122
MLP	Normalisation	a-m-g	100	50	0.883	0.879	0.883	0.110	0.121	10.87
CNN	Normalisation	a-m	100	128	0.876	0.873	0.876	0.121	0.125	42.40
RNN	Normalisation	a-g-m	100	128	0.901	0.902	0.900	0.088	0.109	170.1

Chapter 5

User motion behavioural verification

5.1 Introduction

Continuous authentication is a promising approach to verify users during a work session, e.g., for mobile banking applications, the user does not need to input security information to authorise several operations. Recently, it has been demonstrated that changes in the motion patterns of the user may help to note the unauthorised use of mobile devices. Several approaches have been proposed in this area but with relatively weak performance results. In this chapter, we propose an approach that uses a Siamese convolutional neural network to learn the signatures of the motion patterns from users and achieve a competitive verification accuracy up to 95.8%.

Siamese neural networks have been successfully used in the context of face verification to learn a distance metric of the data space, which can be used to compare previously-unseen classes, i.e., images from people not seen during training [33]. Motivated by this, we propose a continuous authentication system, which detects unauthorised use on the smartphone, using the characteristic motion patterns of each individual interacting with the device. We use a Siamese convolutional neural network (CNN), which learns a distance metric from a large dataset, based on which we can extract deep features for new user authentication.

The authentication process is performed continuously during the whole session without requiring any explicit action from the user. The proposed approach meets the intrinsic design requirements of the platform, and achieve an accuracy rate near to 96% using a simple

classifier (one-class SVM), increasing the effectiveness of similar authentication approaches based on motion behavioural analytics.

5.2 Biometric approaches for user verification on smartphones

In this section, we review some different Biometric Authentication Systems (BASs) utilise for user validation on smartphones and discuss some of the engineering and methodological challenges limiting their implementation.

In verification systems based on face recognition, the system stores feature of images or video frames which are unique to the user. During the verification phase, the features of the new image sample are determined to belong to the user or not [8] . Iris verification system has used too for user verification. Most of the non-mobile iris user-verification systems use images obtained in near-infrared illumination since the iris is more effectively captured under this light condition [154]. When a picture of the iris is taken in real-world scenarios, distortion is introduced, and this causes a significant drop in accuracy [127], especially when the front camera is used. Additionally, the iris is constantly moving, and it is difficult to localise it in eye images. Although the quality of the smartphone camera has improved in recent years, this BAS is one with the highest FER, and smartphones do not rely on this technology yet. A noticeable exception is the Arrows NX F-04G model from Fujitsu Limited available in Japan, which has shown higher performance.

Periocular user-verification systems share a lot of similarities with iris and face recognition schemes. Periocular biometric features are extracted from an image of the facial region in the vicinity of the eye. The main advantage of the iris recognition method is that the data are easier to capture. Furthermore, an invisible light source is not required to capture the image. Extensive research of this method for smartphone user-validation purposes has been performed [128, 119, 101]. In [126], a one-time periocular authentication approach is presented, based on a semi-supervised anomaly detection method using a deep autoencoder. The BAS obtained the best performance results of a 8% FRR and 10% FAR.

More recently, some smartphone manufacturers have included a fingerprint reader, and they used it for user-verification purposes [84]. The main difference with face, iris, and periocular recognition is the focus of features obtained from the print pattern of the individual. An important percentage of smartphone users trust this validation method [178]. However, it

still suffers from high FAR and FRR [136]. Conventional touch sensors introduce physical distortion because of the pressure applied from the individual at the capture frame, and this critically affects the authentication process [136]. Furthermore, the classification is influenced by other environmental factors such as humidity. To mitigate these issues, a touchless fingerprint has been proposed where the device camera is used to capture an image of the dactylogram [136]. Besides acquisition challenges, fingerprint images databases represent a considerable risk: should there be a security breach, and fingerprint patterns are stolen, the authentication patterns could not be reset.

Voice recognition user-verification methods use the voice of each individual to discriminate between users. Banks have been very interested in this technology since it can be integrated into telebanking applications. In smartphone user-validation systems, this method has not received widespread attention. In a comparison among several BASs, voice recognition was found less practical than password entry and face recognition: most of the participants involved gave negative feedback it [140]. A commercial voice-based solution developed by Google called Trusted Voice, advise that the method is less secure than PIN, password, and pattern lock during the activation process. This technology cannot be easily implemented as a continuous authentication method since in the vast majority of activities performed on smartphones do not involve voice [140].

With the embedding of sensors such as an accelerometer, magnetometer, and gyroscope, motion user-verification for smartphones became a subject of an increasing number of studies. Some of the recent literature is focused on identifying users based on their holding patterns [35]. Since motion data on smartphones can be continuously collected, a similar methodology can be used to implement continuous BASs, although the literature in this area is very sparse [92].

In [104], a continuous motion recognition system was proposed base on accelerometer and gyroscope data. The method is applied in two phases. In the first phase, they transformed the observations in a new set of features and estimated their general distribution using a Gaussian mixture model. When a new user joins the scheme, the performance maximum a posteriori adaptation of the mean vectors of the general data model to build a client-specific model. Then, both models are used to produce a verification score for each new observation. The authors obtained an ERR of 18.2% in real-world scenarios.

In [139] the authors introduced a multi-modal approach including accelerometer, gyroscope, and touch-screen observations. The authors use a one-class SVM model to classify the

samples as either belonging to the owner or a guest/attacker. They test the approach with a dataset collected in a controlled environment where users were asked to type a text when sitting and when walking. They obtained an ERR of 7.16% when the user was walking and 10.05% when the user was sitting. The authors deferred the evaluation of the approach to real-world scenarios for future studies.

In [92], it was proposed a continuous verification approach based on keystroke data and the weighted *k*-nearest neighbour algorithm. Experiments in a controlled environment showed an ERR of 3.5% when the re-authentication time was set up to ten minutes. Decreasing the re-authentication time gives considerably worse results, thus limiting the practicality of the approach on real scenarios being the average smartphone session duration 72 seconds, as a recent study suggests [27]. Additionally, an obvious limitation of touch recognition for continuous authentication is the requirement of continuous input from the user. The smartphone activity usage is very diverse, and some of the most popular activities involve few typing.

5.3 Data Selection

Feature Learning is the process of selecting the most important and/or relevant characteristics of a data set, with the aim of improving the prediction performance of the predictors, providing faster and more profitable predictors, and providing a better understanding of the process underlying that generated the data. In the data selection process, a subgroup of features is chosen between all included in the dataset. The subgroup means to include the most influential variables to the problem. There are several ways to conduct the data selection process, which we are going to introduce in this section.

The size of the datasets is different for each problem. Sometimes they can be small while others are tremendously large in size, especially when they have a large number of features, which may cause them to be difficult to process. Some of the difficulties High-dimensional data sets may introduce are :

- Some features may act as noise, which reduces the accuracy of the machine learning approach.
- The model takes longer to train.
- Allocation of unnecessary resources for these characteristics.

For all this, feature selection can be beneficial in our approach. Machine learning models learn from past data. In some cases, captured features may be irrelevant what may influence the effectiveness results of the approach. Finding a meaningful representation of the data may help to improve the accuracy rate of the system and reduce the computational burden.

Previous results indicate that feature vector representations extracted from convolutional neural networks are very powerful [129]. Commonly, the new representation is obtained from a hidden layer of the CNN which has been previously trained. This vector representation has two important properties:

- Dimensionality Reduction: it is a more efficient representation
- Contextual Similarity: it is a more expressive representation

The new data space, which contains the new representations, is usually called the embedded space.

Transfer learning

Many times, in the feature extraction process, people use pre-trained CNN models. This practice is usually called transfer learning. For example in [129] the authors used a publicly available CNN called *OverFeat* [135] which was trained for the classification, localisation and detection tasks of the ImageNet Large Scale Visual Recognition Challenge 2013 [132]. The authors use the features extracted from the *OverFeat* network as a generic image representation for a range of tasks of object image classification, scene recognition, fine-grained recognition, and image retrieval applied to a diverse set of datasets. They selected the tasks and datasets as they are gradually different from the original task and data the *OverFeat* network was trained to solve. Thus, showing the advantages of using the deep learned features.

5.4 Dataset

To evaluate our approach, we use a publicly available dataset of behavioural data on smartphones [172]. The dataset contains samples from 100 volunteers, which includes multiple modalities: movement, orientation, touch, and gesture. The data was collected under three task scenarios (reading, writing, and map navigation) and two body motion conditions (sitting and walking). It is the same dataset used in the experiments in Chapter 4.

Between all the volunteers, 90 of them performed 24 sessions (8 reading sessions, 8 writing sessions, and 8 map navigation sessions), each session lasting about 15 minutes. We will use data from this group of 90 volunteers. We discharge data from the other 10 users because the sequence is too short for the experiment we want to show.

We split the group of 90 users into two subgroups, of 60 and 30 users. We used data from the first subgroup (data from the group of 60 users) to train the feature extractor model (a siamese neural network model). We used the data from the second subgroup (data from the group of 30 users) to train and test the verification model based on the one-class SVM model (note that the testing dataset includes samples not seen in the training phase).

As we have said, to train the feature extraction model, we generate a training dataset combining samples from each of users in the group of 60 users.

- positive pairs: both samples of the pair are from the same user (the number of positives pairs is equal for each of the users, i.e. we balance the number of positive pairs by user).
- negative pairs: we generate negative pairs matching samples from one user with a sample from a different user. We repeat the process from all possible combinations of users, with the same number of negative pairs for each singular user, i.e. we balance the number of positive pairs by user.

Furthermore, we generate the same percentage of positive pairs than negative ones, i.e. we balance the number of positive and negative pairs by class.

Once the deep feature model has been trained, we transform the samples from the group of 30 users. Based on the deep features of the samples from this group of users, we train the one-class SVM model with observations from one of the 30 users -i.e., the legitimate user- and we evaluated the model with a testing dataset including samples from the same user and from one different user, i.e., the fraudulent user. Thus, we evaluate the ability of the approach classifying unseen observations as either belonging to the owner of the device or not. We repeat the same experiment for all possible pairs of legitimate-fraudulent users, and we show the average of the results (in total, 870 scenarios).

Note that, the training dataset includes samples from 18 sessions, 6 sessions for each task-motion condition activity (from the legitimate user). The testing dataset includes observations from 6 sessions, one for each task-motion condition activity from each of the users (from the

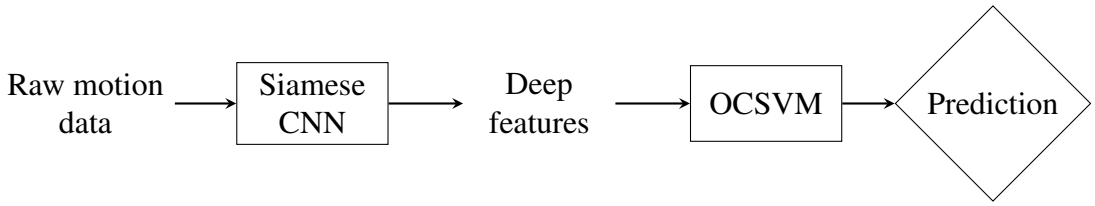


Figure 5.1 Machine learning pipeline of our approach.

legitimate and fraudulent users). The observations from the legitimate user are different from those used to train the model.

5.5 Experiments results

First, we are going to transform the feature space of a group of users of the dataset introduced in 5.4. For the data transformation, we are going to use the Siamese neural network approach presented in 2.2.5 when using different architectures, i.e., MLP, 1d-CNN, and 2d-CNN. We are going to visualise for each architecture, the data transformation of two groups of users containing 6 and 18 users. Through the visualisation, we want to show which of the different architectures used to transform the data lead to a more meaningful representation.

After that, we will implement the user-detection system introduced in 5.4 with the siamese network architecture, which produce a more meaningful data transformation according to the previous experiment. We will compare this pipeline (siamese network + oc-SVM) with others which have not transformed the data using a Siamese neural network, i.e. transforming the data with a CNN, or the data have not been transformed at all, i.e. using the raw data directly.

5.5.1 Distribution of deeply learned features

Here, we are going to visualise the data transformation of users from the dataset introduced in 5.4. We are going to train different Siamese networks 2.2.5 when using different architectures, i.e., MLP, 1d-CNN, and 2d-CNN. We will show the data transformation of two groups of users for each architecture, with 6 and 18 users respectively (we want to understand how to change the new feature space when including more and more users).

To show the data transformation space, we apply Principal Component Analysis to the output vector from the Siamese Network, and we plot the two main principal components as shown in figures 5.2 through 5.8.

MLP Siamese neural network

We have used an MLP Siamese network to learn deeply features of the HMOG dataset. Each subnetwork consists of a multilayer perceptron network with two hidden layers.

Figure 5.2 shows the distribution of deeply learned features of six users of the HMOG dataset. Each subfigure shows the result for four different scenarios with different users. We can see that deeply features of each class are not separate.

Figure 5.3 shows the distribution of the deeply learned features of 18 users of the HMOG dataset. Each subfigure shows the result for four different scenarios with different users. The same of the scenario with six users deeply features are not separate by the user.

1d-CNN Siamese neural network

Here, we used a 1d-CNN Siamese network to transform the data from the same group of users. Each subnetwork consists in four 1d-CNN layers and four pooling layers.

Figure 5.4 shows the distribution of the deeply learned features of six users. Each subfigure shows the result for four different scenarios with different users. We can see that distribution of each class are well separate but close each other in a diagonal.

Figure 5.5 shows the distribution of the deeply learned features of 18 users of the HMOG dataset. Each subfigure shows the result for four different scenarios with different users. Different that the scenario with 6 users clusters are not in a diagonal but still they are quite close one each other.

2d-CNN Siamese neural network

We have used a 2d-CNN Siamese network to learn deep features of the HMOG dataset. Each subnetwork consists in a 2d-CNN network with four convolutional layers and four pooling layers.

Figure 5.6 shows the distribution of the deeply learned features of six users of the HMOG dataset. Each subfigure shows the result for four different scenarios with different users. Generally, samples of the same class are grouped together and far away from groups of the rest of the classes.

Figure 5.7 shows the distribution of the deeply learned features of 18 users of the HMOG dataset. Each subfigure shows the result for four different scenarios with different users. The

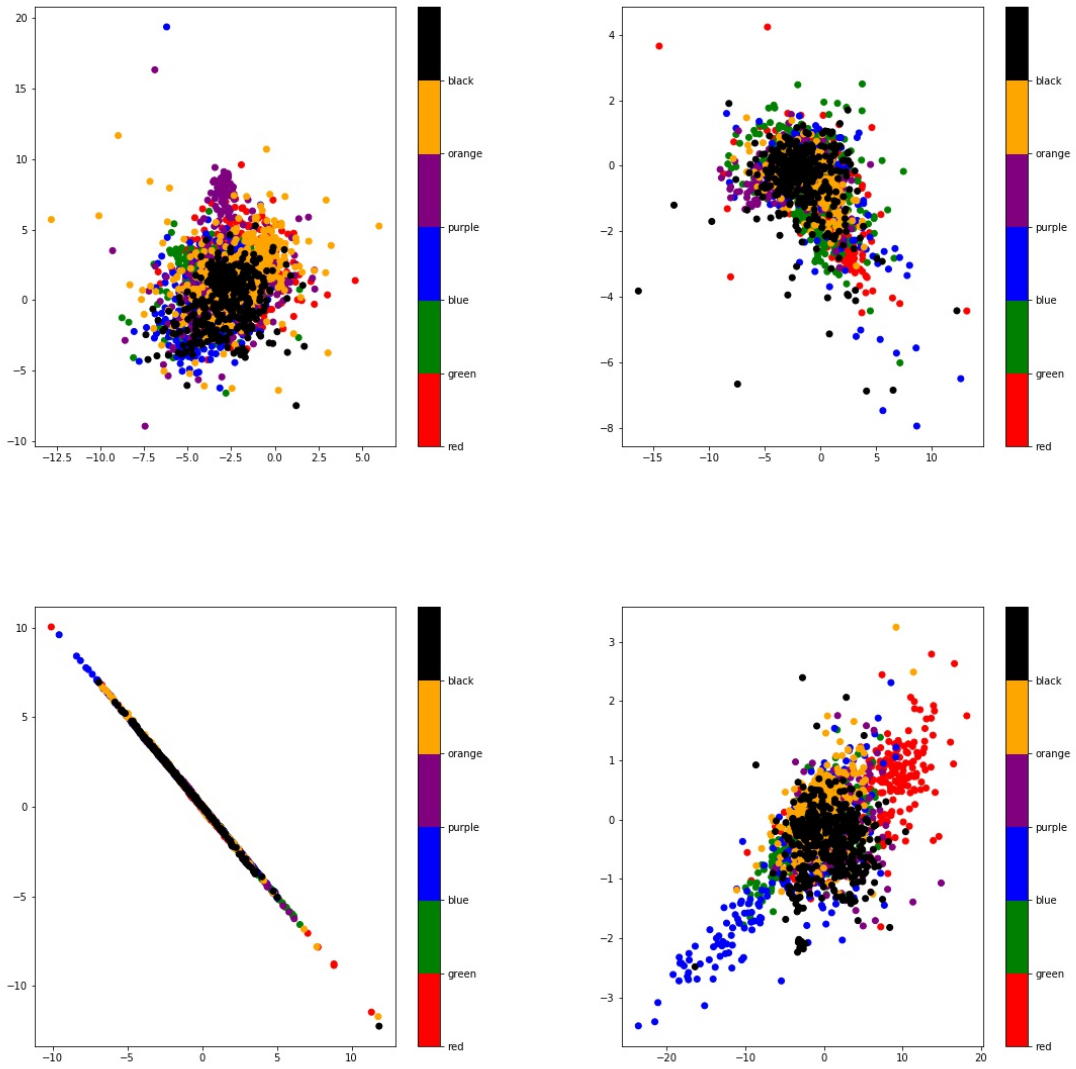


Figure 5.2 Distribution of the deeply learned features of six users of the HMOG dataset using an MLP Siamese network with a pairwise constraint. Each subfigure shows a scenario with different users.

deeply learned features of the same classes are grouped together, but groups from different classes are close to each other.

5.5.2 Motion user detection system

Figure 5.1 shows the machine learning pipeline of our proposed detection system, which is based on motion data. In an off-line phase, we use a Siamese neural network to learn a

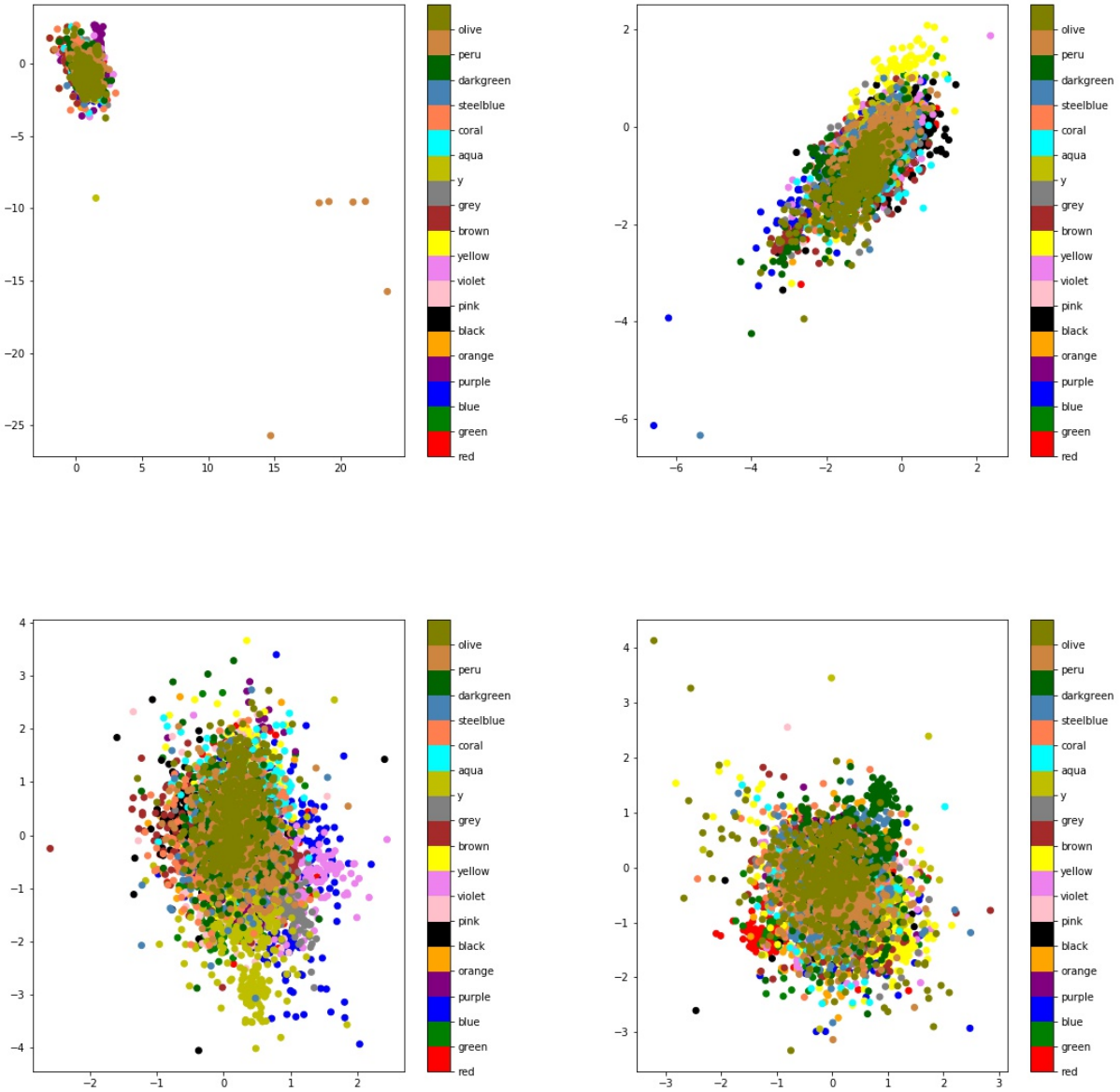


Figure 5.3 Distribution of the deeply learned features of 18 users of the HMOG dataset using a MLP Siamese network with a pairwise constraint. Each sub figure shows a scenario with different users.

meaningful representation of motion patterns. Samples from users are transformed into the embedded space learned by the Siamese network. We use the deep features of each user, i.e., the legitimate user, to train a one-class support vector machine model. New observations captured with the same device will be classified as belonging to the owner or to a different

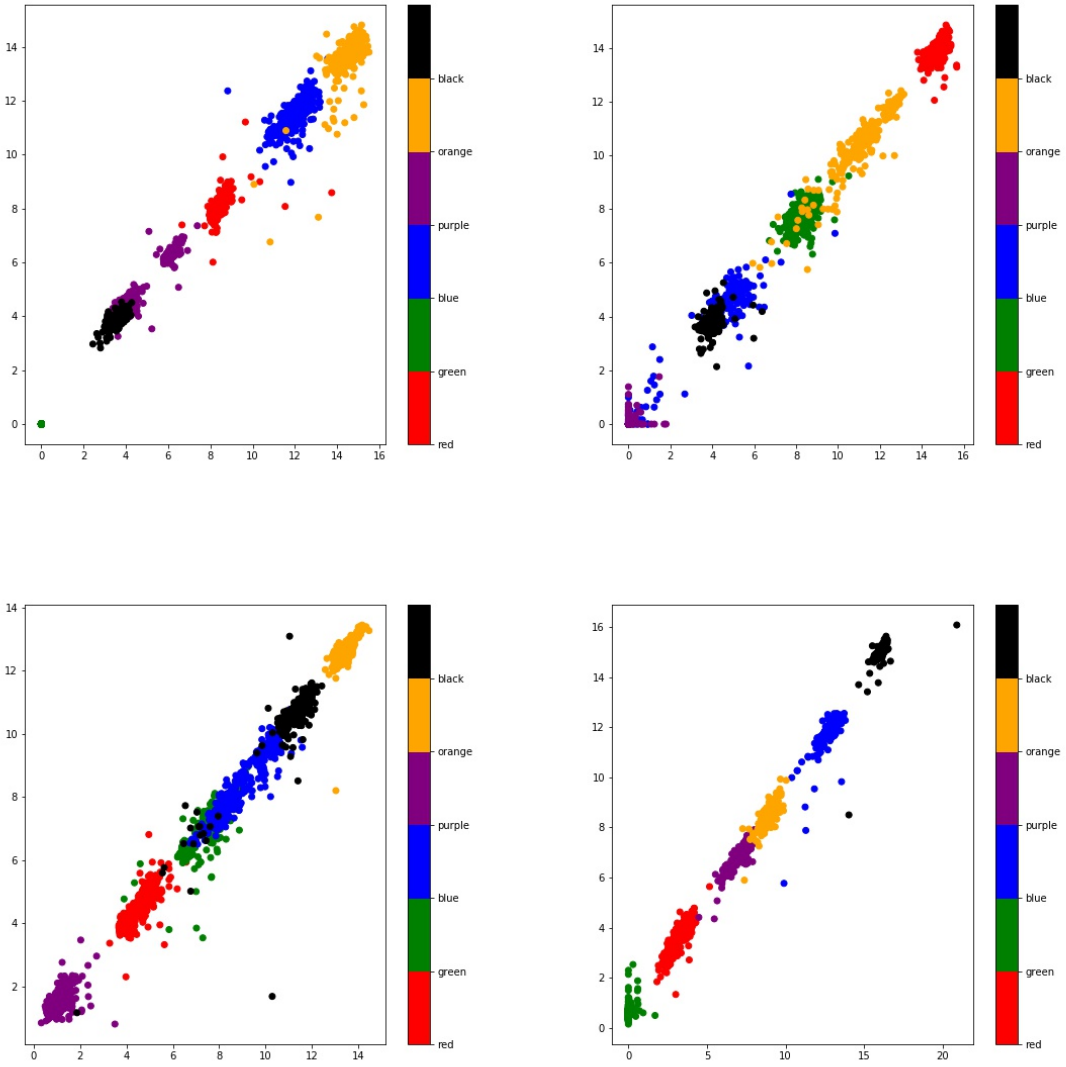


Figure 5.4 Distribution of the deeply learned features of six users of the HMOG dataset using a 1d-CNN Siamese network with a pairwise constraint. Each subfigure shows a scenario with different users.

person, i.e., the intruder. To evaluate the effectiveness of our proposed approach, first, we train a Siamese 2D-CNN with a pairwise constraint with observations from 60 users picked up randomly from the HMOG dataset [172], which represent the user population. We use the 2D-CNN architecture for the Siamese neural network as it was the architecture that shows better transformation representation in the experiments of the previous subsection.

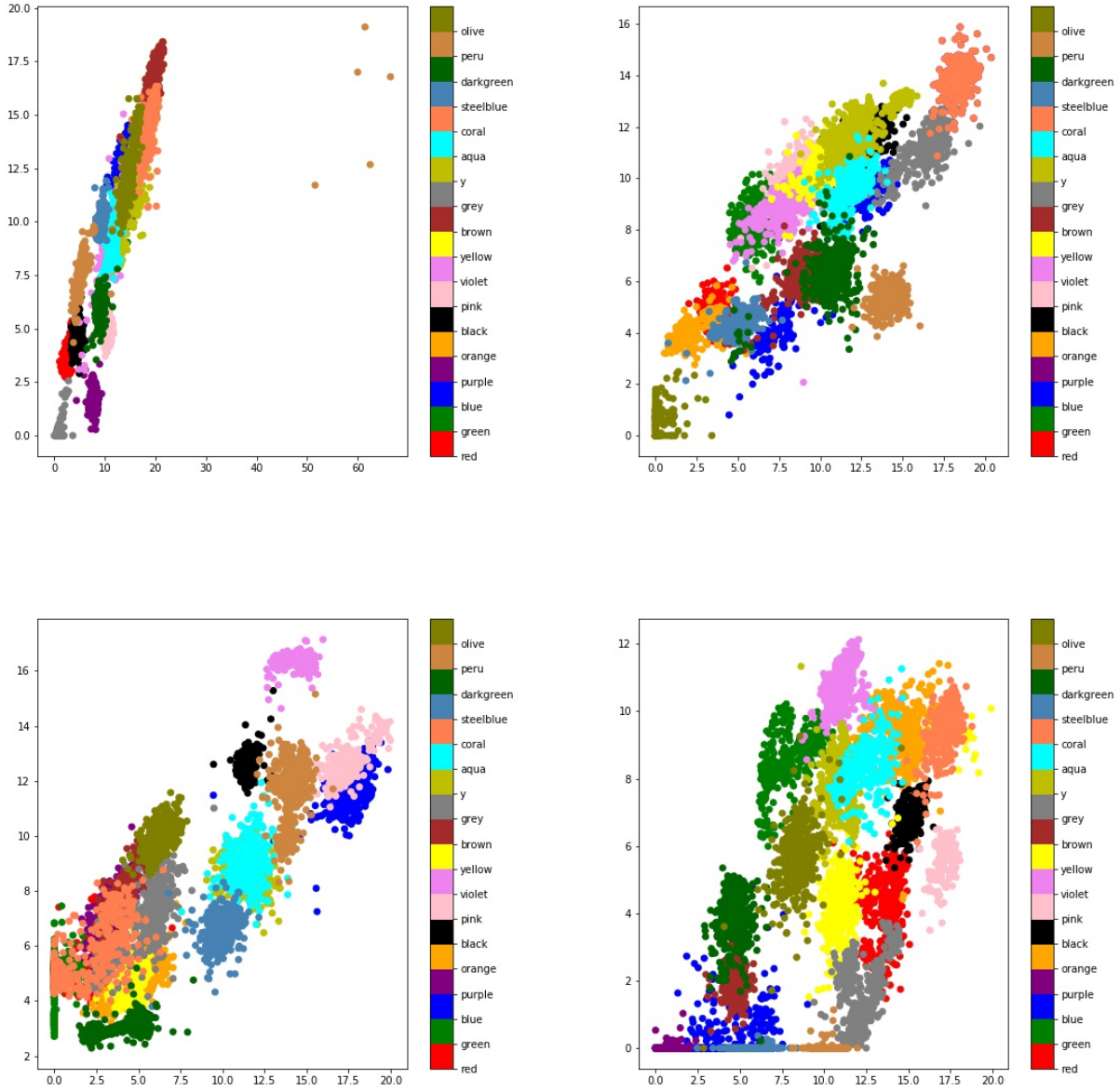


Figure 5.5 Distribution of the deeply learned features of 18 users of the HMOG dataset using a 1d-CNN Siamese network with a pairwise constraint. Each subfigure shows a scenario with different users.

The architecture of the Siamese CNN consists of 4 convolutional layers and 4 max-pooling layers connected alternatively. A 2D block of motion data of size window size by nine channels (one channel for each directional axis of each sensor taken into account, i.e., accelerometer, gyroscope, and magnetometer) is given to a convolutional layer with 32

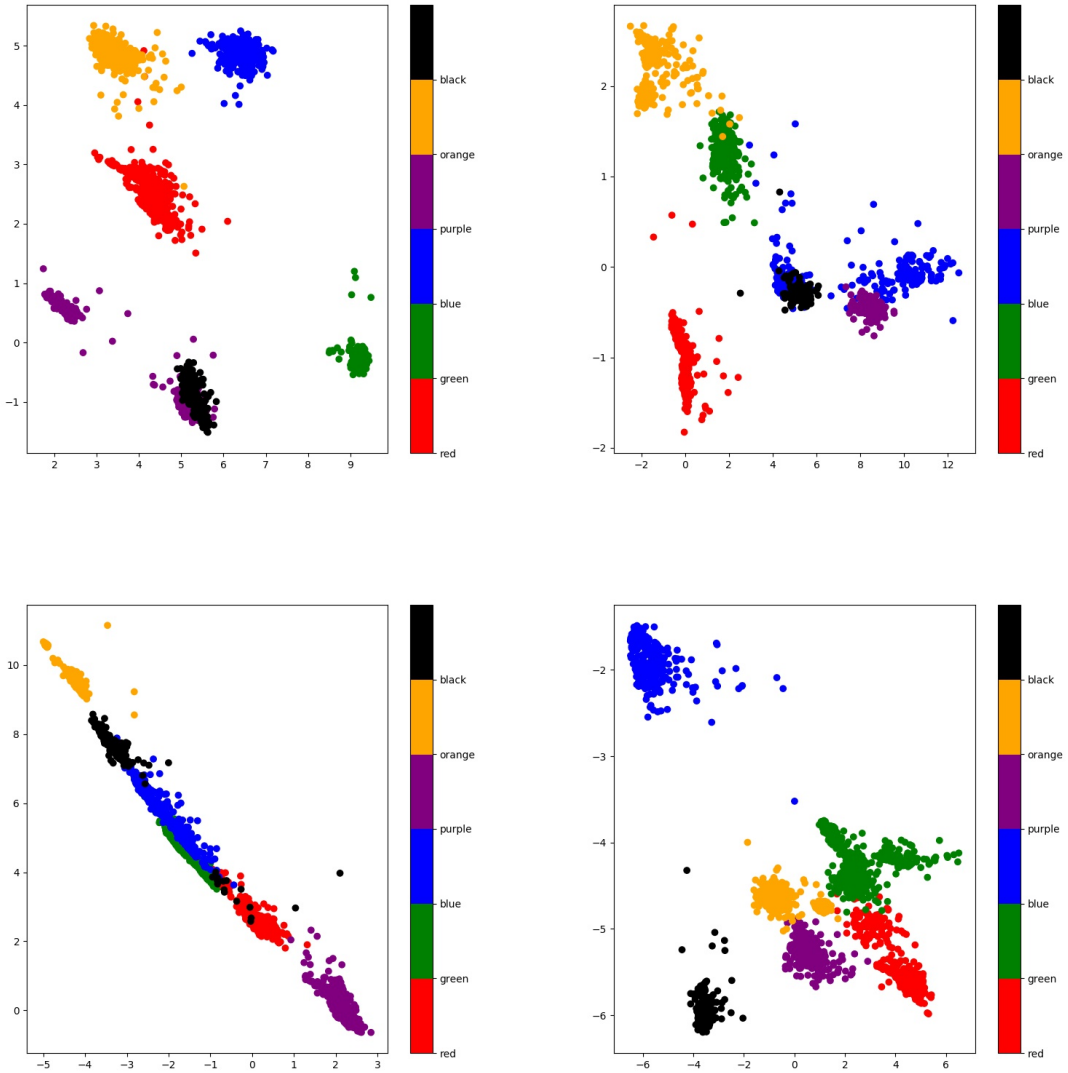


Figure 5.6 Distribution of the deeply learned features of six users of the HMOG dataset using a 2d-CNN Siamese network. Each subfigure shows a scenario with different users.

filters of size 7×7. The configuration of the second and third convolutional layers is 64 filters of size 5×5 and 128 filters of size 3×3, respectively. We change the number of filters of the fourth convolutional layer with size 3×3 depending on the window size of the scenario. Thus, we can compare the accuracy rate for different sampling frequencies when the re-authentication time is the same. We reshape the output of the top layer of the network approximately to a 64-dimensional feature vector.

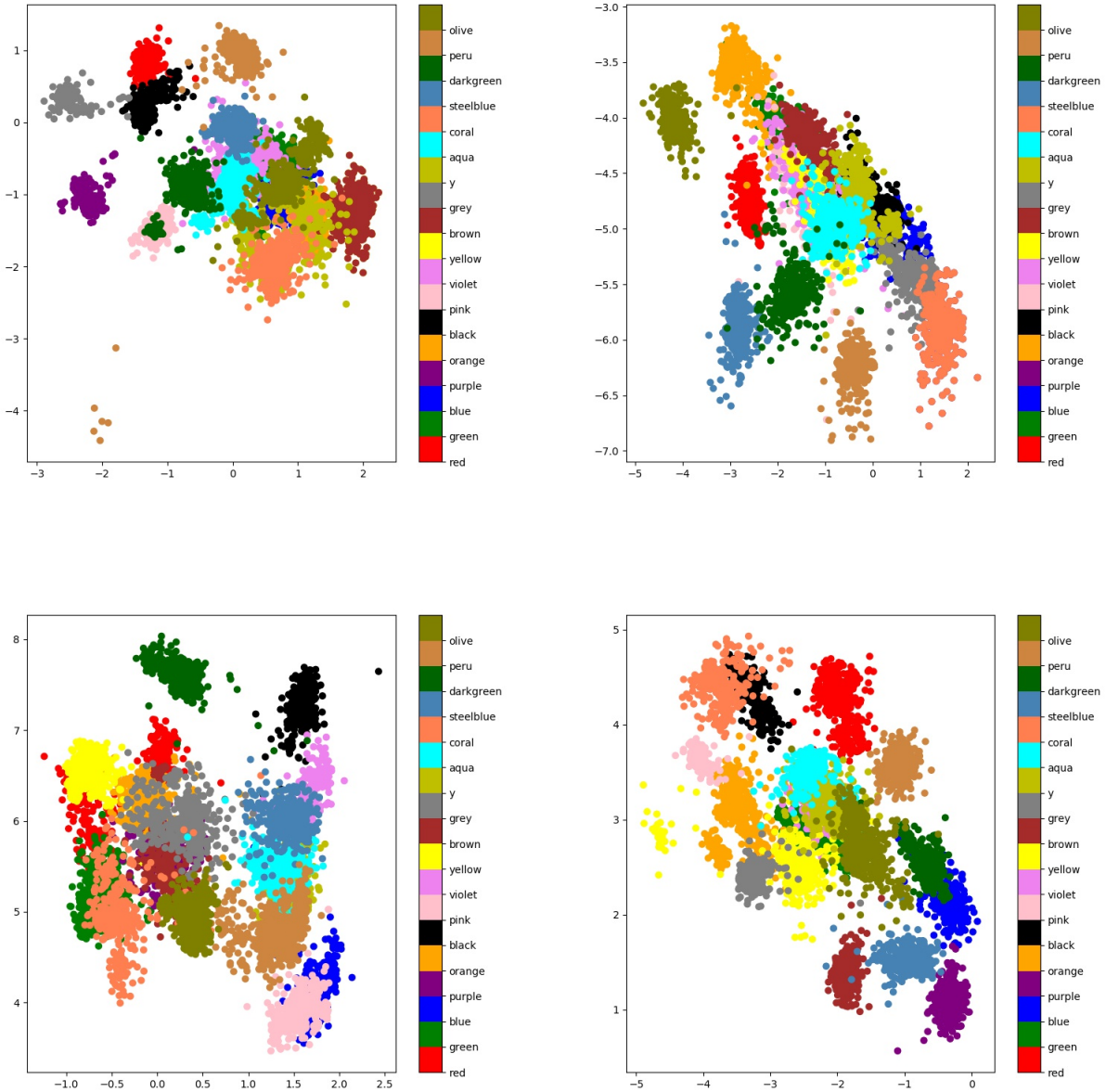


Figure 5.7 Distribution of the deeply learned features of 18 users of the HMOG dataset using a 2d-CNN Siamese network with a pairwise constraint. Each subfigure shows a scenario with different users.

As noted, the resulting feature maps from the convolutional layers are fed to a max-pooling layer, which takes the max over 2×2 spatial neighbourhoods (for all four pooling layers).

To show the distribution of the learned features, we apply dimensionality reduction using the principal component analysis to the extracted features of the testing dataset, including samples from the group of 30 users. Figure 5.8 shows the first two principal components. We can observe that samples of the same user are predominantly grouped together, and groups of observations are separate between them.

Then, using the Siamese model, we transform the observations of the training dataset of the group of 30 users, and we train a one-class SVM model with a radial basis function kernel. We vary the gamma parameter (the inverse of the radius of influence of samples selected by the model as support vectors) with values between 0.0001 and 100, and we show the best accuracy rate obtained between all the results.

We have tested our approach with the following set ups:

- different sequence lengths: 0.5, 1 and 2 seconds.
- different sampling frequencies: 25 Hz and 100 Hz.

Furthermore, to show the importance of the feature extraction process, we also show the accuracy when the one-class SVM model is trained:

- with raw data.
- with the set of angles $\alpha\{x,y,z\}$, $\phi\{x,y,z\}$ and $\beta\{x,y,z\}$ describing the orientation of the three sensors (accelerometer, gyroscope and magnetometer) and their magnitudes $|a|$, $|g|$ and $|m|$ [104] (we call these variable engineered features).
- with statistics extracted from each window: mean, median, standard deviation, variance, max, min, mean of the module, mean of the set of angles for each axe of the three sensors. In total, for each instance, we extract 66 features.
- with a 64-dimensional features vector extracted by a CNN (with the same architecture of the Siamese subnetwork).

Figure 5.9 shows the averaged effectiveness results for the different training datasets. We observe that the model trained with raw data has an accuracy rate of 84.9%. Observations are very noisy, and observations from the legitimate and fraudulent users are not separated in the data space; thus, the boundary which calculates the one-class SVM model cannot separate the group observations from both classes (we can see that false acceptance rate and false rejection rate are quite high). When we train the model with the group of features we call

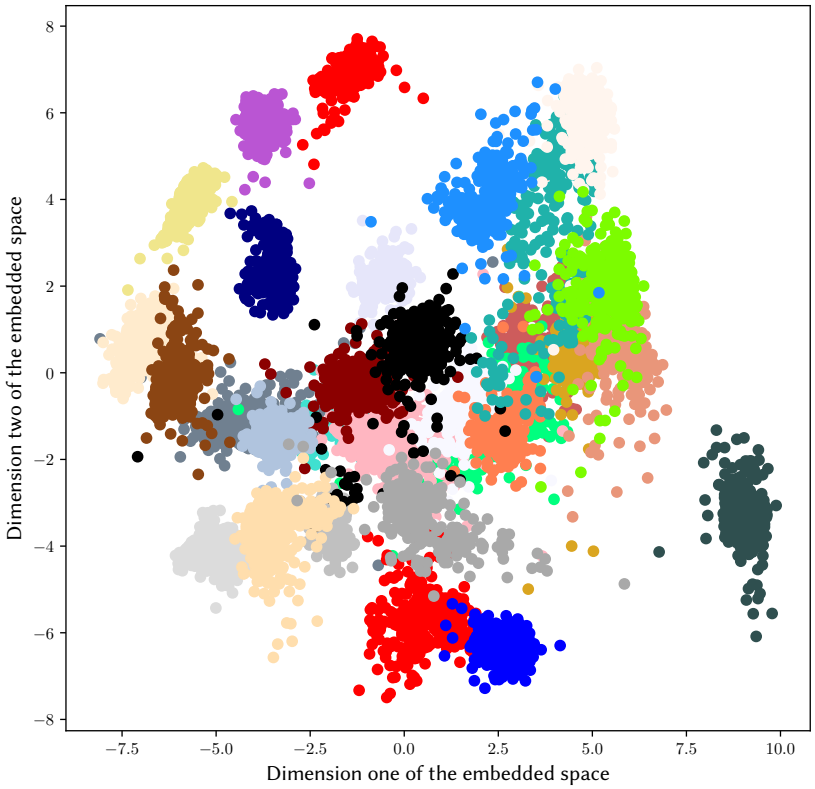


Figure 5.8 Transformation of the observations from 30 users by a Siamese CNN.

engineered (sets of angles describing the orientation and the magnitudes of the three sensors), the accuracy rate increased slightly to 85.5%, and it reached to 86.4% when we train the model with statistics (mean, median, variance, etc.) of the window. Experimental results suggest feature engineering improve moderately performance accuracy.

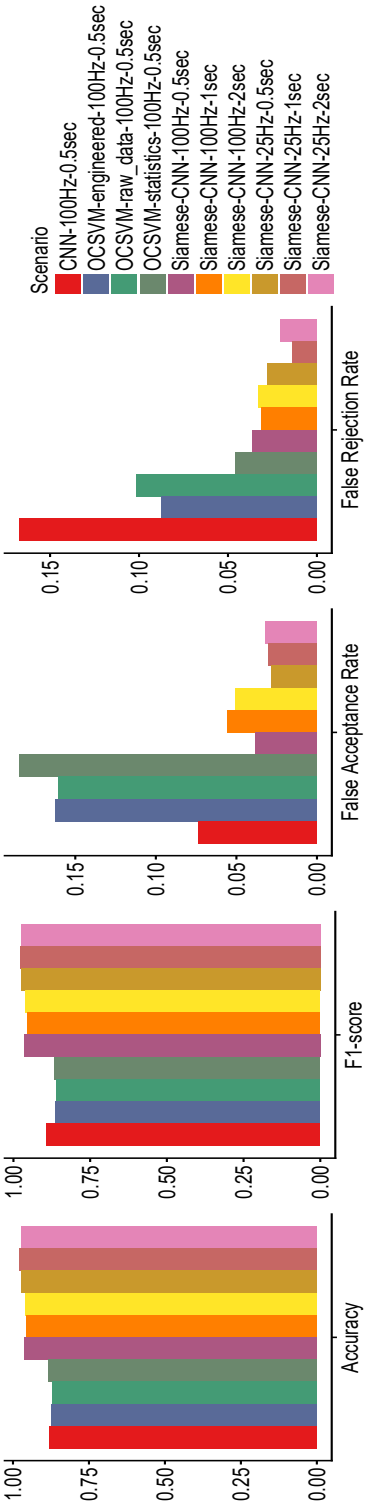


Figure 5.9 Method comparison.

Table 5.1 Confusion matrix of the approach when the feature extraction is done by a Siamese CNN with a sampling frequency of 25 Hz and a window size of 1 second. We show the summation of the results of all the scenarios, including different legitimate and fraudulent users.

	1	-1
1	226916	7987
-1	11745	223155

On the other hand, the accuracy when we perform feature extraction using a conventional CNN is as high as 85.9%. However, we see that, when we transform the input data using the Siamese CNN model, accuracy results are significantly better—using the same sampling frequency and window size, the accuracy rate increases to 94.3%. We can see within the figure that the false acceptance rate decreases significantly (from 9.3% to 5%). Thus, intuitively, the Siamese CNN model can learn representations of the observations from each of the users who are closer to each other in the embedded space, whatever is the activity which is performing the user.

On the other hand, we have evaluated the accuracy of the Siamese CNN model when using different sampling frequencies and window sizes. We can see that when the sampling frequency is 100 Hz, the accuracy rate decreases slightly when increasing the window size, from 94.3% when the window length is 0.5 seconds to 93.8% when it is 2 seconds. When the window size is bigger, the observations will include samples from a higher number of different movements, which could generate representations of the observations which are more widely spread in the embedded data space.

When the input to the model has been sampled with a lower frequency, the accuracy rate is higher, whatever is the window size. When the sampling frequency is 25 Hz, the accuracy is higher when the length of the window is 1 second and lower otherwise. This is the case with the highest accuracy (95.8%) between all the different scenarios (Table 5.1 shows the confusion matrix of the classification results). Nevertheless, these observations indicate that our method is robust regarding the sampling rate and window length, which can reduce the computational burden significantly.

5.6 Conclusion

We propose a continuous authentication biometric system for a smartphone that can detect fraudulent use by exploiting the user's specific motion patterns. We have evaluated the effectiveness of the system with a comprehensive case study.

Throughout the experiments, we have shown that the learning feature extraction process using a Siamese CNN model improves the verification rate, even using a simple novelty detection model (one-class SVM).

Furthermore, we have shown that, by adjusting the parameters of the scenario, sampling frequency and window size, we may be able to improve the effectiveness or efficiency of the authentication system.

Some of the limitations of these work are:

- We have used data from a constraint environment. Although the dataset we have used to perform the experiments included a high number of participants doing a variety of activities, it is not real data. It would be interesting to show the performance with a "wild" dataset.
- We have done some experiments to chosen the neural network architectures, i.e., feedforward, 1d-CNN, and 2d-CNN, but we have not reported them. It would be interesting to make an exhaustive comparison of performance results using a different number of layers and hidden units for the architecture.
- To compare between approaches, we need to choose a threshold to calculate performance accuracy, i.e., we select a threshold for the autoencoder model score output to decide if the instance is legitimate or fraudulent. To show each of the models' results, we have chosen the threshold that maximises the accuracy detection rate, meaning an overestimation of the detection results.

Chapter 6

Unsupervised Machine Learning in Card Payments Fraud Detection Systems

6.1 Introduction

In this chapter, we propose an unsupervised approach which learns the patterns of normal transactions to detect potentially fraudulent transactions. Thus, it can detect previously undiscovered types of fraud. We study several Machine Learning and Deep Learning models i.e. an autoencoder, a multivariate Gaussian Mixture Model and an OC-SVM . We conduct the experiments in a real-world card payments transaction dataset from a European acquirer. Furthermore, we study the importance of transactional attributes and show their effect on detection performance.

In summary, the contributions of this chapter are as follows:

- It provides a comprehensive survey of the state of the art card payment fraud detection systems proposed so far.
- It provides an exhaustive description of the challenges of modelling card payment transactions and solutions to improve the detection of fraud.
- It evaluates the accuracy of different unsupervised approaches on card payments real-world transactions dataset.
- It shows the effect of feature selection in the performance of the detection system.

Table 6.1 Card payment fraud detection approaches based on Unsupervised Machine learning techniques

Author	Year	Techniques	Dataset	Quantitative Results
Aleskerov, E. et al. [9]	1997	Neural Network	sinthetic	Yes
Quah, J. et al. [124]	2008	Self Organized Maps	collected	No
Srivastava, A. et al. [143]	2008	Hidden Markov model	sinthetic	Yes
Bhusari, V. et al. [20]	2011	Hidden Markov model	sinthetic	Yes
KhanT, A. et al. [4]	2011	Hidden Markov model	sinthetic	Yes
Iyer, D. et al. [82]	2011	Hidden Markov model	sinthetic	Yes
Hejazi, M. et al. [79]	2013	one-class SVM	collected	Yes
Bansal, M. et al. [13]	2014	Self Organized Maps	collected	Yes
Tech, V. et al. [150]	2014	K-Menas	sinthetic	No

This chapter is organised as follows. Section 6.2 discusses the traditional fraud detection systems and those based on machine learning techniques proposed so far. Section 6.3 introduces the dataset used to evaluate our approach, describes the feature selection process and discuss the importance of the transaction's attribute. Section 6.4 introduces cost metrics. Section 6.5 discuss the performance of the unsupervised approaches proposed and the trade-off number of attributes-performance. Section 6.6 concludes the chapter.

6.2 Machine Learning approaches for card payments fraud detection

In this section, we discuss some of the machine learning approaches proposed so far for card payments fraud detection. We classify them in supervised and unsupervised systems. The distinction of the group is made based on whether the specific target value to predict is known for the available samples and in the manner that the algorithm is trained.

6.2.1 Supervised learning

The emphasis on card payment fraud detection systems is on supervised classification methods. It is a discriminative technique trained to find previously known fraud patterns. In classification problems, the system scores the input transaction based on similarities with the attributes of the previously seen fraudulent patterns. Depending on whether the score exceeds a predefined threshold, the transaction will be classified, such as legitimate or fraudulent.

Neural Networks (NNs) was one of the first ML techniques use to develop FDS more than 20 years ago, and they have become very popular since then. In 1994, [73] developed a fraud detection system based on a 3-layers P-RCE feedforward network. They used a dataset of transactions processed by Mellon Bank during six months of 1991. The original training dataset was sampled to include 3.33% of fraudulent accounts, and a feature selection process was applied to the original group of attributes. The results showed that when the system flagged 50 accounts as fraudulent per day, 40% of fraudulent transactions were detected. That meant an improvement of the previous operative FDS based on rules. In [25] an FDS was proposed combining a NN with a rule-based approach. They used a dataset of approximately 500K transactions from a German acquirer. After subsampling the number of legitimate transactions, both modules were combined in a unique sequential system, resulting in a 90% TNR performance and 91.5% TPR.

Reference [100] compared the accuracy of an Artificial Neural Network (ANN) with a Bayesian Belief network. They tested the approaches in a real-world dataset, but they did not describe it because of a privacy agreement with the source entity. After feature selection, the best accuracy of the NN was 22.3% ERR. The BBN performed slightly better, detecting 8% more of fraudulent transactions.

Reference [69] compared the performance of an FDS based in a NN with four other systems based in an Artificial Immune Systems (AIS), a Naive Bayes (NB), a Bayesian Network (BN) and a Decision Tree (DT) algorithms. They used a real-world dataset from a Brazilian card issuer. They applied feature selection, and only they consider 17 features from the 33 available. To evaluate the approaches the minimise the monetary losses. The NN and the AIS methods obtained the best accuracy results.

Authors of [14] compared the performance of FDSs based in two different NNs, a Committed Neural Network and a Clustered Committed Neural Network. They tested the approaches in a real-world dataset with almost 4 million legitimate transactions and 1000 fraudulent transactions. For training, they undersample the number of legitimate transactions to 7000. The Clustered Committed network architecture showed better detection results, with the highest accuracy of 91.7% TPR and 82.6% TNR.

More recently, [67] compared the performance of a NN with a Convolutional Neural Network (CNN), a Random Forest (RF) and a Support Vector Machine (SVM). They tested the approach in a real-world dataset from a bank consisting of 260 million transactions (100 were fraudulent). The CNN obtained the best accuracy result.

In [87] a the authors compared the performance of two FDSs based in an RF and a Recurrent Neural Network (RNN) models. They investigated the effectiveness of the approach in two real-world datasets. One of the datasets consisted of e-commerce transactions, and the other one in a face to face transactions. The LSTM performed better on the offline transactions dataset, but there was no difference in both approaches in the online dataset. Furthermore, the authors pointed out that the LSTM model patterns were different from patterns seen for the RF model.

[158] proposed a game-theoretic approach. They model the interaction between an attacker and an FDS, such as a multi-stage game between two players, both trying to maximise financial gain. The model consists of a two-tiered architecture, being the first layer of defence a set of rules. They tested the approach with volunteers. Most of them were not able to learn the game's strategy, i.e., the established set of rules, but 40% of them were able to do so.

In 2010 [19] compare the accuracy of an SVM, an RF, and LR models. They tested the approaches in a real-world dataset from an international credit card issuer, including 50 million transactions (with 2420 fraudulent transactions from 506 cards). They created 16 derived features from the original ones, i.e., Average daily over a month. RF obtained the highest accuracy with a 78% F-score, followed by LR with 70% and SVM with 62%.

In [133] the authors compared the effectiveness of two FDSs based in an SVM and DT algorithms. The dataset was the same used in [19]. DT obtained the best accuracy rate with approximately 95% while SVM got 93%.

[166] conducted a study to show whether transaction aggregation may improve the fraud detection rate. The analysis showed that RF, LR, SVM, KNN and Quadratic Discriminant (QDA) improve their accuracy with aggregation. However, DT (CART) did not. They showed the result in two independent datasets from two banks. In both analyses, QDA obtained the highest detection accuracy.

Machine Learning (ML) techniques have demonstrated to be useful to detect fraudulent payments transactions, but keeping a low FAR with a high detection rate is a difficult task. We have seen that FDS has a high FAR when keeping a high detection rate, [18]. FAR has a high impact on the effectiveness of the system. It has associated a cost and customer relations are directly affected.

On the other hand, one characteristic present on all the real card fraud transactional datasets used to train the model is that they are very imbalanced. The percentage of fraudulent trans-

actions is extremely lower than that for legitimate transactions. Usually, the percentage of fraudulent transactions is just between 0.1% and 0.5% [25]. In this scenario, misclassification arises because of the difficulty of the FDS to learn the fraud patterns.

6.2.2 Unsupervised learning

One of the main advantages of using unsupervised techniques in card fraud detection systems is the possibility of found undiscovered fraudulent patterns. However, approaches for card fraud detection systems based on unsupervised techniques are less common.

In 1997, an FDS based in an auto-associative NN was proposed in [9]. Differently from the FDS's based on Supervised Neural Networks proposed in [73] and [25], this model was trained only with legitimate transactions (300 samples). They test the approach in a synthetic dataset generated with a Gaussian model. Each transaction consists of four attributes, and the rate of normal samples was 5 : 1. The results of the test showed that the system classified all the legitimate transactions correctly and misclassified 15.09% of the fraudulent transactions. The limitation of this system is that they used one network per customer, and they tested the approach only in synthetic data simulated from a Gaussian distribution.

In 2008, an FDS based on a Hidden Markov Model (HMM) was proposed in [143]. Same that in [9], this FDS created a spending habit model for each cardholder. The category of items purchased was represented as the underlying finite Markov chain. The transactions were observed through the stochastic process that produces the sequence of the amount of money spent on each transaction. The observation symbols were defined, clustering the purchase values of the historical transactions of each cardholder. They were clustered in three price ranges low, medium and high. They tested the system in a synthetic dataset. The test results showed the best result of 80% of accuracy.

In 2014, [13]compared two approaches based on SOM and decision tree algorithms. The approaches clustered the data into four groups low, high, risky and highly risky. Both methods were tested in four datasets, including 500, 100, 1500 and 2000 transactions (not more information was specified about the data). SOM had slightly better FPR (23.52% and 28% respectively) and 20% better TPR than decision tree (92.5% and 72.5%). Furthermore, the authors conclude that using the longest dataset, FPR improved 50% and TPR 20%.

In the same year, an unsupervised FDS based on a K-Means algorithm was suggested in [150]. The system was tested in a synthetic dataset. Some of the attributes of the dataset

were transaction ID, transaction amount, transaction country, transaction date, credit card number, merchant category id, cluster id and indicative of fraud. The classification classes were the same four groups of the previous approach [13], i.e., low, high, risky and high risky. The authors did not show the results quantitatively.

Table 6.1 synthesises the main aspects, i.e. authors, technique, type of dataset and analysis of quantitative results for each of the unsupervised approaches reviewed in this section. We can see that only two of the approaches [79, 13] were tested in real-world data while showing quantitative results. Between these two approaches, the one based on the OC-SVM model [79] achieved a higher accuracy result, i.e. 93%.

6.3 Dataset

A card fraudulent transactional dataset is a group of m transactions T:

$$\mathbf{T} = (\mathbf{t}_1, ..., \mathbf{t}_m) \tag{6.1}$$

Each transaction can be seen as a data tuple of n attributes a:

$$\mathbf{t}_i = (a_1, ..., a_d) \tag{6.2}$$

We find that datasets in card FDS literature have between 20 and 50 attributes (numerical and categorical). The range of unique values of attributes can vary from two values up to several hundred thousand [30].

Some common attributes are current transaction description, transaction history description, payment history description and regional descriptors [51, 73]. Attributes documented in real data sets are the date, amount of the transaction, merchant category code (CCC), transaction type [99]. Many times a full list of them is not specified because of confidentiality concerns.

Usually, merchants record a much bigger list of attributes related to the transaction, such as the number of items purchased, but transaction risk profiling systems usually do not have access to them [4]. Some synthetic datasets included personal parameters of the user, such as age and income of the cardholder. When developing an FDS, some authors refuse to use them to build the models of the systems because those are not available in practice [82].

As we have seen, in the card fraud detection literature, only a few publications use real fraud transactional datasets [69]. There are not many card fraud transactional datasets publicly available [163] mainly because of anonymity and security reasons [163], i.e., financial institutions usually do not make public the private information of their customers. Furthermore,

companies are not in a position to share sensitive information with their competitors. On the other hand, usually, reveal information concerned with fraud detection systems is declared to violate a vital security interest.

To show the effectiveness of our approach, we use a publicly available dataset realised for a leader in electronic transactions [121]. The datasets contain transactions made by credit cards by European cardholders. Each transaction consists of 30 features. To preserve the confidentiality of the customers, most of the variables are the principal components transformation of the original values. Only features 'Time' and 'Amount' preserve the original value. 'Time' is the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the economic transaction amount.

Furthermore, each transaction has associated a label which indicates whether the transaction is either legitimate or fraudulent such as:

$$label = \begin{cases} 1, & \text{if } T \text{ is fraudulent} \\ 0, & \text{otherwise} \end{cases}$$

No more extra background information has been given for the rest of the features.

Table 6.2 shows an example of some features for several transactions.

We normalise the feature Time and Amount with the max-min technique introduce in Section 4.6. Furthermore, we transform the feature Time to indicate the hour of the day which the transaction in the following manner:

$$f(t) = \frac{t}{60 * 60} \pmod{n}.$$

Table 6.2 Example of the values of the features of the transactions

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	Amount	Class
0	-1.360	-0.073	2.536	1.378	-0.338	0.462	0.240	0.099	0.364	0.091	-0.552	-0.618	149.62	0
0	1.192	0.266	0.166	0.448	0.060	-0.082	-0.079	0.085	-0.255	-0.167	1.613	1.065	2.69	0
1	-1.358	-1.340	1.773	0.380	-0.503	1.800	0.791	0.248	-1.515	0.208	0.625	0.066	378.66	0
1	-0.966	-0.185	1.793	-0.863	-0.010	1.247	0.238	0.377	-1.387	-0.055	-0.226	0.178	123.50	0
2	-1.158	0.878	1.549	0.403	-0.407	0.096	0.593	-0.271	0.818	0.753	-0.823	0.538	69.99	0
2	-0.426	0.961	1.141	-0.168	0.421	-0.030	0.476	0.260	-0.569	-0.371	1.341	0.360	3.67	0
4	1.230	0.141	0.045	1.203	0.192	0.273	-0.005	0.081	0.465	-0.099	-1.417	-0.154	4.99	0
7	-0.644	1.418	1.074	-0.492	0.949	0.428	1.121	-3.808	0.615	1.249	-0.619	0.291	40.80	0
7	-0.894	0.286	-0.113	-0.272	2.670	3.722	0.370	0.851	-0.392	-0.410	-0.705	-0.110	93.20	0
9	-0.338	1.120	1.044	-0.222	0.499	-0.247	0.652	0.070	-0.737	-0.367	1.018	0.836	3.68	0
10	1.449	-1.176	0.914	-1.376	-1.971	-0.629	-1.423	0.048	-1.720	1.627	1.200	-0.671	7.80	0
10	0.385	0.616	-0.874	-0.094	2.925	3.317	0.470	0.538	-0.559	0.310	-0.259	-0.326	9.99	0
10	1.250	-1.222	0.384	-1.235	-1.485	-0.753	-0.689	-0.227	-2.094	1.324	0.228	-0.243	121.50	0
11	1.069	0.288	0.829	2.713	-0.178	0.338	-0.097	0.116	-0.221	0.460	-0.774	0.323	27.50	0
12	-2.792	-0.328	1.642	1.767	-0.137	0.808	-0.423	-1.907	0.756	1.151	0.845	0.793	58.80	0
12	-0.752	0.345	2.057	-1.469	-1.158	-0.078	-0.609	0.004	-0.436	0.748	-0.794	-0.770	15.99	0
12	1.103	-0.040	1.267	1.289	-0.736	0.288	-0.586	0.189	0.782	-0.268	-0.450	0.937	12.99	0
13	-0.437	0.919	0.925	-0.727	0.916	-0.128	0.708	0.088	-0.665	-0.738	0.324	0.277	0.89	0
14	-5.401	-5.450	1.186	1.736	3.049	-1.763	-1.560	0.161	1.233	0.345	0.917	0.970	46.80	0
15	1.493	-1.029	0.455	-1.438	-1.555	-0.721	-1.081	-0.053	-1.979	1.638	1.078	-0.632	5.00	0
16	0.695	-1.362	1.029	0.834	-1.191	1.309	-0.879	0.445	-0.446	0.569	1.019	1.298	231.71	0
17	0.962	0.328	-0.171	2.109	1.130	1.696	0.108	0.522	-1.191	0.724	1.690	0.407	34.09	0
18	1.167	0.502	-0.067	2.262	0.429	0.089	0.241	0.138	-0.989	0.922	0.745	-0.531	2.28	0
18	0.247	0.278	1.185	-0.093	-1.314	-0.150	-0.946	-1.618	1.544	-0.830	-0.583	0.525	22.75	0
22	-1.947	-0.045	-0.406	-1.013	2.942	2.955	-0.063	0.856	0.050	0.574	-0.081	-0.216	0.89	0
22	-2.074	-0.121	1.322	0.410	0.295	-0.960	0.544	-0.105	0.476	0.149	-0.857	-0.181	26.43	0
23	1.173	0.353	0.284	1.134	-0.173	-0.916	0.369	-0.327	-0.247	-0.046	-0.143	0.979	41.88	0

6.3.1 Imbalanced dataset

The dataset includes transactions that occurred over two days, where 492 out of 284,807 are fraudulent transactions. Note that the dataset is highly unbalanced; the positive class (frauds) account for 0.172% of all transactions.

One characteristic present on all the real card fraud transactional datasets is that they are very imbalanced. The percentage of fraudulent transactions is extremely lower than that for legitimate transactions. Usually, the percentage of fraudulent transactions is just between 0.1% and 0.5% [25]. In this scenario, misclassification arises because of the difficulty of the FDS to learn fraud patterns.

We find some incongruences in the literature when some authors such as [59] state to use real card fraudulent transactional datasets with a ratio of 20% of fraudulent transactions without applying any sampling process.

In this same paper, they studied the influenced by using different distributions to train a system. The system tested was a meta-learning method. The meta-classifier and base classifiers were based on four methods ID3 decision tree, CART decision tree, BAYES and RIPPER. The different distributions fraudulent/legitimate used were: 20%/80%, 25%/75%, 33%/67%, 50%/50% and 67%/33% (note that the original datasets was sampled from 500.000 transactions to 50.000 with the different distributions. 84% of the data was used for the training set, 8% for the meta-learning set and 8% for the testing set). They measure the performance of each combination classifier-distribution for meta-classifiers and base classifiers. They find that indistinctly for meta-classifiers and base classifiers, each algorithm obtained a higher TP rate and lower FP rate when data distribution was 50%/50%.

However, in unsupervised approaches, the goal is to model only normal available data. Thus, we will not have to tackle imbalance during training.

6.3.2 Sampling

1.

2.

3.

Sampling is a very common process when building FDS.. Usually, datasets to train FDS are very skewed, thus, undersampling and oversampling are common techniques to balance datasets in this area cite809570. In [14], we can see an example. The original dataset included 3.728.713 legitimate transactions and 1006 fraudulent transactions. The dataset resulted from the sampling process consisted of 7.210 legitimate transactions and 581 fraudulent transactions. Thus, the percentage of fraudulent transactions changed from 0.02% to 7.45%.

A different approach was followed in [32]. To train the model, the authors created several datasets with different class distributions. . The final model was obtained, averaging the partial results calculated firstly.

-
-

We will conduct a series of experiments to show how it affects sampling to the approaches we propose. Furthermore, we will use sampling to build a sensible-cost system.

6.3.3 Training and testing dataset

We have split the original dataset into training and testing datasets for the experiments we conduct in the next section. The training dataset includes 75% of the legitimate transactions, and the testing dataset consists of the 25% of the legitimate transactions and all the fraudulent transactions. We will repeat the experiments ten times, and each of the time, we will choose randomly the split $75\% - 25\%$ of the legitimate transactions.

6.3.4 Feature selection

Most of the FDS in the literature use a feature selection process because:

- Improve training time: some models are computationally intensive when building the models. If the amount of data that they have to compute is fewer, the time to train the model will be lower.
- Improve response of real-time systems: FDSs are expected to detect fraudulent transactions in real-time. Detection can be faster if the number of attributes of each transaction is low.

The number of attributes of the system can be significantly reduced. For example in [69] the number of attributes was reduced from 33 to 17.

Some of the authors specify which feature selection process they used, such as in [Ush et al.] which based the process in a GA but many of them do not. On the other hand, the information provided for some attributes can be essential for the making decision process. In [73] a comparison of the performance of an FDS using two different groups of attributes is made. One of the groups includes payment-related information and the other not. The results showed that the model trained without payment-related information produced twice as many TP as the model, which includes the related-payment information. Furthermore, accuracy was improved, too, including this extra information.

In [134] a comparison of the impact on the effectiveness of using different groups of features is made. They divided attributes into four groups, history descriptors, regional descriptors, daily amount descriptors and daily account descriptors. When the regional descriptors group of features was not taken into account, the effectiveness of the FDS was almost 80% less. However, the effectiveness of the system only was committed by 5% when the daily account descriptors were not taken into in account, 10% when history descriptors and 22% when daily amount descriptors.

Figure 6.1 shows the relation between the variables of the dataset, which we are using in the present investigation. We can see that more than half of the features are correlated to "amount", "time" and "Class", but they are not correlated between them. On the other hand, as well we can see that the rest of the features are not correlated between them either.

We have used an extra-Trees algorithm to calculate the importance of the features of the dataset such as in [95]. We use the depth of the node assigned to each of the features to calculate the relative importance of that feature. Features on the top of the tree contribute to a higher rate to the final prediction of the model. We used the average between several randomised trees to reduce the variance of the prediction

Figure 6.2 shows the relevance importance of the features obtained. We can observe that features 'V17' and 'V14' are relatively much more informative than other features. Later, we will test the accuracy of the different approaches taking into account different groups of features.

Figures 6.3 and 6.4 show the density function of the two most important features (V17 and V14) and figures 6.5 and 6.6 the less important (V2 and V23). We can observe for the two

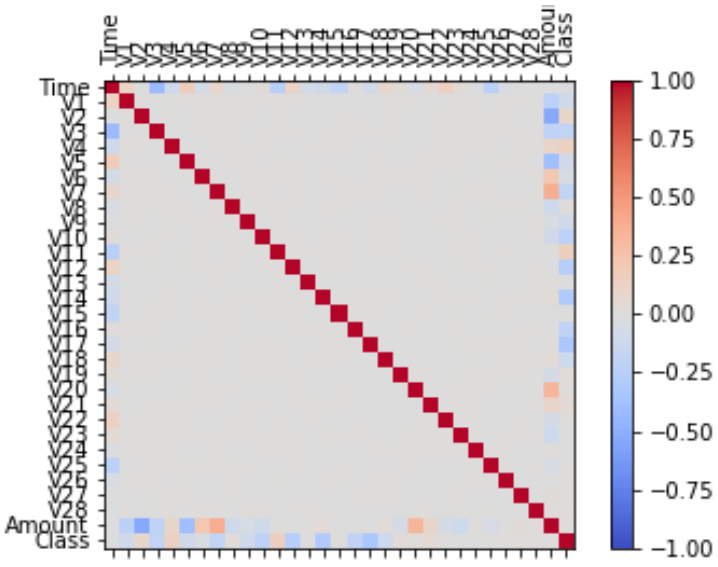


Figure 6.1 Correlation coefficient between figures.

most relevant features, that density functions of fraudulent and non-fraudulent transactions do not overlap as much for those of the less important features.

6.4 Effectiveness metrics

Datasets of electronic transactions are highly imbalanced. As we have said in 6.3.2, these include a much higher number of samples of normal transactions than fraudulent. This characteristic makes essential choosing the sensitive metrics which allow us to show the real detection capabilities of each approach.

Figures 6.7 and 6.8 show the detection results of a fraud detection system in the dataset describe in 6.3 when we refer to the predominant class as negative and positive respectively. The detection system is based on an autoencoder introduced in 2.3.1. The graphs show the ROC and Precision-Recall (PR) curves, accuracy and F-score when varying the similarity threshold (as we explain in the next section) to classify the samples. We can see that the precision-recall curve, accuracy and F-score show a very high performance when we refer to the normal class as positive and very low performance when we refer to them as negative. However, ERR is the same beyond the nomenclature we use.

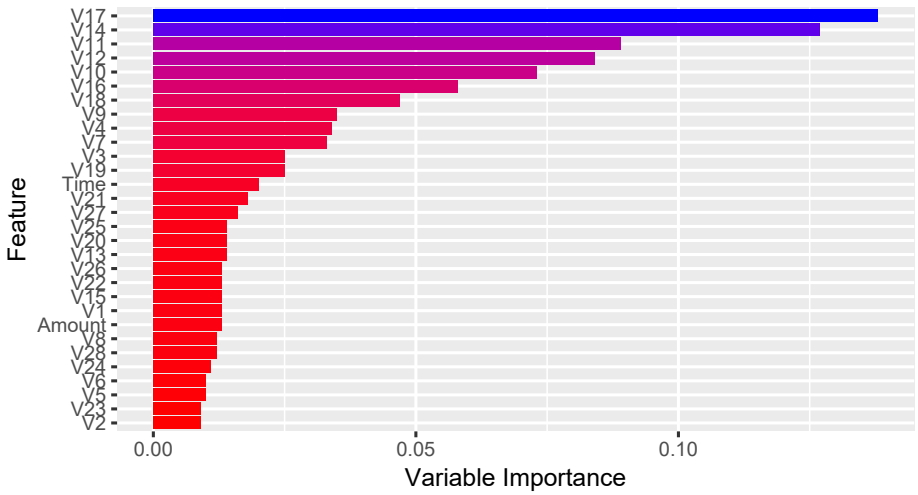


Figure 6.2 Feature importance.

However, some authors state that true positive rate (TPR) 6.5, true negative rate (TNR) 6.6 and false positive rate (FPR) 4.2 are not suitable for card FDS because they establish the same importance for the misclassification of each transaction without taking into account economic losses. For this reason, some authors suggest the implementation of sensible misclassification costs.

A way to do build a sensible cost system is make the misclassification of certain class more expensive. In [70], based in the domain knowledge from fraud preventions specialists and experimental results the authors of the paper set an average cost of \$1 for every verification ($C_{FP} = \$1$) and an average loss of \$100 for every undetected fraud transaction ($C_{FN} = \$100$), resulting in a cost function:

$$\$cost = \$100 * FN + \$1 * (FP + TP) \tag{6.3}$$

Because in this paper they sampled the original dataset available to 10% of legitimate transactions and 100% of fraudulent transactions, they adjust the cost function to:

$$\$cost = \$100 * FN + \$10 * FP + \$1 * TP \tag{6.4}$$

Since in card payments, the amount of each transaction is different, some authors suggest a misclassification cost where the importance of misclassified a transaction depends on the amount saved or lost. In [52] the authors introduce a variable misclassification cost system. We can have a clear view of the metrics throw the confusion matrix shown in fig. 6.9 . FP transactions have a misclassification cost of CFP, which is a fixed cost based on the domain

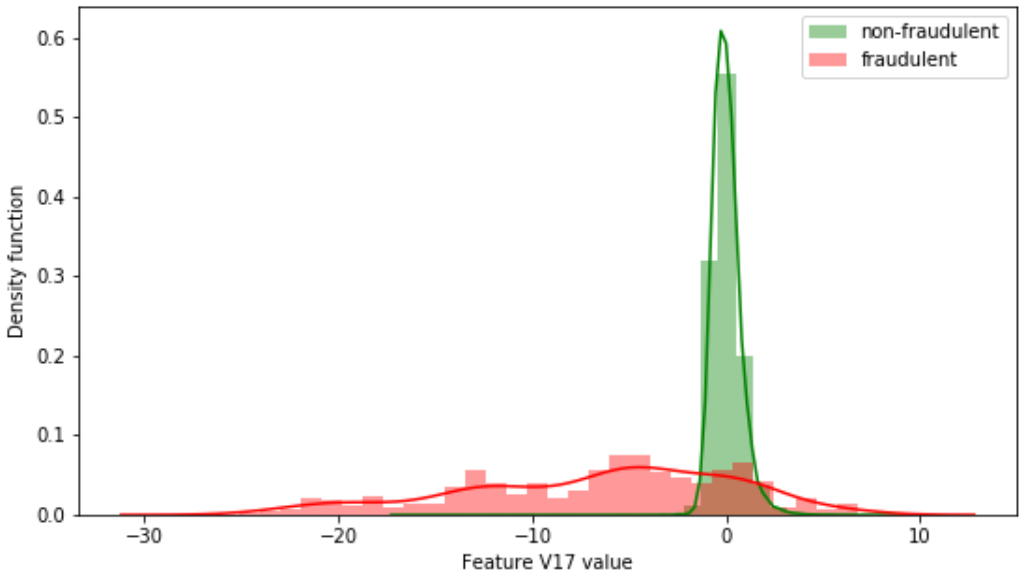


Figure 6.3 Density function of the feature V17.

expertise of the authors. FN transactions have a variable cost equal to the available usable limit of the card in the moment of the transaction.

In [113], the same variable misclassification costs system was used to update an FDS of a Turkey bank. Improvement of the effectiveness of the FDS was shown. When redefining the model, the Bank was not kind to increment the current FP percentage. To achieve this aim was set an extra 50% cost for an FP when the FP rate was up to 150% of the previous FP rate, an extra 100% cost when the FP rate was between 150% and 200% and an extra 500% cost when was bigger than 200%.

Another sensible misclassification cost system can be model sampling the training dataset. Costs of the fraudulent and legitimate transactions conveyed by the appearance of the examples in the dataset [48]. In section s:results, we will sample the dataset according to the economic amount of the transaction and we will evaluate the economic losses due related to the proposed detection system using the following metrics:

$$NegativeClassProfit = \frac{CostTP}{CostTP + CostFN} \tag{6.5}$$

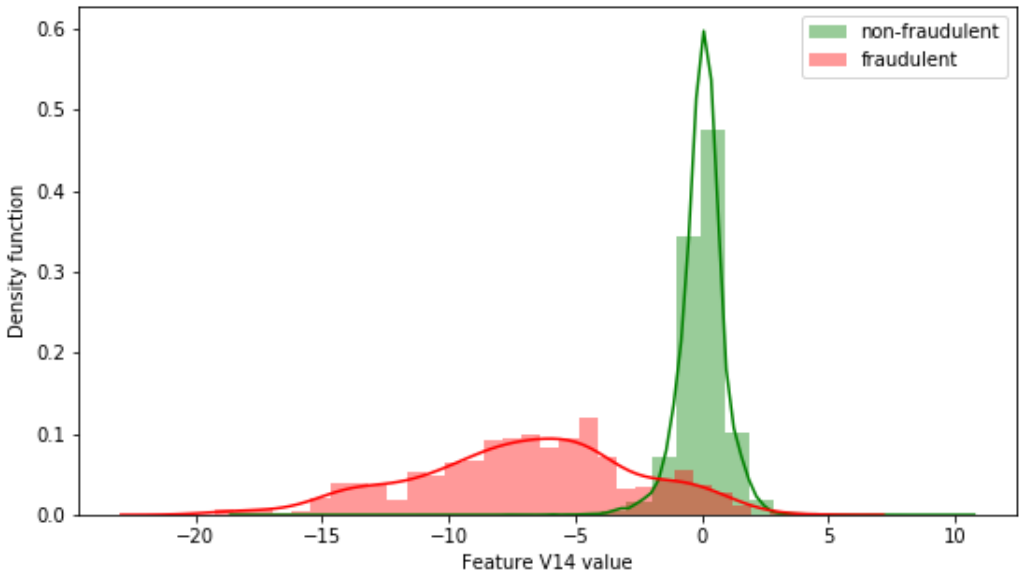


Figure 6.4 Density function of the feature V14.

$$PositiveClassProfit = \frac{CostTN}{CostTN + CostFP} \tag{6.6}$$

6.5 Proposed approaches for card payments FDS

We propose an unsupervised FDS for card payments transactions. We compare the performance between three systems based on a deep learning technique (autoencoder), in a statistical model (a multivariate Gaussian model) and an ML model (OC-SVM, which was previously used in [79] to detect fraudulent payment transactions).

In our system, we use only normal transactions to train the model. It learns the characteristics of the normal samples. Once the model has been trained, each new transaction is classified as normal or fraudulent depending on how similar they are to the learnt patterns.

Deep Learning Autoencoders Giving a card fraudulent transactional dataset $\mathbf{T}$ of m transactions $\mathbf{t}$, the input of the model is the transaction vector $\mathbf{T}$ and the output is that in equation 2.50. In this experiments, we define the activation function as the hyperbolic tangent function [110] and we restrict the degrees of freedom using a tied architecture, where the encoding matrix is the transpose of the decoding matrix, i.e. $\mathbf{W}_d = \mathbf{W}_u^T$ [161].

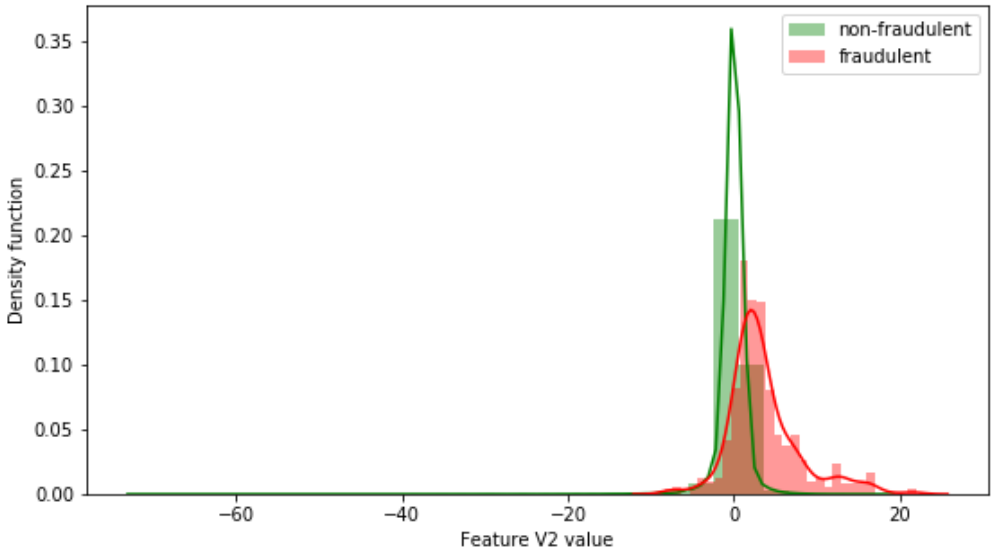


Figure 6.5 Density function of the feature V2.

Furthermore, we test three architectures with one, three and five hidden layers to achieve higher flexibility (and abstraction) of the model.

In the learning process, only data from legit transactions are input to the system, thus learning the model the characteristic patterns of them.

Our approach uses gradient descent and backpropagation to learn the encoder function to obtain the output feature vector $\hat{\mathbf{T}}$. The algorithm is training minimising the mean squared error (MSE). We run the training process for that number of iteration that MSE converge. The MSE gives us the reconstruction loss error value between the input and output features vector, which we call the similarity score for each transaction.

During the detection phase, once we have calculated the match score of the new instance, it is compared against the decision threshold. If the match score is higher than the decision threshold, the authentication request instance will be classified as fraudulent, as shown in 6.10.

Multivariate Gaussian distribution Given the card payments transnational dataset $\mathbf{T} = (\mathbf{t}_1, ..., \mathbf{t}_m)$ we will take into account only those transactions labeled as a normal. We assume that each attribute is normally distributed and we calculate the Gaussian parameters i.e. the mean μ_i and variance σ^2 for each of the features as in 2.52 and 2.53.

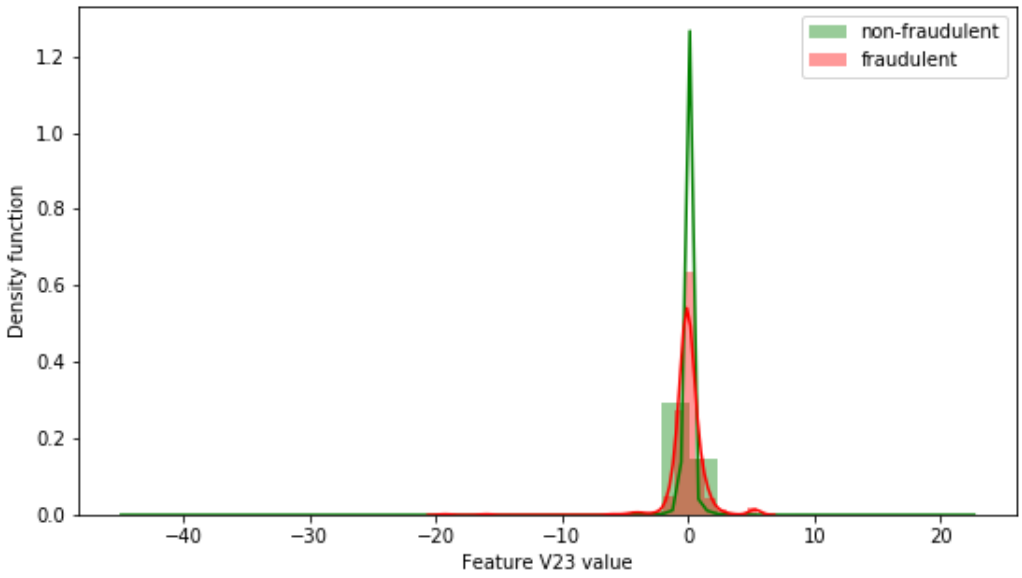


Figure 6.6 Density function of the feature V23.

And we will consider the transaction as fraudulent if $P(\mathbf{t}) < \varepsilon$ where ε is the probability threshold.

Performance comparison of the proposed approaches

In this section, we compare the performance of the two proposed approaches i.e. autoencoder and multivariate Gaussian model, with an approach proposed in [79] which was based in an OC-SVM model. So, we compare the performance of these approaches:

- an autoencoder with one hidden layer and 15 hidden units.
- a multivariate Gaussian model with one component.
- a one-class SVM.

Table 6.3 shows the ERR obtained by each model. We can see that the autoencoder and GMM models get the highest ERR, 9.8% and 9.7% respectively. On the other hand, oc-class SVM obtain the lowest ERR (11.2%).

Figure 6.11 shows the ROC curve of the three different models. The ROC curve of the autoencoder and multivariate Gaussian model are very similar. On the other hand, although

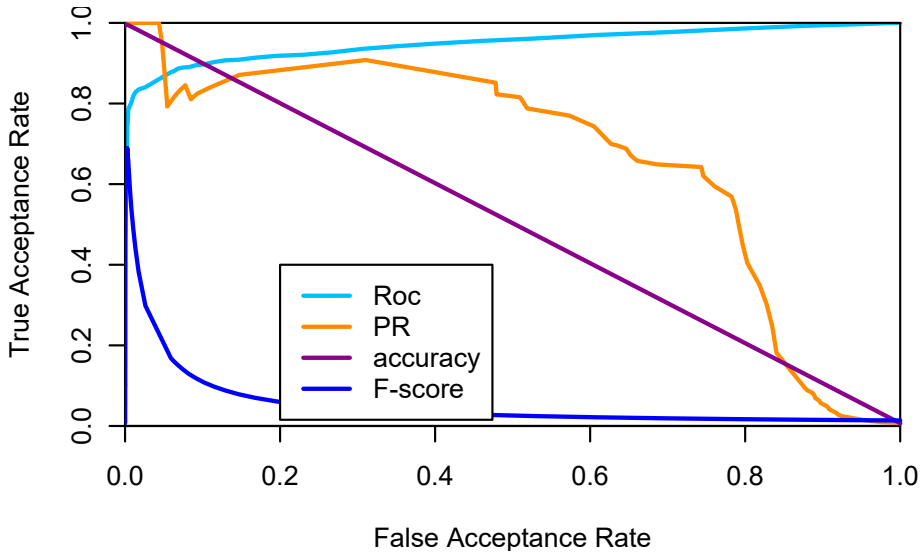


Figure 6.7 Accuracy results when the preominant class is the negative class.

the oc-SVM is the model with the highest ERR, it keeps a higher True Acceptance Rate ($TAR = 1 - FRR$), when the FAR is very small.

Table 6.3 ERR of three different unsupervised models on the dataset shown previously.

Models	ERR
Autoencoder	9.8%
multivariate Gaussian model	9.7%
OC-SVM	11.2%

Deeper and variational autoencoder architectures

Figure 6.12 show the ROC curve of the autoencoder with three different architectures:

- one hidden layers with 15 hidden units
- three hidden layers with 15x7 hidden units
- five hidden layers with 15x30x7 hidden units

We can see that the numbers of hidden layers and hidden units do not affect the accuracy of the model on this dataset. We have repeated the same experiment for the number of hidden units equal to 30, 15, 7 and 3 for each of the layers, obtaining similar results to the shown figure.

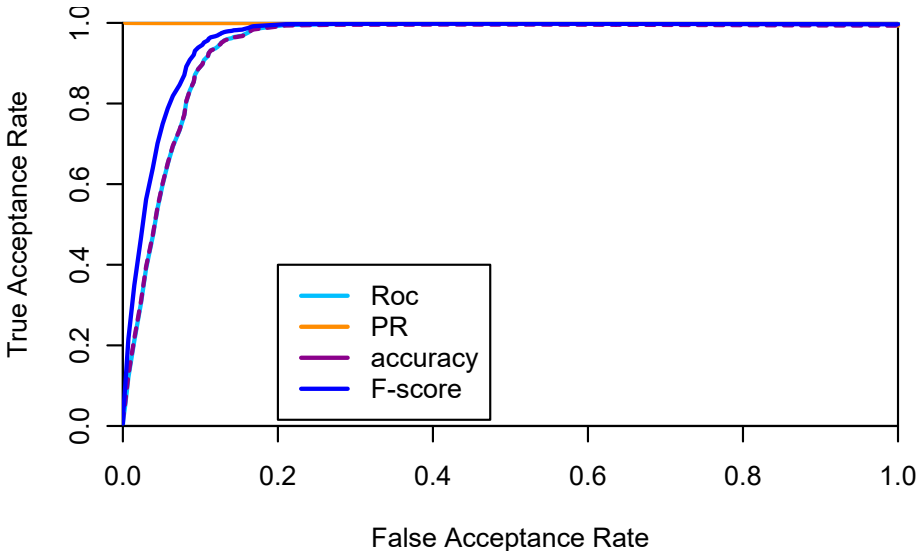


Figure 6.8 Accuracy results when the preominant class is the positive class.

We are going to compare the performance of the autoencoder and a variational autoencoder (VAE) models.

. Figure 6.13 shows the comparison of the performance of the autoencoder and a variational autoencoder (VAE)models. We can see that performance, in this case, is the same for both models, and there is not an improvement of the accuracy when using a VAE model.

6.5.1 Feature importance experiments

We have seen that the autoencoder and multivariate Gaussian model obtained very similar accuracy results. However, deep learning models can manage a high number of input features and transform the input vector of features in a more meaningful lower-dimensional data representation [115]. On the other hand, multivariate Gaussian models work better on low dimensional space problems. Thus, we are going to reduce the input dimensionality space from 30 features to 7 using the five layers autoencoder, i.e. the embedded representation of the middle layer of the autoencoder is used as the input of the multivariate Gaussian model. Figure 6.14 shows the ROC curve of the approach and the ROC curve of the multivariate Gaussian model for comparison. We can see that ERR does not improve, but the autoencoder+multivariate Gaussian model approach keeps higher TAR when FAR is very small.

On the other hand, we are going to test the accuracy of the multivariate Gaussian model when changing the input vector taking into account different numbers of attributes in groups

		prediction outcome		
		p	**n**	**total**
actual value	**p′**	TP $cost = 0$	FN $cost = C_{FN}$	P′
	n′	FP $cost = C_{FP}$	TN $cost = 0$	N′
	total	P	N	

Figure 6.9 Confusion matrix realted to cost.

according to the importance calculate in Section 6.3.4. Figure 6.15 show the ROC curve of the results. We can observe that the model taking into account the two more informative features has the lowest ERR (8.55%) and the model only taken into account the most informative feature has the highest (12.5%).

When the model includes features one by one (by incremental importance), the ERR constantly increase until including 8 features, after that, the ERR increase but not improving the accuracy of the model taking into account four or fewer features (except the model taking into account one feature).

6.5.2 Subsampling

We want to see the effect of the multivariate Gaussian model's performance when subsampling the training dataset by the monetary amount of the transactions. Figure 6.17 shows the ROC graph of the model when considering two features and subsampling the transactions according to their monetary amount, i.e., transactions of an amount higher than 100, 1000, 5000, and 10000 euros. In the graph for comparison, we include the performance results of the model considering two features and all the features without subsampling.

As we saw before, the ERR of the model, when considering two features, is 8.55%. When subsampling the transactions with an amount higher than 100 euros, the ERR increases slightly to 8.65%. However, when subsampling the transactions an amount higher than 1000 euros, the ERR decreases to 8.35%, the highest accuracy over all the experiments we have

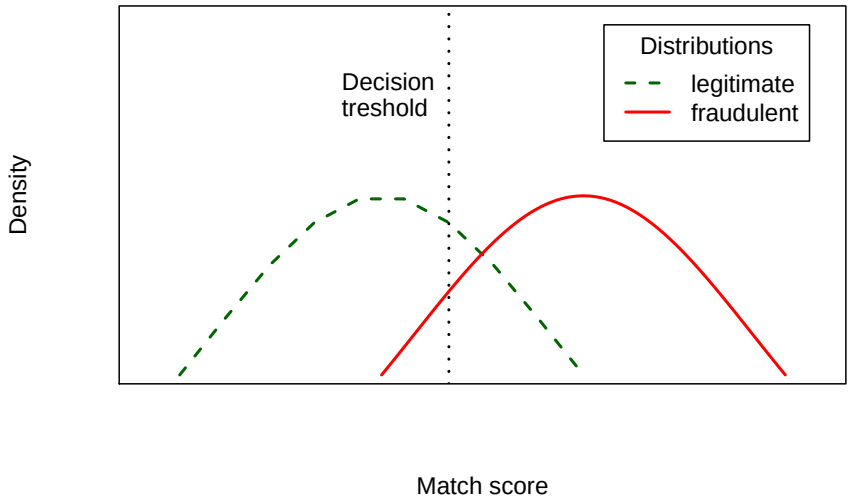


Figure 6.10 An example of the decision threshold for match score distributions from two different users, one legitimate and one fraudulent.

conduct. When subsampling the transactions with an amount higher than 5000 and 10000, the ERR increases to 9.17% and 18.4%, respectively.

So, when sampling the training dataset by the monetary amount of the transactions, we can improve the approach's accuracy slightly, and we do not need to model all the transactions, which computationally could mean a series of advantages such as reducing the computational burden. But we want to know that sampling the training dataset does not mean to increase the monetary losses of fraud. Figure 6.18 shows the Negative Class Losses Rate-Positive Class Profit Rate, i.e., the rate between the monetary amount of the legitimate transaction and the fraudulent transactions. We can see that economic losses are higher for the models:

- taking into account all the features and without subsampling the rate is 18.5%
- taking into account two features and training only with the transactions with amount higher than 10000 the rate is 18.9%

For the rest of the models are very similar with a slightly higher profit with the model taking into account two features and training with transactions higher than 100:

- taking into account two features without subsampling the rate is 13.55%
- taking into account two features subsampling transactions higher than 100 the rate is 13.5%

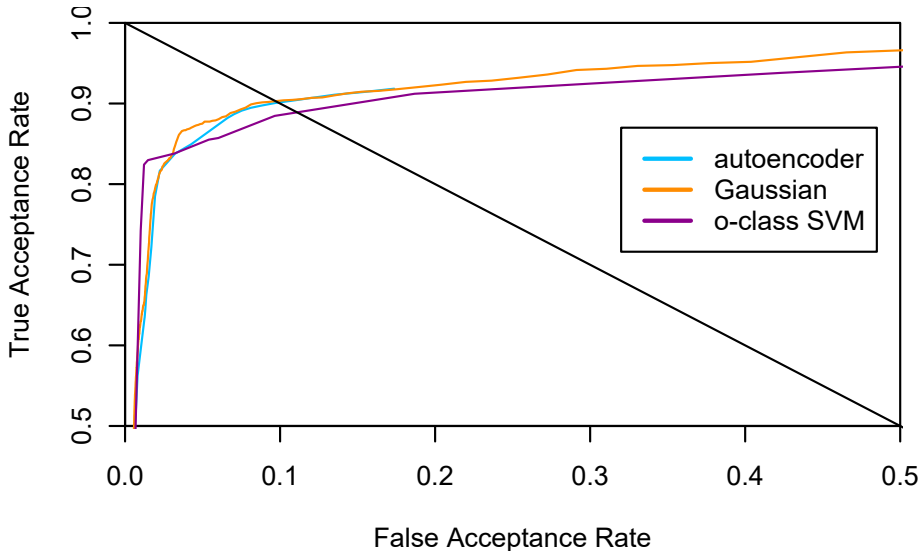


Figure 6.11 ROC curves of the three unsupervised approaches i.e autoencoder,multivariate Gaussian model and OC-SVM used to model normal card payment transactions.

- taking into account two features subsampling transactions higher than 1000 the rate is 13.7%
- taking into account two features subsampling transactions higher than 5000 the rate is 13.85%

Then, when subsampling the transactions with a monetary amount higher than 1000, we improve the ERR of the system, but we have slightly higher losses. But we could sample transactions with a monetary amount of 100. In this case, although the ERR is lower than when not sampling, we will reduce the fraud losses.

6.6 Conclusion

,This chapter has proposed two unsupervised ML approaches to model card payments transactions and detect fraudulent activity. We have seen that both proposed systems based on an autoencoder and a multivariate Gaussian model, have better detection accuracy than a previously proposed approach based in an oc-SVM.

On the other hand, we have shown that the feature extraction and selection ability of deep learning models to build a new data representation of the raw data, does not help to improve the accuracy of the multivariate Gaussian model. However, we have demonstrated that only

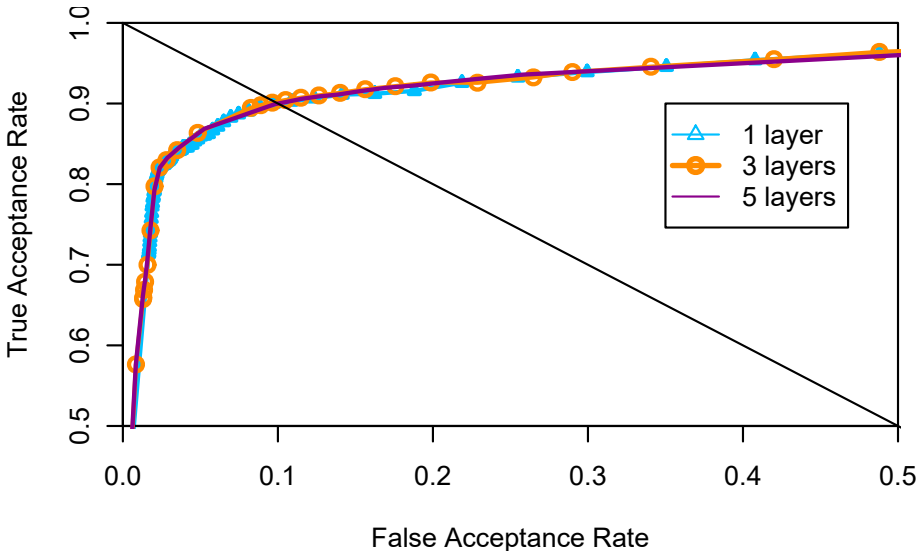


Figure 6.12 ROC curves of the autoencoder model with different number of layers.

taking into account the two most important attributes selected by a tree model; we improve the detection accuracy of the model.

,Furthermore, we have shown that sampling the training dataset beyond advantages such as reduce the computational burden of the model, It helps to improve the ERR of the model and decrease the monetary fraud losses

Some of the limitations of the experiments showed here are:

- ,We have used a public dataset which a limited number of transactions. Furthermore, we do not know the name of many of the features. Knowing the name, we could compare the results more exhaustively with other works, and we will be able to propose better solutions
- ,For the autoencoder architecture, we have chosen the number of hidden units after an experimental exploration. It would be interesting in future work to compare and report differences in accuracy when using architectures with a different number of hidden units.

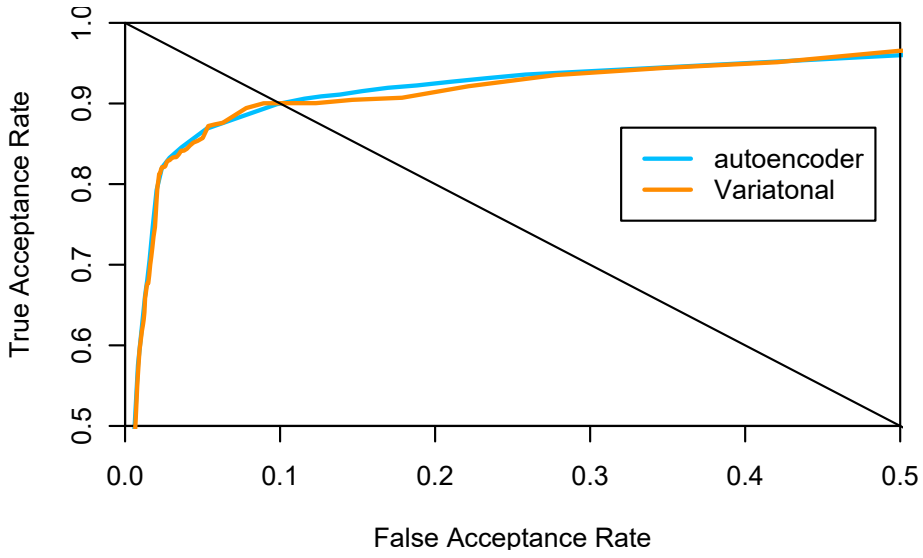


Figure 6.13 ROC curves of different autoencoders.

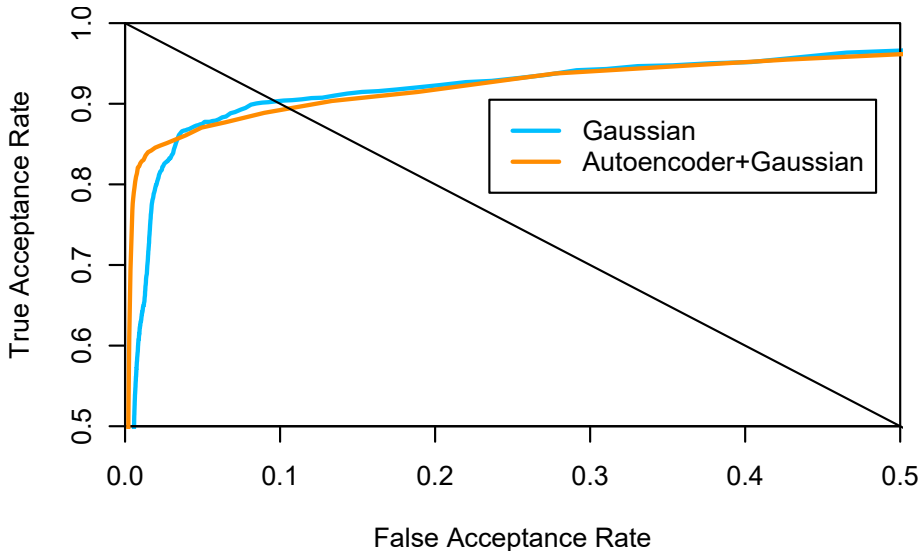


Figure 6.14 ROC curve of the model autoencoder+multivariate Gaussian model train end to end.

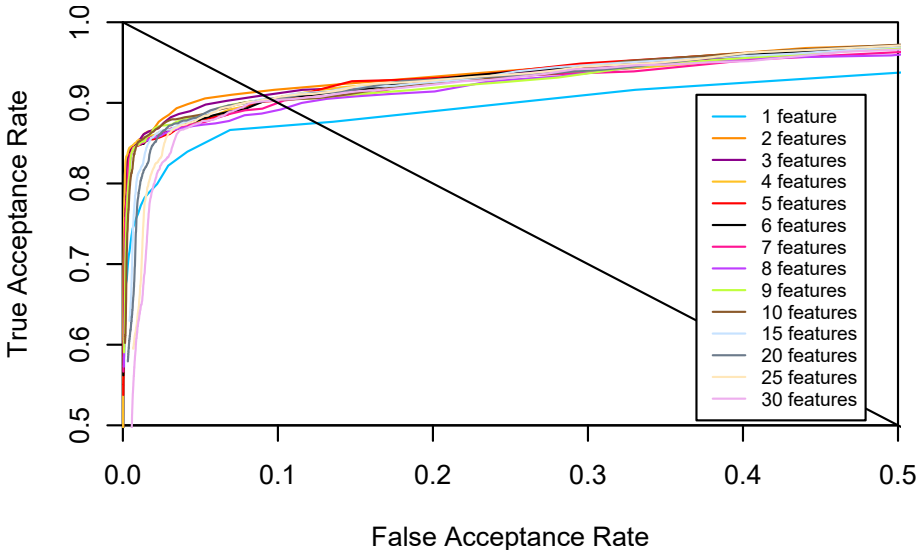


Figure 6.15 ROC curves of multivariate Gaussian models taking into account different groups of features by importance.

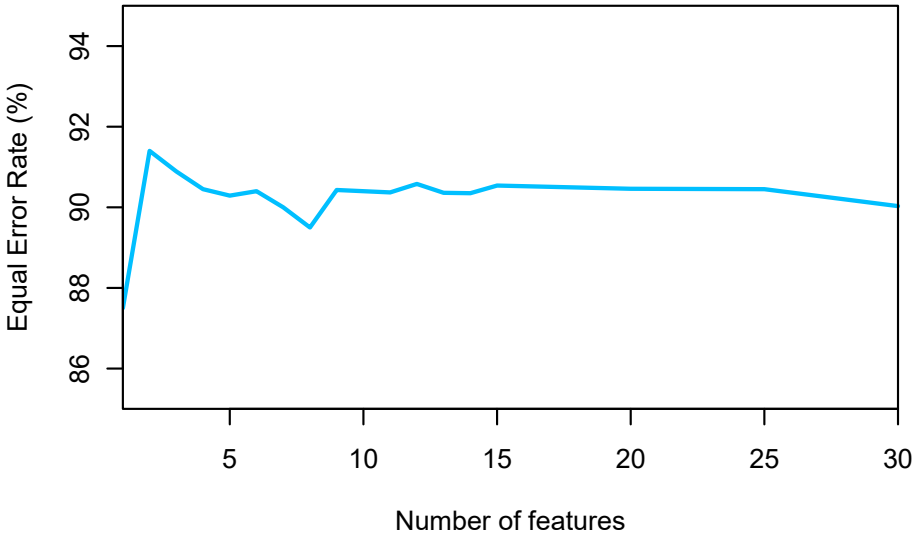


Figure 6.16 ERR the of multivariate Gaussian models taking into account different groups of features by importance.

# features	ERR
1	87.5%
2	91.45%
3	90.89%
4	90.45%
5	90.29%
6	90.4%
7	90.0%
8	89.5%
9	90.43%
10	90.4%
11	90.37%
12	90.58%
13	90.36%
14	90.35%
15	90.54%
20	90.46%
25	90.45%
30	90.03%

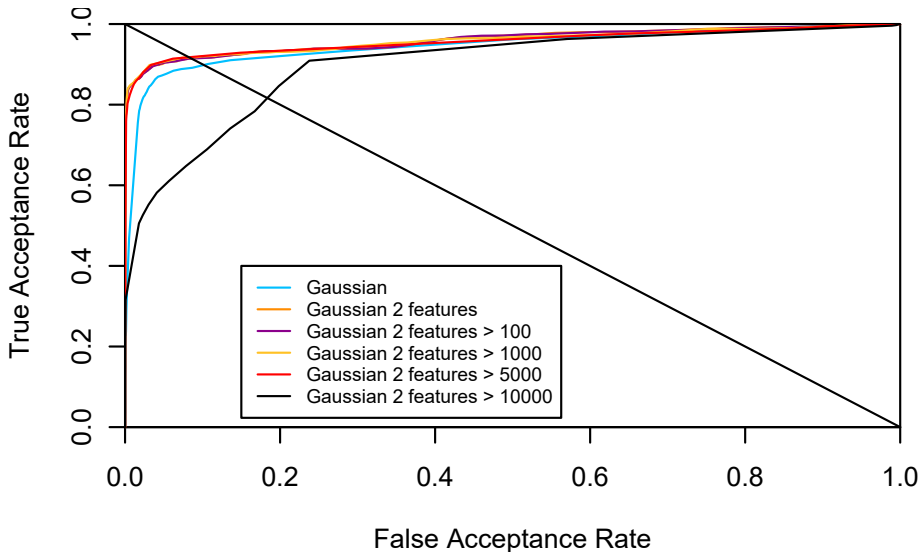


Figure 6.17 Cost ERR the of multivariate Gaussian models taking into account two features and sumsampling the training dataset by amount of the transactions.

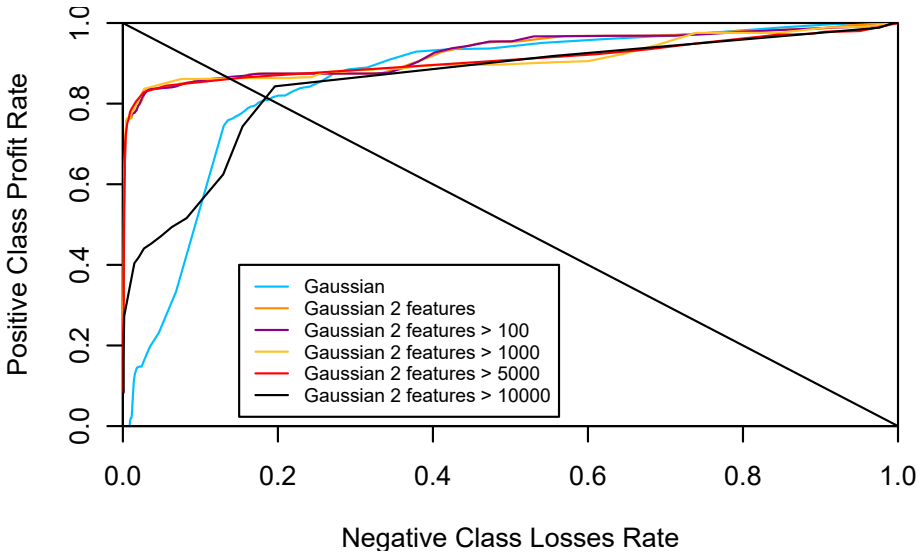


Figure 6.18 Cost ERR the of multivariate Gaussian models taking into account two features and sumsampling the training dataset by amount of the transactions.

Chapter 7

Conclusion

In this chapter, we summarise the research work presented in this thesis on investigating approaches to enhance the security of the electronic card payment system. We outline the contributions and limitations of these works, and discuss open research problems in the field, motivating a number of future work directions.

7.1 Thesis Summary

In this thesis, we have investigated enhancing the security of the electronic card payment system.

Making an exhaustive analysis of the card payment systems, we can immediately realise that they evolve very fast through time. The main actors involved in transactions, i.e., the merchants, the acquirers, and the issuers, continuously adapt new technologies to the systems to offer new services to cardholders. Cardholders quickly habituate to the latest advances, and the popularity of card payment systems has increased exponentially for the last 20 years.

Analysing the losses because of fraud in the same period, we observed a similar increasing trend to the higher use and expenditure of card payment transactions, but understanding the former event's causation becomes essential to be able to reduce the damage. Our investigation highlights that the increment of payment technology's utilisation does not cause an increment in fraudulent activities.

In recent years, we have identified the use of mobile devices and data breaches, such as the current main contributors to fraud losses on electronic card payments. We have investigated

several approaches that consider both aspects, and we developed all the approaches using Machine Learning techniques.

It seems that improving authentication approaches for mobile devices should be urgent to battle current fraudulent activities in payment systems. We have proposed approaches for two different scenarios. An identification approach is useful in scenarios where a group of people shares the same device and a verification approach to certify the user's identity.

In Chapter 4, we have proposed an identification user system for Smartphones. We have based the system on motion data, i.e., accelerometer, magnetometer, and gyroscope data. We compared several approaches based on distinct supervised ML techniques. We have shown that it is possible to distinguish between users very accurately based on motion patterns. Also, we have investigated different technical aspects of the approach. We concluded that specific normalisation helps each approach to perform better, i.e., LR and SVM perform better when standardising and neural networks when scaling. Smaller window sizes improve the accuracy rates, contrary to sampling with smaller frequencies. And we have considered using a different combination of motion data sources to establish than using data from more sensors improves the accuracy of the model, but the improvement is minimal when using data from three sensors instead of two, but with the extra computational burden for the system.

In Chapter 5, we have proposed a verification system for Smartphones. The system was based on motion data as well. We demonstrated that by transforming the raw data in a more meaningful representation using a deep learning technique, we could substantially increase the identification system's accuracy. For this, we have compared the verification system's performance when conducting feature extraction, i.e., extracting some statistics from the instance, extracting features with a CNN, and with a siamese neural network. We demonstrated that the siamese NN architecture, based on a 2D-CNN, cluster more meaningfully the samples from the different users in the latent space, which leads to better classification accuracy. The Siamese CNN model can learn representations of the observations from each of the users who are closer to each other in the embedded space, whatever is the activity that is performing the user. Here too, we compared different technical aspects of the approach. Similar to the results in Chapter 4, reducing the window size increase the performance. Although bigger window sizes should include longer movements, it seems that to differentiate between users, micro-movements characterised better the person.

Demonstrating a successful effectiveness rate of the proposed authentication approaches and because they have been based on motion data allows us to use them as a continuous

authentication approaches. Integrating security measures without the requirement of explicit actions from the users, we manage to improve the user experience. We have shown that with relatively small windows sizes, i.e., from 0.5 to 2 seconds, It is possible to get high accuracy detection rates, and it means that we can identify very frequently and continuously the users.

Motion data can be easily acquired from merchants, but issuers hold millions of transaction records which they can use to identify users. Chapter 6 investigates the development of a card payments fraud detection system that learns from past data to detect fraudulent transactions. If a data breach made public access credentials such as passwords, we could still look at the user transactional, behavioural profile to recognise disparities with their everyday conduct. Similar approaches were proposed in the literature before, but we based the system on an unsupervised Machine Learning technique instead of the supervised, which have been habitual. Thus, the system may recognise unseen patterns of fraud instead of searching only for those who have already been caught up. We compared the performance of two systems based in an autoencoder and a multivariate Gaussian model with an approach proposed previously in the literature, based in an oc-SVM. The two proposed methods had better accuracy than the proposed so far. The initial results showed that both proposed approaches got the same detection performance. After conducting feature selection to train the multivariate Gaussian model, the autoencoder performs it intrinsically; the former performs better. However, at this point, we thought that the ultimate goal of a transactional fraud detection system is to reduce economic losses, beyond that loss of reputation and loss play a crucial role in banking. Thus, we investigated and showed that by training the model with a subsampled dataset, we could keep the same economic losses reducing the computational burden.

7.2 Limitations and Future research directions

Here we motivate a number of areas of future research arising from lessons learned throughout the PhD.

7.2.1 Unconstrained environment

We have tested the identification approach proposed in Chapter 4, and the verification approach proposed in Chapter 5 with a dataset wich only contains motion samples from users performing a specific activity. Although the results on motion identification and verification

are promising, both approaches should be tested in an unconstrained environment, e.g., based on daily activities.

7.2.2 Multi-biometric system

We have seen that including data from more sensors in our identification and verification approaches improves the accuracy of the systems. It would be interesting to see if we can continue improving the performance, including data from other sensors.

7.2.3 Collection new transactional features

The accuracy of our card payments fraud detection system seems to be limited because of the scarcity of the features recorded. It would be interesting testing if collecting new features could help to detect more fraudulent transactions.

7.2.4 Merging our approaches

After seeing that motion data is useful to recognise users and that it is possible to learn the pattern of normal transactions, It would be interesting to see if merging the authentication score and the fraudulent transactional score helps to detect fraud.

7.2.5 Others transactional dataset

The transactional dataset consists of real transactions from a leading European acquirer. However, the dataset only includes transactions from two consecutive days, and values and names of most of the features have been to preserve some confidentiality. It is very complicated to have access to real transactional datasets, but testing the results in another bigger dataset will reinforce our conclusions. Furthermore, knowing the name of all the attributes will help better understand the current fraud detection systems' strengths and weaknesses and design new ones.

7.2.6 Testing Neural Networks architecture effect

In Chapters 4,5 and 6, we have tested several models based on neural network methods, i.e., feedforward neural network, CNN, and RNN. We chose the specifications of the, i.e., number of layers and number of hidden units after some experimentation, but in the future

wok will be interesting to see an exhaustive comparison reporting and comparing the effects in performance when using different architectures.

7.2.7 Overestimation of the results

In the experiments of this thesis, we have chosen the parameters of the models based on the test performance. For this reason, some of the reported results through the thesis are over-optimistic. To obtain a more unbiased estimate of predictive performance, a methodology such as nested cross-validation should be used.

Bibliography

[1] (2015). Contactless card limit rises to £30 as card use surges. In *BBC News*.

[2] (2017). The Nilson Report on Card Fraud. Technical report.

[3] (2018). Statista - U.S. payment card fraud losses by type 2018.

[4] A., K. and A., S. (2011). Credit card fraud detection using hidden markov model. In *Information and Communication Technologies (WICT), 2011 World Congress on*, volume 2, pages 1062–1066.

[5] Aditium (2014). Explorando la tarjeta Contactless. In *S Lab*.

[6] Ahmad, B. I., Langdon, P. M., Liang, J., Godsill, S. J., Delgado, M., and Popham, T. (2019). Driver and passenger identification from smartphone data. *IEEE Transactions on Intelligent Transportation Systems*, 20(4):1278–1288.

[7] Akariman, Q., Jati, A., and Novianty, A. (2015a). Face recognition based on the android device using lbp algorithm. pages 166–170.

[8] Akariman, Q., Jati, A., and Novianty, A. (2015b). Face recognition based on the android device using lbp algorithm. pages 166–170.

[9] Aleskerov, E., Freisleben, B., and Rao, B. (1997). CARDWATCH: a neural network based database mining system for credit card fraud detection. In *Proc. IEEE/IAFE 1997 Comput. Intell. Financ. Eng.*, pages 220–226. IEEE.

[10] Angulo, J. and Wästlund, E. (2012). Exploring touch-screen biometrics for user identification on smart phones. In Camenisch, J., Crispo, B., Fischer-Hübner, S., Leenes, R., and Russello, G., editors, *Privacy and Identity Management for Life*, pages 130–143, Berlin, Heidelberg. Springer Berlin Heidelberg.

[11] Appalaraju, S. and Chaoji, V. (2017). Image similarity using Deep CNN and Curriculum Learning. *Proc. 2017 Grace Hopper India Annu. Conf.*, pages 1–9.

[12] Ashford, W. (2016). More than one in 10 employees fall for social engineering attacks. In *Computer Weekly*.

[13] Bansal, M. (2014). Credit card fraud detection using self organised map. In *International Journal of Information and Computation Technology*, volume 4, pages 1343–1348.

[14] Bekirev, A. S., Klimov, V. V., Kuzin, M. V., and Shchukin, B. A. (2015). Payment card fraud detection using neural network committee and clustering. *Optical Memory and Neural Networks*, 24(3):193–200.

[15] Ben-Shabat, Y., Lindenbaum, M., and Fischer, A. (2018). 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152.

[16] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. In *IEEE Trans. Neural Networks*, volume 5, pages 157–166.

[17] Benmakrouha, F., Hespel, C., and Monnier, E. (2010). An algorithm for rule selection on fuzzy rule-based systems applied to the treatment of diabetics and detection of fraud in electronic payment. In *International Conference on Fuzzy Systems*, pages 1–5.

[18] Benson Edwin Raj, S. and Annie Portia, A. (2011). Analysis on credit card fraud detection methods. In *2011 Int. Conf. Comput. Commun. Electr. Technol.*, pages 152–156. IEEE.

[19] Bhattacharyya, S., Jha, S., Tharakunnel, K., and Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3):602 – 613. On quantitative methods for detection of financial fraud.

[20] Bhusari, V. and Patil, S. (2011). Application of hidden markov model in credit card fraud detection. 2.

[21] Bo, C., Zhang, L., Jung, T., Han, J., Li, X.-Y., and Wang, Y. (2014). Continuous User Identification via Touch and Movement Behavioral Biometrics. *Performance Computing and Commun.Conf. (IPCCC), IEEE Int..*, (61428203):1–8.

[22] Boles, W. (1998). A security system based on human iris identification using wavelet transform. *Engineering Applications of Artificial Intelligence*, 11(1):77 – 85.

[23] Boodaei, M. (2011). Mobile Users 3 Times More Vulnerable to Phishing Attacks.

[24] Braintree, K. T. F. P. (2018). MOBILE PAYMENTS 2018 Report. Technical report.

[25] Brause, R., Langsdorf, T., and Hepp, M. (1999). Neural data mining for credit card fraud detection. *Proc. 11th Int. Conf. Tools with Artif. Intell.*, pages 103–106.

[26] Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., Lecun, Y., Moore, C., Säckinger, E., and Shah, R. (1993). Signature Verification Using a Siamese Time Delay Neural Network. *Int. J. Pattern Recognit. Artif. Intell.*, 07(04):669–688.

[27] Budiu, R. (2015). Mobile User Experience: Limitations and Strengths. In *Nielsen Norman Group*.

[28] business-driven security, R. (2018). RSA Quaterly fraud report Q1 2018. Technical Report 1.

[29] Capgemini; BNP Paribas (2017). World payments report 2017. Technical report.

[30] Carneiro, E. M., Dias, L. A. V., d. Cunha, A. M., and Mialaret, L. F. S. (2015). Cluster analysis and artificial neural networks: A case study in credit card fraud detection. In *2015 12th International Conference on Information Technology - New Generations*, pages 122–126.

[31] Centeno, M. P., v. Moorsel, A., and Castruccio, S. (2017). Smartphone continuous authentication using deep learning autoencoders. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 147–1478.

[32] Chan, P. K., Fan, W., Prodromidis, A. L., and Stolfo, S. J. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and their Applications*, 14(6):67–74.

[33] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 1:539–546.

[34] Cilia, D. and Inguanez, F. (2018). Multi-model authentication using keystroke dynamics for smartphones. In *2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pages 1–6.

[35] Conti, M., Zachia-Zlatea, I., and Crispo, B. (2011). Mind how you answer me!

[36] Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Mach. Learn.*, 20(3):273–297.

[37] Council of the European Union (2017). EU Policy Cycle for Organised and Serious International Crime 2018/2021.

[38] Dahlberg, T., Mallat, N., Ondrus, J., and Zmijewska, A. (2008). Past, present and future of mobile payments research: A literature review. *Electronic Commerce Research and Applications*, 7(2):165 – 181. Special Section: Research Advances for the Mobile Payments Arena.

[39] Daniels, M. (2019). Digital Media Report 2019 The Digital Market Outlook provides all insights for a deep understanding of the Digital Media market. (December 2018).

[40] Dasgupta, D. (1997). Artificial neural networks and artificial immune systems: similarities and differences. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 1, pages 873–878 vol.1.

[41] Datta, T. and Manousakis, K. (2015). Using svm for user profiling for autonomous smartphone authentication. In *2015 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pages 1–5.

[42] De Luca, A., Hang, A., Brudy, F., Lindner, C., and Hussmann, H. (2012). Touch me once and I know it's you! Implicit authentication based on touch screen patterns. *Chi 2012*, pages 987–996.

[43] De Luca, A., von Zezschwitz, E., Nguyen, N. D. H., Maurer, M.-E., Rubegni, E., Scipioni, M. P., and Langheinrich, M. (2013). Back-of-device authentication on smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2389–2398, New York, NY, USA. ACM.

[44] Delamaire, L., Abdou, H., and Pointon, J. (2009). Credit card fraud and detection techniques: A review. *Banks and Bank Systems*, 4.

[45] Demeter, B. (2012). Amazon Won't Mind If You Use a Stolen Contactless Credit Card. *Credit Card Assist*.

[46] Dhamija, R., Tygar, J. D., and Hearst, M. (2006). Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM.

[47] Dodge, J. (2010). Data breach. *Health data management*, 18(9).

[48] Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 155–164, New York, NY, USA. ACM.

[49] Dreiseitl, S. and Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5):352 – 359.

[50] Du, Y. E. (2011). Biometric Technologies. *Encyclopedia of Information Science and Technology, Second Edition*, pages 369–374.

[51] Duman, E. and Ozcelik, M. H. (2011). Detecting credit card fraud by genetic algorithm and scatter search. *Expert Syst. Appl.*, 38(10):13057–13063.

[52] Duman, Y. S. and E. (2011). Detecting Credit Card Fraud by Decision Trees and Support Vector Machines. 1:422–447.

[53] Dunphy, P., Heiner, A. P., and Asokan, N. (2010). A closer look at recognition-based graphical passwords on mobile devices. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, SOUPS '10, pages 3:1–3:12, New York, NY, USA. ACM.

[54] Egelman, S., Cranor, L. F., and Hong, J. (2008). You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1065–1074. ACM.

[55] Elizondo, D. (2006). The linear separability problem: some testing methods. *IEEE Transactions on Neural Networks*, 17(2):330–344.

[56] Etaher, N., Weir, G. R., and Alazab, M. (2015). From zeus to zitmo: Trends in banking malware. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 1386–1391. IEEE.

[57] European Central Bank (2018). Fifth report on card fraud. Technical report.

[58] European Comission (2018). Payment services (PSD 2) - Directive (EU) 2015/2366.

[59] FanAndreas, S. J. S. W. and ScienceFlorida, P. K. C. (1998). Agent-based fraud and intrusiondetection in financial informationsystems.

[60] Felt, A. and Wagner, D. (2011). Phishing on mobile devices. *presentation at W2SP: Web*, 2:1–10.

[61] Feng, T., Liu, Z., Kwon, K. A., Shi, W., Carbunar, B., Jiang, Y., and Nguyen, N. (2012). Continuous mobile authentication using touchscreen gestures. In *2012 IEEE Conference on Technologies for Homeland Security (HST)*, pages 451–456.

[62] FICO (2018). EVOLUTION OF CARD FRAUD IN EUROPE 2016.

[63] Financial Fraud Action UK (2010a). FRAUD The Facts - 2010 -The Definitive Overview of Payment Industry Fraud and Measures to improve it. Technical report, Financial Fraud Action UK, London.

[64] Financial Fraud Action UK (2010b). FRAUD The Facts - 2019 -The Definitive Overview of Payment Industry Fraud . Technical report, Financial Fraud Action UK, London.

[65] Financialfraudactionuk (2016). THE DEFINITIVE OVERVIEW OF PAYMENT INDUSTRY FRAUD.

[66] Fitzgerald, K. (2018). Data: The new causes of mobile payments fraud.

[67] Fu, K. et al. (2016). Credit card fraud detection using convolutional neural networks. In *Neural Information Processing*, Cham. Springer International Publishing.

[68] Fukuda, T., Morimoto, Y., Morishita, S., and Tokuyama, T. (1996). Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 13–23, New York, NY, USA. ACM.

[69] Gadi, M. F. et al. (2008a). Credit card fraud detection with artificial immune system. In *Proceedings of the 7th International Conference on Artificial Immune Systems*, ICARIS '08, pages 119–131.

[70] Gadi, M. F. A., Wang, X., and do Lago, A. P. (2008b). Credit card fraud detection with artificial immune system. In Bentley, Peter J.and Lee, D. J. S., editor, *Artificial Immune Systems*, pages 119–131, Berlin, Heidelberg. Springer Berlin Heidelberg.

[71] Gascon, H., Uellenbeck, S., Wolf, C., and Rieck, K. (2014). Continuous authentication on mobile devices by analysis of typing motion behavior.

[72] Gharibi, W. and Mirza, A. (2011). Software vulnerabilities, banking threats, botnets and malware self-protection technologies. *arXiv preprint arXiv:1105.1720*.

[73] Ghosh and Reilly (1994). Credit card fraud detection with a neural-network. *1994 Proc. Twenty-Seventh Hawaii Int. Conf. Syst. Sci.*, 3:621–630.

[74] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. *Nature*, 521(7553):800.

[75] Guenther, N. and Schonlau, M. (2016). Support vector machines. *Stata J.*, 16(4):917–937.

[76] Haber, M. (2016). Password Reuse – Overcome the Vulnerability. *Beyondtrust.com*.

[77] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2:1735–1742.

[78] Hastie, T., Tibshirani, R. J., and Friedman, J. (2009). The Elements of Statistical Learning. *Math. Intell.*, 27(2):745.

[79] Hejazi, M. and Singh, Y. (2013). One-class support vector machines approach to anomaly detection. *Applied Artificial Intelligence*, 27:351–366.

[80] Hope, D. (2016). Be warned: Scammers are making fake versions of your favorite apps to steal your data. In *Readers Digest*.

[81] ISO (2015). ISO/IEC 7812-1:2015 Identification cards — Identification of issuers — Part 1: Numbering system. Technical report.

[82] Iyer, D., Mohanpurkar, A., Janardhan, S., Rathod, D., and Sardeshmukh, A. (2011). Credit card fraud detection using Hidden Markov Model. In *2011 World Congr. Inf. Commun. Technol.* IEEE.

[83] Jakobsson, M., Shi, E., Golle, P., and Chow, R. (2009). Implicit authentication for mobile devices. *Proc. of the 4th USENIX conference on Hot topics in security (HotSec'09)*, page 9.

[84] Jo, Y.-H., Jeon, S.-Y., Im, J.-H., and Lee, M.-K. (2016). Security analysis and improvement of fingerprint authentication for smartphones. *Mobile Information Systems*, 2016:1–11.

[85] Johnson, T. (2018). Mcommerce Statistics and Trends in 2019. In *cpcstrategy.com*.

[86] Jr, C. F. (2018). Phishing Threats Move to Mobile Devices. *darkreading.com*.

[87] Jurgovsky, J. et al. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100:234 – 245.

[88] Karlik, B. and Olgac, A. (2010). Performance analysis of various activation functions in generalized MLP architectures of neural networks. *Int. Journal of Artificial Intell. and Expert Systems (IJAE)*, 1(4):111–122.

[89] Karpersky Daily (2015). Bank cards: hidden riks.

[90] Koschuch, M., Hudler, M., Eigner, H., and Saffer, Z. (2013). Token-based authentication for smartphones. In *2013 International Conference on Data Communication Networking (DCNET)*, pages 1–6.

[91] Lee, H., Lee, S., Kim, T., and Bahn, H. (2008). Secure user identification for consumer electronics devices. *IEEE Transactions on Consumer Electronics*, 54(4):1798–1802.

[92] Lin, C. C., Chang, C. C., and Liang, D. (2013). A novel non-intrusive user authentication method based on touchscreen of smartphones. *Proc. - 2013 Int. Symp. Biometrics Secur. Technol. ISBAST 2013*, pages 212–216.

[93] Linck, K., Pousttchi, K., and Wiedemann, D. G. (2006). Security Issues in Mobile Payment from the Customer Viewpoint. *ECIS 2006 proc.*, (2923):1–12.

[94] Lookout (2018). Mobile phishing 2018: Myths and facts facing every modern enterprise today.

[95] Louppe, G. (2014). *Understanding Random Forests: From Theory to Practice*. PhD thesis.

[96] Lower, R. (2014). The future of m-commerce. Technical report, Barclays.

[97] Lu, L. and Liu, Y. (2015). Safeguard: User reauthentication on smartphones via behavioral biometrics. *IEEE Transactions on Computational Social Systems*, 2(3):53–64.

[98] Lu, Y., Kuo, C., Chen, H., Chen, C., and Chou, S. (2018). A svm-based malware detection mechanism for android devices. In *2018 International Conference on System Science and Engineering (ICSSE)*, pages 1–6.

[99] Lunt, G. O. B. (2001). Journal et. al. - 2001 - Prospecting for gems in credit card data *.pdf. (October 2000):173–200.

[100] Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (1993). Credit card fraud detection using bayesian and neural networks. In *In: Maciunas RJ, editor. Interactive image-guided neurosurgery. American Association Neurological Surgeons*, pages 261–270.

[101] Martin, R. R. and Christoph, S. (2015). Fusion of Face and Periocular Information for Improved Authentication on Smartphones. pages 2115–2120.

[102] Mehrnezhad, M., Ali, M. A., Hao, F., and van Moorsel, A. (2016). Nfc payment spy: A privacy attack on contactless payments. In Chen, L., McGrew, D., and Mitchell, C., editors, *Security Standardisation Research*, pages 92–111, Cham. Springer International Publishing.

[103] Mueller, J. (2016). Siamese Recurrent Architectures for Learning Sentence Similarity. *Proc. 30th Conf. Artif. Intell. (AAAI 2016)*, (2012):2786–2792.

[104] Neverova, N., Wolf, C., Lacey, G., Fridman, L., Chandra, D., Barbello, B., and Taylor, G. (2016). Learning Human Identity from Motion Patterns. *IEEE Access*, 4:1810–1820.

[105] Ng, A. (2000). 3 Margins : Intuition. *Intell. Syst. their Appl. IEEE*, pt.1(x):1–25.

[106] Nickel, C., Brandt, H., and Busch, C. (2011). Benchmarking the performance of svms and hmms for accelerometer-based biometric gait recognition. In *2011 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 281–286.

[107] Nickel, C., Wirtl, T., and Busch, C. (2012a). Authentication of smartphone users based on the way they walk using k-nn algorithm. In *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 16–20.

[108] Nickel, C., Wirtl, T., and Busch, C. (2012b). Authentication of smartphone users based on the way they walk using k-NN algorithm. *Proc. 2012 8th Int. Conf. Intell. Inf. Hiding Multimed. Signal Process. IIH-MSP 2012*, pages 16–20.

[109] Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 271–279. Curran Associates, Inc.

[110] Olgac, A. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence And Expert Systems*, 1:111–122.

[111] O'Mahony, D., Peirce, M., and Tewari, H. (2001). *Electronic Payment Systems for E-commerce*. Artech House Universal Personal Communications Series. Artech House.

[112] Orr, R. J. and Abowd, G. D. (2000). The smart floor: A mechanism for natural user identification and tracking. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, pages 275–276, New York, NY, USA. ACM.

[113] Ozçelik, M. H., Duman, E., Isik, M., and Cevik, T. (2010). Improving a credit card fraud detection system using genetic algorithm. *2010 Int. Conf. Netw. Inf. Technol.*, pages 436–440.

[114] Pagliery, J. (2015). Hackers are draining bank accounts via the Starbucks app. In *CNN tech*.

[115] Parreño-Cenetno, M., Castruccio, S., and van Moorsel, A. (2017). Smartphone Continuous Authentication Using Deep Learning Autoencoders. *Privacy, Security and Trust 2017*.

[116] Parreño-Centeno, M., Guan, Y., and van Moorsel, A. (2018). Mobile Based Continuous Authentication Using Deep Features. *Procedings of 2nd Int. Work. Embed. Mob. Deep Learn.*, page 6.

[117] Patel, K., Han, H., and Jain, A. K. (2015). Secure Smartphone Unlock: Robust Face Spoof Detection on Mobile. 11(10):2268–2283.

[118] Patel, K., Han, H., and K. Jain, A. (2016). Secure face unlock: Spoof detection on smartphones. 11.

[119] Pereira, T. D. F., McCool, C., Marcel, S., Hadid, A., Pietikinen, M., Matjka, P., Rernock, J., Poh, N., Kittler, J., Larcher, A., Levy, C., Matrouf, D., Bonastre, J. F., Tresadern, P., Cootes, T., Oishi, S., Ichino, M., Yoshiura, H., Raja, K. B., Raghavendra, R., Stokkenes, M., and Busch, C. (2014). Smartphone Authentication System Using Periocular Biometrics. *Proc. 2012 IEEE Int. Conf. Multimed. Expo Work. ICMEW 2012*, 1(8):3–5.

[120] Platt, J. (2000). *Probabilities for Support Vector Machines*.

[121] Pozzolo, A. D., Caelen, O., Johnson, R. A., and Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. *Proc. - 2015 IEEE Symp. Ser. Comput. Intell. SSCI 2015*, pages 159–166.

[122] PYMNTS (2015). MOBILE COMMERCEAs Mobile Commerce Grows, M-Commerce Fraud Grows Even Faster.

[123] Qian, G., Sural, S., Gu, Y., and Pramanik, S. (2004). Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, page 1232–1237, New York, NY, USA. Association for Computing Machinery.

[124] Quah, J. T. et al. (2008). Real-time credit card fraud detection using computational intelligence. *Expert Systems with Applications*, 35(4):1721 – 1732.

[125] Ragan, S. (2017). Equifax says website vulnerability exposed 143 million US consumers. *csonline.com*.

[126] Raghavendra, R. and Busch, C. (2016). Learning deeply coupled autoencoders for smartphone based robust periocular verification. *Image Process. (ICIP), 2016 IEEE Int. Conf.*

[127] Raja, K. B., Raghavendra, R., and Busch, C. (2014). Smartphone based robust iris recognition in visible spectrum using clustered K-means features. *BIOMS 2014 - 2014 IEEE Work. Biometric Meas. Syst. Secur. Med. Appl. Proc.*, pages 15–21.

[128] Raja, K. B., Raghavendra, R., Stokkenes, M., and Busch, C. (2015). Multi-modal authentication system for smartphones using face, iris and periocular. *Proc. 2015 Int. Conf. Biometrics, ICB 2015*, pages 143–150.

[129] Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, pages 512–519.

[130] Ross, A. and Jain, A. K. (2004). Multimodal biometrics: An overview.

[131] Roy, A., Halevi, T., and Memon, N. (2015). An hmm-based multi-sensor approach for continuous mobile authentication. In *MILCOM 2015 - 2015 IEEE Military Communications Conference*, pages 1311–1316.

[132] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[133] Sahin, Y. and Duman, E. (2011a). Detecting credit card fraud by decision trees and support vector machines. *IMECS 2011 - International MultiConference of Engineers and Computer Scientists 2011*, 1:442–447.

[134] Sahin, Y. and Duman, E. (2011b). Detecting credit card fraud by decision trees and support vector machines. 1:442–447.

[135] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks.

[136] Shabrina, N., Isshiki, T., and Kunieda, H. (2016). Fingerprint authentication on touch sensor using Phase-Only Correlation method. In *2016 7th Int. Conf. Inf. Commun. Technol. Embed. Syst.*, volume 2016, pages 85–89. IEEE.

[137] Shelton, J., Dozier, G., Adams, J., and Alford, A. (2012). Permutation-based biometric authentication protocols for mitigating replay attacks. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–5.

[138] Shen, Y., Hu, W., Yang, M., Wei, B., Lucey, S., and Chou, C. T. (2014). Face recognition on smartphones via optimised Sparse Representation Classification. In *IPSN-14 Proc. 13th Int. Symp. Inf. Process. Sens. Networks*, pages 237–248. IEEE.

[139] Sitova, Z., Sedenka, J., Yang, Q., Peng, G., Zhou, G., Gasti, P., and Balagani, K. S. (2016). HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users. *IEEE Transactions on Information Forensics and Security*, 11(5):877–892.

[140] Smith, A. (2015). U.S. Smartphone Use in 2015. In *Pew Reseacrh Cent.*

[141] Smith, B. T. and Science, A. (2004). Lagrange Multipliers Tutorial in the Context of Support Vector Machines. *Science (80-.).*

[142] Sonawane, K. (2019). Mobile Payment Market Expected to Reach $ 4,574 Billion by 2023. In *www.alliedmarketresearch.com*.

[143] Srivastava, A., Kundu, A., Sural, S., and Majumdar, A. (2008). Credit card fraud detection using hidden markov model. In *IEEE Transactions on Dependable and Secure Computing*, volume 5, pages 37–38.

[144] Statista (2018). Number of smartphone users worldwide from 2014 to 2020.

[145] Sterling, G. (2016). Majority of emails opened on apple devices, android users pay more attention. In *Marketing Land*.

[146] Stolfo, S., Fan, W., Lee, W., Prodromidis, A., and Chan, P. (2000). Cost-based modeling for fraud and intrusion detection: results from the jam project, in: DARPA Information Survivability Conference and Exposition. *DISCEX'00, Proceedings*, 2(IEEE, 2000):130–144.

[147] Sun, J., Yan, K., Liu, X., Yang, C., and Fu, Y. (2017). Malware detection on android smartphones using keywords vector and svm. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 833–838.

[148] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1701–1708.

[149] Tajalizadehkhoob, S., Asghari, H., Gañán, C., and Van Eeten, M. (2014). Why them? extracting intelligence about target selection from zeus financial malware. In *Proceedings of the 13th Annual Workshop on the Economics of Information Security, WEIS 2014, State College (USA), June 23-24, 2014*. WEIS.

[150] Tech, V. M. (2014). Fraud Detection in Credit Card by Clustering Approach. *Int. J. Comput. Appl.*, 98(3):975–8887.

[151] Teresa Bryan, Francine Dubois, Anne Hagen, Mary Hughes, Keith Koval, Janet LaFrence, Lisa Weimar, A. Z. (2017). Card-Not-Present Fraud around the World. *U.S. Payments Forum*, 1(March).

[The UK association] The UK association. Contactless Statistics.

[The UK Cards Association] The UK Cards Association. Card present transactions.

[154] Trokielewicz, M. (2016). Iris recognition with a database of iris images obtained in visible light using smartphone camera. *ISBA 2016 - IEEE Int. Conf. Identity, Secur. Behav. Anal.*, pages 1–6.

[155] Tsikerdekis, M. and Zeadally, S. (2014). Online deception in social media. *Communications of the ACM*, 57(9):72–80.

[Ush et al.] Ush, A., Khan, S., Akhtar, N., and Qureshi, M. N. Real-time credit-card fraud detection using artificial neural network tuned by simulated annealing algorithm.

[157] van Gerven, M. and Bohte, S. (2017). Editorial: Artificial neural networks as models of neural information processing. *Frontiers in Computational Neuroscience*, 11:114.

[158] Vatsa, V., Sural, S., and Majumdar, A. K. (2005). A game-theoretic approach to credit card fraud detection. In Jajodia, S. and Mazumdar, C., editors, *Information Systems Security*, pages 263–276, Berlin, Heidelberg. Springer Berlin Heidelberg.

[159] Vignan, D. B. and Technology, I. (2009). Biometric Authentication : A Review Biometric Authentication : A Review. (October 2016):12–28.

[160] Vincent, P. et al. (2008a). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103.

[161] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008b). Extracting and composing robust features with denoising autoencoders.

[162] Wang, D., Li, J., and Memik, G. (2010). User identification based on finger-vein patterns for consumer electronics devices. *IEEE Transactions on Consumer Electronics*, 56(2):799–804.

[163] Wang, S. (2010). A comprehensive survey of data mining-based accounting-fraud detection research. In *2010 International Conference on Intelligent Computation Technology and Automation*, volume 1, pages 50–53.

[164] Wang, W., Zhu, M., Wang, J., Zeng, X., and Yang, Z. (2017). End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 43–48.

[165] Welke, P., Andone, I., Blaszkiewicz, K., and Markowetz, A. (2016). Differentiating smartphone users by app usage. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, pages 519–523, New York, NY, USA. ACM.

[166] Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., and Adams, N. M. (2009). Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55.

[167] Worldpay (2018). The art and science of global payments A definitive report from Worldpay Global Payments Report. (November).

[168] Xin, K., Li, G., Qin, Z., and Zhang, Q. (2012). Malware detection in smartphone using hidden markov model. In *2012 Fourth International Conference on Multimedia Information Networking and Security*, pages 857–860.

[169] Xu, H., Zhou, Y., and Lyu, M. R. (2014). Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones. In *Proceedings of the Tenth USENIX Conference on Usable Privacy and Security*, SOUPS'14, pages 187–198, Berkeley, CA, USA. USENIX Association.

[170] Xu, Q., Erman, J., Gerber, A., Mao, Z., Pang, J., and Venkataraman, S. (2011). Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference - IMC '11*, page 329, New York, New York, USA. ACM Press.

[171] Yaginuma, Y., Kimoto, T., and Yamakawa, H. (1996). Multi-sensor fusion model for constructing internal representation using autoencoder neural networks. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 3, pages 1646–1651 vol.3.

[172] Yang, Q., Peng, G., Nguyen, D. T., Qi, X., Zhou, G., Sitová, Z., Gasti, P., and Balagani, K. S. (2014). A multimodal data set for evaluating continuous authentication performance in smartphones.

[173] Yang, Y. C. (2010). Web user behavioral profiling for user identification. *Decision Support Systems*, 49(3):261 – 271.

[174] Yip, M., Shadbolt, N., and Webber, C. (2012). Structural analysis of online criminal social networks.

[175] Yip, M., Webber, C., and Shadbolt, N. (2013). Trust among cybercriminals? carding forums, uncertainty and implications for policing. *Policing and Society*, 23(4):516–539.

[176] Yiyan, L., Fang, Z., Wenhua, S., and Haiyong, L. (2016). An hidden markov model based complex walking pattern recognition algorithm. In *2016 Fourth International Conference on Ubiquitous Positioning, Indoor Navigation and Location Based Services (UPINLBS)*, pages 223–229.

[177] Young, J. (2018). The rise of M-commerce in The United Kingdom. *The London Economic*.

[178] Zirjawi, N., Kurtanović, Z., and Maalej, W. (2015). A survey about user requirements for biometric authentication on smartphones. *2nd Int. Work. Evol. Secur. Priv. Requir. Eng. ESPRE 2015 - Proc.*, pages 1–6.

[179] Zviran, M. and Haga, W. J. (1993). A Comparison of Password Techniques for Multilevel Authentication Mechanisms. *The Computer Journal*, 36(3):227–237.