Data Mining, Fraud Detection and Mobile Telecommunications: Call Pattern Analysis with Unsupervised Neural Networks

by

Olusola Adeniyi Abidogun

A Thesis Presented

In Fulfillment of the

Requirements for the Degree of

Master of Science

in

Computer Science

University of the Western Cape

Supervisor: Professor Christian W. Omlin

August, 2005

Data Mining, Fraud Detection and Mobile Telecommunications: Call Pattern Analysis with Unsupervised Neural Networks

OLUSOLA ADENIYI ABIDOGUN

Keywords

Data Mining, Knowledge Discovery, Call Profiling, Mobile Telecommunications, Subscription Fraud, Fraud Detection, Self-Organizing Maps, Long Short-Term Memory Recurrent Neural Networks, Unsupervised Training, Spatiotemporal Pattern Modelling.

Abstract

Huge amounts of data are being collected as a result of the increased use of mobile telecommunications. Insight into information and knowledge derived from these databases can give operators a competitive edge in terms of customer care and retention, marketing and fraud detection.

One of the strategies for fraud detection checks for signs of questionable changes in user behavior. Although the intentions of the mobile phone users cannot be observed, their intentions are reflected in the call data which define usage patterns. Over a period of time, an individual phone generates a large pattern of use. While call data are recorded for subscribers for billing purposes, we are making no prior assumptions about the data indicative of fraudulent call patterns, i.e. the calls made for billing purpose are unlabeled. Further analysis is thus, required to be able to isolate fraudulent usage. An unsupervised learning algorithm can analyse and cluster call patterns for each subscriber in order to facilitate the fraud detection process.

This research investigates the unsupervised learning potentials of two neural networks for the profiling of calls made by users over a period of time in a mobile telecommunication network. Our study provides a comparative analysis and application of Self-Organizing Maps (SOM) and Long Short-Term Memory (LSTM) recurrent neural networks algorithms to user call data records in order to conduct a descriptive data mining on users call patterns.

Our investigation shows the learning ability of both techniques to discriminate user

call patterns; the LSTM recurrent neural network algorithm providing a better discrimination than the SOM algorithm in terms of long time series modelling. LSTM discriminates different types of temporal sequences and groups them according to a variety of features. The ordered features can later be interpreted and labeled according to specific requirements of the mobile service provider. Thus, suspicious call behaviours are isolated within the mobile telecommunication network and can be used to to identify fraudulent call patterns. We give results using masked call data from a real mobile telecommunication network.

Declaration

I, the undersigned hereby declare that *Data Mining, Fraud Detection and Mobile Telecommunications: Call Pattern Analysis with Unsupervised Neural Networks* is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

OLUSOLA ADENIYI ABIDOGUN

August 2005

Dedication

To the One who is before all things and in Him all things hold together and to my late father Mr. E. A. Abidogun and my loving mother Mrs. J. O. Abidogun.

Acknowledgments

My profound gratitude goes to my supervisor Professor Christian W. Omlin for his valuable contribution and expertise. I appreciate the freedom he gave to me to follow my own ideas and for his being an excellent reference point and critic to test my ideas. His patience, motivation and guidance contributed significantly to the completion of this dissertation.

Further, I would like to express my appreciation to Dr. (Mrs.) Magdalena Klapper-Rybicka, formally of IDSIA (Istituto Dalle Molle di Studi sull'Intelligenza Artificiale) – Dalle Molle Institute for Artificial Intelligence, for her assistance with the Unsupervised LSTM RNN implementation. My sincere gratitude to Peter Kleiweg, of the University of Groningen, Netherlands for his assistance with the Extended SOM implementation.

I wish to acknowledge the support of the Telkom-Cisco Centre of Excellence for IP and Internet Computing, and the National Research Foundation for funding this research.

Special thanks to my family who always supported me in everything I wanted to do.

Finally, I am grateful to everybody who helped me to finish this thesis.

Articles

- O.A. Abidogun and C. W. Omlin, Intelligent Fraud Detection - Work in Progress, In *Proceedings of 7th Southern African Telecommunication Networks & Applications Conference* (SATNAC 2003), September 7th - 10th, 2003 Southern Cape, South Africa.

- O.A. Abidogun and C. W. Omlin, Intelligent Fraud Detection, In *Proceedings of 1st African Conference on the Digital Common* (Idlelo 2004), January 13th - 16th 2004, University of the Western Cape, Bellville, Cape Town, South Africa.

- O.A. Abidogun and C. W. Omlin, Call Profiling Using Self–Organizing Maps, In *Proceedings of the 4th International Conference on Computational Intelligence for Modelling, Control and Automation* (CIMCA 2004), July 12th - 14th 2004, Gold Coast, Australia.

- O.A. Abidogun and C. W. Omlin, A Self–Organizing Maps Model for Outlier Detection in Call Data from Mobile Telecommunication Networks, In *Proceedings of 8th Southern African Telecommunication Networks & Applications Conference* (SATNAC 2004), September 6th - 8th 2004, Spier Wine Estate, South Western Cape, South Africa.

- O.A. Abidogun and C.W. Omlin, Fraud Detection in Mobile Telecommunication Networks: Call Profiling with Unsupervised Neural Networks, In *Proceedings of 12th International Conference on Telecommunications* (ICT 2005), May 3 - 6 2005, Cape Town, South Africa.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Motivation 2

1.3 Premises 3

1.4 Problem Statement 4

1.5 Research Hypotheses 5

1.6 Technical Objectives 6

1.7 Methodology 6

1.8 Accomplishments 6

1.9 Thesis Outline 7

2 Fraud Detection **8**

2.1 Introduction 8

2.2 Definition of Fraud 8

2.3 Development of Fraud 9

2.4 Previous Work 10

2.5 Summary 16

3 Time Series Modelling with Artificial Neural Networks **17**

3.1 Introduction 17

3.2 Neural Networks Overview 18

3.3 Feedforward Neural Networks 19

3.4 Time Delay Neural Networks 21

3.5 Finite Impulse Response and Infinite Impulse Response Neural Networks . . . 23
3.6 Bi-Directional Neural Networks . . . 27
3.7 Recurrent Neural Networks . . . 30
3.7.1 Back-Propagation Through Time . . . 32
3.7.2 Real-Time Recurrent Learning . . . 33
3.8 The Vanishing Gradient Problem . . . 35
3.8.1 Time Constants . . . 36
3.8.2 Ring's Approach . . . 37
3.8.3 Searching Without Gradients . . . 37
3.8.4 Long-Short Term Memory . . . 37
3.9 Summary . . . 37

4 Long Short-Term Memory Recurrent Neural Networks 39
4.1 Introduction . . . 39
4.2 Traditional LSTM RNNs . . . 40
4.2.1 Forward Propagation . . . 42
4.3 LSTM RNNs With Forget Gates . . . 43
4.4 LSTM RNNs With Peephole Connections . . . 45
4.5 Learning In LSTM RNNs . . . 46
4.6 Information Theoretic Models . . . 49
4.6.1 Binary Information Gain Optimization (BINGO) . . . 50
4.6.2 Nonparametric Entropy Optimisation (NEO) . . . 50
4.7 Applications of LSTM RNNs . . . 51
4.7.1 Blues Improvisation . . . 51
4.7.2 Automatic Speech Recognition . . . 52
4.7.3 Named Entity Recognition . . . 52
4.8 Summary . . . 53

5 Call Pattern Analysis in Mobile Telecommunication Networks 54
5.1 Introduction . . . 54
5.2 Data Mining . . . 54

5.3 Self-Organizing Maps 55
5.4 Experiments 57
5.4.1 Experiment 1: Self Organizing Map Model 57
5.4.2 Experiment 2: Unsupervised Training of LSTM Recurrent Neural Networks 62
5.5 Assessment of Models 69
5.6 Summary 70

6 Conclusions and Directions for Future Research 71
6.1 Conclusion 71
6.2 Directions for Future Research 72
6.2.1 LSTM RNN Trained With Other Objective Functions 72
6.2.2 LSTM RNN for Predictive Data Mining Tasks 72

Bibliography 74

List of Figures

3.1 **A Multi-layer Feedforward Neural Network with l layers of units:** The input layer N_i is distinguished by its lack of incoming connections. Similarly, the ouput layer N_o is deprived of outgoing connections. The remaining layers possess both types of connectivity and are referred to as hidden layers. 20

3.2 **A Time Delay Neural Network (TDNN) Representation:** TDNN is a multilayer feedforward network whose hidden and output neurons are replicated across time 22

3.3 **A FIR Digital Filter:** The filter contains weighted tapped delay lines and do not have recurrent connections. The unit delay operator z^{-1} represents the input at a given time step 24

3.4 **An IIR Digital Filter:** The filter contains weighted tapped delay lines and do have connections that are locally recurrent. The unit delay operator z^{-1} represents the input at a given time step 25

3.5 **A Neuron Structure with both FIR and IIR Filters** 25

3.6 **Outline of the signal flow in the bi-directional computing architecture for time series prediction** 27

3.7 **A Bi-directional Neural Network Model:** The model consists of two mutually connected sub networks performing direct and inverse signal transformations bi-directionally 28

3.8 **Elman RNN, Jordan RNN and a Fully RNN** 30

4.1 **A LSTM RNN:** The network is similar in structure to a fully RNN except that the hidden layer is replaced by a memory block 41

4.2 **A Memory Block with One Cell** 41
4.3 **A Memory Block with Forget Gates** 44
4.4 **A Memory Block with Peephole Connections** 45

5.1 **An extended Kohonen Map showing difference and resemblance between map units** 60
5.2 **Extended Kohonen Maps for 3 runs on the same input vector of size 9 showing the preservation of the topological property of the maps** 61
5.3 **Trained network output for a subscriber with 9 call transactions** . 64
5.4 **Trained network output for a subscriber with 13 call transactions** 65
5.5 **Trained network output for a subscriber with 22 call transactions** 66
5.6 **Trained network output for a subscriber with 30 call transactions** 66

List of Tables

5.1 **A set of 9 feature vectors for a user transactions in 6 months**: The calls made are labeled C1-C9 for traceability. 59

5.2 **Trained Weight Vectors for Input features in Table 5.1** 59

5.3 **A set of 9 feature vectors for a user call transactions in 6 months with discrete time steps to reflect the temporal order of the sequence**: The calls made are labeled C1-C9 for traceability. 62

5.4 **Trained network output for a subscriber with 9 call transactions** . 63

5.5 **Trained network output for a subscriber with 13 call transactions** 63

5.6 **Trained network output for a subscriber with 22 call transactions** 64

5.7 **Trained network output for a subscriber with 30 call transactions** 65

5.8 **Clusters Identified by the LSTM network for Figure 5.3** 67

5.9 **Clusters Identified by the LSTM network for Figure 5.4** 67

5.10 **Clusters Identified by the LSTM network for Figure 5.5** 67

5.11 **Clusters Identified by the LSTM network for Figure 5.6** 68

Chapter 1

Introduction

This chapter gives a general overview of the thesis. It starts with a background review of the thesis. It describes the motivation for the research and the premises on which the research work is based. The chapter further describes the problem statement and the research hypotheses that were tested in our experiments. Technical objectives are highlighted and the methodology of the dissertation are discussed. Finally, we summarize the results of this thesis and close with an outline of subsequent chapters.

1.1 Background

Telecommunication fraud occurs whenever a perpetrator uses deception to receive telephony services free of charge or at a reduced rate [11]. It is a worldwide problem with substantial annual revenue losses for many companies. Globally, telecommunications fraud is estimated at about 55 billion US dollars [2]. In the United States of America, telecommunication fraud is generally considered to deprive network operators of approximately 2 percent of their revenue. However, as noted by [9], it is difficult to provide precise estimates since some fraud may never be detected, and the operators are reluctant to reveal figures on fraud losses. The situation can significantly be worse for mobile operators in Africa for, as a result of fraud, they become liable for large hard currency payments to foreign network operators. Thus, telecommunication fraud is a significant problem which needs to be addressed, detected and prevented in the strongest possible manner. Popular examples of fraud

in the telecommunication industry include subscription fraud, identity theft, voice over the internet protocol (VoIP) fraud, cellular cloning, billing and payment fraud on telecom accounts, prepay and postpaid frauds and PBX fraud.

1.2 Motivation

Huge amounts of data are being collected and warehoused as a result of increased use of mobile communication services. Insight information and knowledge derived from these databases can give operators a competitive edge in terms of customer care and retention, marketing and fraud detection. Thus, telecommunication fraud has become a high priority item on the agenda of most telecommunication operators.

Fraud is a significant source of lost revenue to the telecommunications industry; furthermore, it lowers customers' confidence in the security of transactions available via the service operator. Since operators are facing increasing competition and losses have been on the rise [75], fraud has gone from being a problem – which carriers were willing to tolerate – to being one that dominates the front pages of both the trade and general press [84].

Efficient fraud detection and analysis systems can save telecommunication operators a lot of money and also help restore subscribers' confidence in the security of their transactions. Automated fraud detection systems enable operators to respond to fraud by detection, service denial and prosecutions against fraudulent users. The huge volume of call activity in a network means that fraud detection and analysis is a challenging problem.

In general, the more advanced a service, the more it is vulnerable to fraud. In the future, operators will need to adapt rapidly to keep pace with new challenges posed by fraudulent users. In addition, the number of actors involved in the provision of a service is likely to increase, making the possibilities for fraud to expand beyond the simple case of subscribers trying to defraud an operator. While conventional

approaches to fraud detection and analysis such as rule-based systems based on thresholds for particular parameters may be sufficient to cope with some current types of fraud, they are less able to cope with the myriad of new possibilities. In addition, fraudsters can change their tactics fairly easily to avoid detection; for instance, systems based on thresholds can be fooled by keeping the call duration below that of the detection threshold.

Therefore, there is need for the consideration of dynamic and adaptive fraud detection and analysis approaches; artificial intelligence techniques offer the promise to effectively address some of these challenges.

1.3 Premises

A Call Detail Record (CDR) is created for every completed call in a mobile telecommunication network. These data records are referred to as Toll Tickets (TT). Toll Tickets contain a wealth of information about the call made by a subscriber. Besides their billing role, Toll Tickets constitute an enormous database within which other useful knowledge about callers can be extracted. One example is the detection of anomalous usage of the mobile telecommunication network. Although various other CDR-like services exist on which anomalous usage can be detected, this study is based on Global System for Mobile communications CDR.

Specifically, this study is based on call detail record for prepaid service subscribers from a real mobile telecommunication network. The data set contains 500 *masked* subscribers, each with calling data for a period of 6 months. We based our investigation on Mobile Originating Calls (MOC) extracted from the data set. These are calls that were initiated by the subscribers within the 6 months period. Fraud related to prepaid service fall under subscription fraud which is the topic of this thesis.

1.4 Problem Statement

Over a period of time, an individual handset's Subscriber Identity Module (SIM) card generates a large pattern of use. The pattern of use may include international calls and time-varying call patterns among others. Anomalous use can be detected within the overall pattern such as subscribers abuse of free call services such as emergency services.

Anomalous use can be identified as belonging to one of two types [36]:

1. The pattern is intrinsically fraudulent; it will almost never occur in normal use. This type is relatively easy to detect.
2. The pattern is anomalous only relative to the historical pattern established for that phone.

In order to detect fraud of the second type, it is necessary to have knowledge of the history of SIM usage. Hence, a descriptive analysis of the call profiling for each subscriber can be used for knowledge extraction. Interpretation by way of clustering or grouping of similar patterns can help in isolating suspicious call behaviour within the mobile telecommunication network. This can also help fraud analysts in their further investigation and call pattern analysis of subscribers. While call data are recorded for subscribers for billing purposes, it is interesting to know that no prior assumptions are made about the data indicative of fraudulent call patterns. In other words, the calls made for billing purposes are unlabeled. Further analysis is thus required to be able to identify possible fraudulent usage. Because of the huge call volumes, it is virtually impossible to analyse without sophisticated techniques and tools. So there is need for techniques and tools to intelligently assist humans in analysing large volumes of calls. One such technique is unsupervised learning. Consequently, this thesis investigates the following open problem.

The unsupervised learning potentials of two neural networks for the profiling of calls made by users over a period of time in a mobile telecommunication network. Specifically, our study provides a comparative analyses and application of Self–Organizing Maps (SOM - a feedforward neural network) and Long Short-Term Memory (LSTM) recurrent neural networks algorithms to user call data records in order to conduct a descriptive data mining on users call patterns.

1.5 Research Hypotheses

The large volume of calls by an individual handset's SIM over a period of time can be regarded as a dynamic time-varying process and this can be captured and represented as a time series.

Unsupervised detection of input regularities is an important application of feedforward neural networks (FNNs) [53]. Typical real-world inputs, however, are not static but temporal sequences with embedded statistical regularities and redundancies [53]. FNNs, therefore, necessarily ignore a large potential for compactly encoding data [4, 6]. While much work has been done on unsupervised learning in feedforward neural networks architectures, its potential with theoretically more powerful recurrent networks and time-varying inputs has rarely been explored [53].

Consequently, the hypothesis of this research is that LSTM Recurrent Neural Networks can be used to discriminate user call patterns in an unsupervised learning approach. The unsupervised LSTM RNN can provide a better discrimination than the SOM (a FFN architecture) in terms of long time series modelling of the user call data. LSTM discriminates different types of temporal sequences and group them according to a variety of features. The ordered features can later be interpreted and labeled according to specific requirements of the mobile service provider. Thus, suspicious call behaviours can be isolated within the mobile telecommunication network, in order to detect fraudulent call patterns.

1.6 Technical Objectives

The primary aim of this thesis is to investigate the unsupervised learning potentials of two novel neural networks for the profiling of calls made by users over a period of time in a mobile telecommunication network. Specifically, this study provides a comparative analyses and application of Self Organizing Maps (SOM) and Long Short-Term Memory (LSTM) recurrent neural networks algorithms to user call data records in order to conduct a descriptive data mining on users call patterns.

In addition, from this investigation we aim to highlight the salient call features from the call detail records in order to identify anomalous usage of the mobile phone network services. Finally, we hope to establish that LSTM algorithm provides a better discrimination than the SOM algorithm in terms of the unsupervised long time series modelling. LSTM discriminate different types of temporal sequences and group them according to a variety of features.

1.7 Methodology

Our method of research proceeds with the normalization of our call data records which contained a 6-month call data set of 500 *masked* subscribers from a real mobile telecommunication network. We extracted from the data set, Mobile Originating Calls (MOC). These are calls that were initiated by the subscribers. Within the 6 months period, a total of 227,318 calls originated from the 500 subscribers. The SOM and LSTM RNN models are then applied to unsupervised discrimination of the normalized call data set. Results are reported which estimates the performances of the two learning models.

1.8 Accomplishments

We applied SOM and LSTM RNNs to the problem of fraud detection in mobile telecommunication networks in an unsupervised learning approach. To our knowledge, these models have not been applied to unlabeled call data record before. We

give readers insight into various time series modelling approach and supply sufficient motivation for using LSTM RNNs as a preferred solution to our problem statement. Our results show the feasibility of applying LSTM RNNs to unsupervised discrimination of unlabeled call data records in a mobile telecommunication network for the purpose of call pattern analysis.

1.9 Thesis Outline

The rest of this thesis is organized as follows: In Chapter 2 we introduce the problem of fraud detection and proceed to a review of previous work. In Chapter 3, we present an overview of the various artificial neural network time series modelling techniques as well as some application domains; we conclude with problems pertaining to modelling long time series. Chapter 4, focuses on the theory of long short-term memory recurrent neural networks and discuss a few of its application. We then present our application of long short-term memory recurrent neural networks to fraud detection in mobile telecommunication networks in Chapter 5, as well as our Self Organizing Maps model application. In Chapter 6, we discussed the results of our experiments and conclude with possible direction for future research.

Chapter 2

Fraud Detection

2.1 Introduction

This chapter introduces the problem of fraud detection, starting from definitions. A historical development of fraud is also discussed with a review of previous work.

2.2 Definition of Fraud

In many of the existing literature, the intention of the subscriber plays a central role in the definition of fraud. [51] defines fraud as any transmission of voice or data across a telecommunications network where the intent of the sender is to avoid or reduce legitimate call charges. Likewise, [25] defines fraud as obtaining unbillable services and undeserved fees. According to [51], the serious fraudster sees himself as an entrepreneur, admittedly utilizing illegal methods, but motivated and directed by essentially the same issues of cost, marketing, pricing, network design and operations as any legitimate network operator. [44] considers fraud as attractive from the fraudsters' point of view, since detection risk is low, no special equipment is needed, and the product in question is easily converted to cash. It is important to state that although the term fraud has a particular meaning in legislation, this established term is used broadly to mean misuse, dishonest intention or improper conduct without implying any legal consequences.

2.3 Development of Fraud

Historically, earlier types of fraud used technological means to acquire free access [48]. Cloning of mobile phones by creating copies of mobile terminals with identification numbers from legitimate subscribers was used as a means of gaining free access [25]. In the era of analog mobile terminals, identification numbers could be easily captured by eavesdropping with suitable receiver equipment in public places, where mobile phones were evidently used. One specific type of fraud, tumbling, was quite prevalent in the United States [25]. It exploited deficiencies in the validation of subscriber identity when a mobile phone subscription was used outside of the subscriber's home area. The fraudster kept tumbling (switching between) captured identification numbers to gain access. [25] state that the tumbling and cloning fraud have been serious threats to operators' revenues. First fraud detection systems examined whether two instances of one subscription were used at the same time (overlapping calls detection mechanism) or at locations far apart in temporal proximity (velocity trap). Both the overlapping calls and the velocity trap try to detect the existence of two mobile phones with identical identification codes, clearly evidencing cloning. As a countermeasure to these fraud types, technological improvements were introduced.

However, new forms of fraud came into existence. A few years later, [84] reports the so-called subscription fraud to be the trendiest and the fastest-growing type of fraud. In similar spirit, [44] characterizes subscription fraud as being probably the most significant and prevalent worldwide telecommunications fraud type. In subscription fraud, a fraudster obtains a subscription (possibly with false identification) and starts a fraudulent activity with no intention to pay the bill. It is indeed non-technical in nature and by call selling, the entrepreneur-minded fraudster can generate significant revenues for a minimal investment in a very short period of time [51]. From the above explanation it is evident that the detection mechanisms of the first generation fraud soon became inadequate. More advanced detection mechanisms must, therefore, be based on the behavioural modeling of calling activity – this is the central subject of

this thesis.

2.4 Previous Work

In this section we review published work with relevance to fraud detection in telecommunications networks.

Telecommunications companies have been studying fraud and fraud detection for many years and have probably spent more time and money on this than the research community [58]. However, most of their efforts do not reach beyond the limits of the companies and have not been available to the public research community. Still, a number of published papers on the subject are available.

[65] describes toll fraud, how it occurs and offers ways to secure systems from hackers. [65] also raises questions about who should be responsible for prevention of toll-fraud - subscribers, long-distance carrier, operators or manufacturer of telecommunication equipment.

[82] discusses various aspects of digital transmission of wireless communication. It describes the vulnerability of wireless communication to a type of wireless fraud known as tumbling. [82] noted that this fraud could easily allow fraudsters to steal telephone services and digital technology. The work also refers to clone fraud and attempts to foil cloners.

[82] describes a solution that creates a profile of normal use for subscribers and then track calling patterns, in terms of frequency, destination, length, origination, parties called, time of day and distance. The authors also describe a solution that can detect the unique signal characteristic of each individual cellular phone and compare it with a database of prints, each of which is assigned to a unique electronic serial number. The system denies call access once the calling telephone does not have the electronic fingerprint it is supposed to. [82] noted that none of these solutions

however is foolproof and that their adoption is slow.

[1] reports on the increasing incidence of phone fraud with corporation and telecommunication firms as victims. It describes the alliance formed to curb phone fraud, preventive measures undertaken by customers and telephone companies to combat fraud and cloning of cellular phone.

[90] comments on the crime of theft and sale of cellular-phone access codes and consequent billing to phone companies and consumers in the United States. It highlights the costs involved in telecommunication fraud and losses incurred. [90] evaluated the extent of the crime and ways in which cellular-phone access codes can be misused. Efforts toward preventive technology were also discussed.

[64] presents an analysis of a report on fraud in the wireless telecommunication industry, estimated losses from fraud; increase in the incidence of subscription fraud and emerging solutions for subscription fraud.

Fraud in telecommunications networks can be characterized by fraud scenarios, which essentially describe how the fraudster gained the illegitimate access to the network [48]. Detection methodologies designed for one specific scenario are likely to miss plenty of the others. For example, velocity trap and overlapping calls detection methodologies are solely aimed at detecting cloned instances of mobile phones and do not catch any of the subscription fraud cases. As stated earlier in Section 2.2, the nature of fraud has changed from cloning fraud to subscription fraud, which makes specialized detection methodologies inadequate. Instead, the focus is on the detection methodologies based on the calling activity which in turn can be roughly divided into two categories [48]. In *absolute analysis*, detection is based on the calling activity models of fraudulent behavior and normal behavior. *Differential analysis* approaches the problem of fraud detection by detecting sudden changes in behavior. Again, it is important to state that the latter approach defines the focus of this thesis. Using differential analysis, methods typically alarm deviations from

the established patterns of usage. When current behavior differs from the established model of behavior, an alarm is raised. In both cases, the analysis methods are usually implemented by using probabilistic models, neural networks or rule-based systems. Henceforth, we shall focus our review on some prominent work which are considered relevant to the work presented in this thesis.

[25] report on the use of a knowledge-based approach to analyze call records delivered from cellular switches in real time. They state that the application of uniform thresholds to all of a carrier's subscribers essentially forces comparison against a mythical average subscriber. Instead, they choose to model each subscriber individually and allow the subscribers' profile to be adaptive in time. In addition, they use knowledge about the general fraudulent behavior, for example, suspicious destination numbers. The analysis component in their system determines if the alarms, taken together, give enough evidence for the case to be reviewed by a human analyst. In their conclusion, the system is credited with the ability to detect fraud quickly – allowing the analysts to focus on the most likely and dangerous fraud cases. In [9], the authors report their first experiments detecting fraud in a database of simulated calls. They use a supervised feedforward neural network to detect anomalous use. Six different user types are simulated stochastically according to the users' calling patterns. Two types of features are derived from this data, one set describing the recent use and the other set describing the longer-term behavior. Both are accumulated statistics of call data over time windows of different lengths. This data is used as input to the neural network. The performance of their classifier is estimated to be 92.5 % on the test data, which has limited value in the light of simulated data and the need to give class-specific estimates on accuracy. This work has also been reported in [35].

[16, 17] focus on unsupervised learning techniques in analyzing user profiles over sequences of call records. They apply their adaptive prototyping methods in creating models of recent and long-term behavior and calculate a distance measure between the two profiles. They discuss on-line estimation techniques as a solution to avoid storing call detail records for calculating statistics over a time period. Their user

profiles are based on the user-specific prototypes, which model the probability distribution of the call starting times and call durations. A large change in user behavior profiles expressed by the Hellinger distance between profiles is reported as an alarm. In [70, 71], work on fraud detection based on supervised feedforward neural network techniques is reported. The authors criticize thresholding techniques that are based on excessive usage detection since these might be the very best customers if they are legitimate users. In order to use supervised learning techniques, they manually label the estimated user profiles of longer term and recent use, similar to those in [17], into fraudulent and non-fraudulent and train their neural networks on these user profiles. In [70], they report having classified test data with detection probabilities in the range of 80 - 90 % and false alarm probabilities in the range of 2 - 5 %. Collaborative efforts of the two previous groups to develop a fraud detection system have been reported in [69, 18]. Interesting in this context is the performance of the combination of the methods. In [49], performance of the combination of the tools is considered. They form an aggregated decision based on individual decisions of the rule-based tool, unsupervised and supervised user profiling tools with the help of logistic regression. They report improved results, particularly in the region of low false positives. In all, their combined tool detects 60 % of the fraudsters with a false alarm rate of 0.5 %. [68] reports on the final stage of their fraud detection system for GSM networks – BRUTUS with rule-based, supervised and unsupervised learning approaches integrated for fraud detection. The performance of the hybrid detection tool was optimized in terms of the number of subscribers raising alarms. Specifically, the report on the performance curves showed trade-off between percentage of correctly identified fraudsters versus the percentage of new subscribers raising alarms.

[33, 34] present rule-based methods for fraud detection. The authors use adaptive rule sets to uncover indicators of fraudulent behavior from a database of cellular calls. These indicators are used to create profiles, which then serve as features to a system that combines evidence from the multiple profilers to generate alarms. They use rule selection to select a set of rules that span larger sets of fraudulent cases.

Furthermore, these rules are used to formulate monitors, which are in turn pruned by a feature selection methodology. The output of these monitors is weighted together by a learning, linear threshold unit. They assess the results with a cost model in which misclassification cost is proportional to time.

Some work in fraud detection is based on detecting changes in geographical spread of call destinations under fraudulent activity. This view is promoted in [104, 85, 22, 24]. In [104], call data were clustered for further visualization. [22] in turn use neural networks in classification and some authors use human pattern recognition capabilities in recognizing fraud [24, 85].

Fraud and uncollectible debt detection with Bayesian networks has been presented in [31, 30, 32]. They perform variable and dependency selection on a Bayesian network. They also state that a Bayesian network that fits the database most accurately may be poor for a specific task such as classification. However, their problem formulation is to predict uncollectible debt, which includes cases where the intention was not fraudulent and which does not call for user profiles.

[63] reports on the generation of high quality test data for evaluation of fraud detection system. [63] stated that data generation process is designed to collect and analyse authentic data in order to find important statistical properties, which could be used to simulate users with a finite state machine, and to simulate the target system using event-driven simulation. [63] noted that by starting out from authentic data, high quality synthetic data can be artificially created, while preserving important statistical properties of the initial data. They conclude that the generation process is especially suitable for adaptive detection schemes where an exact understanding of the available fraud types is missing.

[15] reports on a review of statistics and machine learning tool as effective technologies for fraud detection in telecommunication.

Some papers, e.g. [19, 20, 21, 44], also describe the current fraud situation that telecommunication companies face, and well-known frauds. However, these papers do not discuss any details of the detection process or any organized fraud model. Because fraud happens over time, methods that deal with time series are relevant to this work.

[74] addresses a Support Vector Machine (SVM) based on user profiling method for fraud detection. An SVM ensemble was employed for fraud behaviour learning and alarm fraud decision-making. They report that user profiling can eventually be induced to binary classification and multi-classification problems of support vector machine. From their simulation results, it was established that the ensemble of SVM is a kind of cross-validation optimization of SVM, and it has been proved to have a better fraud pattern detection performance than other machine learning models, such as multi layer perceptron (MLP) and self-organizing maps (SOM), in terms of adaptability and pattern reorganization accuracy.

[48] presented user profiling and classification techniques for fraud detection in mobile communication networks. The author reported on the identification of relevant user groups based on call data: with each user assigned to a relevant group. He used neural networks and probabilistic models in learning user usage patterns from call data. The author attempts to promote the dynamic modelling of behavioural patterns for fraud detection.

Except for [16, 17], many of the literature available on fraud detection in telecommunications networks focus on supervised learning techniques. Likewise the kind of available data exemplified fraudulent and normal behaviour; in other words, the call data used is labeled. In addition, very few work exist on dynamic modelling of behaviour, although many authors state fraud to be a dynamic phenomenon. For example, [34] doubt the usefulness of hidden Markov models in fraud detection as, in this domain, one is concerned with two states of nature and one single transition between them. However [48] used extensively dynamic models in temporal modelling

of behaviour but in relation to labeled call data.

Available literature on fraud detection in telecommunications networks, therefore, to our knowledge, and as earlier noted, does not provide the basis where unsupervised learning in a dynamic model fashion is applied to unlabelled call data. This distinctively differentiates our work from already existing ones and clearly defines also the central focus of this thesis.

2.5 Summary

Fraud detection is defined with a historical development of fraud discussed. Various existing papers on fraud detection in telecommunications network were reviewed. In addition, it was stated that fraud detection is usually approached by absolute or by differential analysis. It was mentioned that this thesis focuses on the differential analysis approach. Furthermore, we highlighted the central theme of our work which focuses on unsupervised learning in a dynamic model fashion applied to unlabelled call data. An approach which to our knowledge has not been addressed before. The dynamic, time series modelling capabilities of artificial neural networks is presented in the next chapter.

Chapter 3

Time Series Modelling with Artificial Neural Networks

3.1 Introduction

Human beings are capable of gathering vast amounts of sensory data from their environment which in turn enables them to formulate logical decisions [87]. This data can be represented as a time series which the brain organizes and performs complex operations on that allow us to predict and classify sequences in nature. Artificial neural networks (ANNs) are simple mathematical models devised in an attempt to emulate some of these human functions. Recurrent neural networks (RNNs) are ideally suited for modelling dynamical systems with hidden states. The outputs of such processes are typically recorded in the form of time series.

Time series modelling is concerned with the analysis of temporal relationships in a sequence of values [92]. The goals of time series modelling include analysis and simulation in order to understand the underlying dynamics of the series, recognition, pattern classification, dynamic process control and prediction of future values in the series. We refer the reader to [88] for a review of introductory concepts of time series.

Various machine learning methods have been applied to time series prediction and classification tasks. For example, electroencephalogram classification [89] and dynamic gesture recognition [14]. Time can be continuous or discrete.

We give a brief introduction to neural networks, followed by various neural network architectures that have been applied to time series modelling tasks. We conclude with an explanation of the long-term dependency problem that is inherent in traditional RNNs.

3.2 Neural Networks Overview

There are two major classes of ANNs, namely feedforward neural networks (FNNs) and recurrent neural networks (RNNs). In feedforward networks, activation is "piped" through the network from input units to output units. They are also referred to as static networks [57]. FNNs contain no explicit feedback connections [77]. Conventional FNNs are able to approximate any finite function as long as there are enough hidden nodes to accomplish this [42]. RNNs however, are dynamical networks with cyclic path of synaptic connections which serves as the memory elements for handling time-dependent problems.

ANNs have the capability to *learn* from their environment, and to *improve* their performance through learning. Learning is achieved by the ANN through an iterative process of adjustments applied to its synaptic weight and bias level.

There are diverse varieties of learning algorithms for the design of ANNs. They differ from each other in the way in which the adjustment to a synaptic weight of a neuron is formulated. Learning algorithms can be described as a prescribed set of well-defined rules for the solution of a learning problem [42]. Basic learning algorithms include error-correction learning, memory-based learning, Hebbian learning, competitive learning, and Boltzmann learning.

Two fundamental learning paradigms exist. Learning paradigms for ANNs are described in terms of the way and manner by which the interconnected neurons relates to its environment. There exist supervised or associative learning and unsupervised or self-organizing learning paradigms. The former requires an input pattern along

with matching output patterns which is given by an external teacher whereas the latter requires input patterns from which it develops its own representation of the input stimuli.

3.3 Feedforward Neural Networks

A feedforward network has a layered structure (see Figure 3.1). Each layer consists of processing units (or neurons). The layers are arranged linearly with weighted connections between each layer. All connections point in the same direction - a previous layer feeds into the current layer and that layer feeds into the next layer. Information flows from the first, or input, layer to the last, or output, layer. The input layer is distinguished by its lack of incoming connections. Similarly, the output layer is deprived of outgoing connections. The remaining layers possess both types of connectivity and are referred to as hidden layers. Hidden layers contain hidden units which are able to extract higher-order statistics [42]. This is particularly valuable when the size of the input layer is large. Common activation functions in the network may be those whose output is a nonlinear differential function, e.g. sigmoid function, of its input and hence, suitable for gradient descent learning. The error backpropagation learning algorithm and the generalized delta rule - both algorithms are examples of error-correction learning algorithms - are common gradient descent approaches that are used to train these static networks. The backpropagation algorithm is defined according to [66] as follows:

Training examples are presented to a neural network in the form $(\overrightarrow{x}, \overrightarrow{t})$ where $\overrightarrow{x}$ is a vector of network input data, and $\overrightarrow{t}$ is the vector of desired network output signals.

- We construct a feedforward neural network with n_{in} inputs, n_{hidden} units, and n_{out} output units.
- We initialise the networks synaptic weights to a small random value
- We repeat the following steps until the termination condition is met:

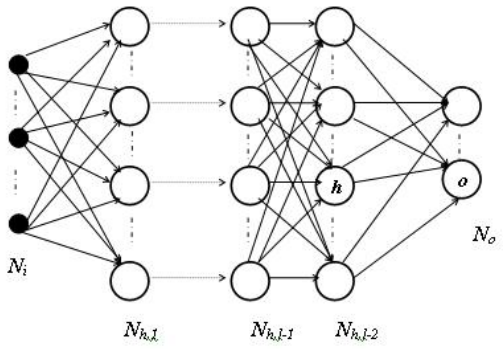

Figure 3.1: **A Multi-layer Feedforward Neural Network with l layers of units:** The input layer N_i is distinguished by its lack of incoming connections. Similarly, the ouput layer N_o is deprived of outgoing connections. The remaining layers possess both types of connectivity and are referred to as hidden layers.

– For each $(\overrightarrow{x}, \overrightarrow{t})$ in the training set, propagate the input forward through the network

– backpropagate the error through the network:

  * For each network output unit k, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

  * For each hidden unit h, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \epsilon outputs} w_{kh}\delta_k$$

  * We then update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

where

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

It is important to note that two basic methods exist for backpropagation learning [42]. These are sequential or on-line or stochastic mode (presented above) and batch

mode backpropagation. The former learning approach updates weights after each training example is presented to the network, while the latter approach requires that the entire training set be presented to the network followed by the weight updates.

Conventional FFNNs only have a very limited ability to deal with time-varying input since they are static learning devices. They can, however, be adapted to deal with temporal relationships as we shall see in the next section.

3.4 Time Delay Neural Networks

Time delay neural network (TDNN) is a popular neural network that uses time delays to perform temporal processing. It was first described in [60] with a a more elaborate study reported by [94]. TDNN is a multilayer feedforward network whose hidden neurons and output neurons are replicated across time (see Figure 3.2). TDNN was specifically developed for speech recognition. The purpose of the architecture is to have a network that contains context which is able to represent sequences. In TDNNs, context or short-term memory is represented as input history. The number of steps that a TDNN is able to process depend entirely on the time window that stores the input sequence. [57] noted that the buffer size limits the length of longest sequence which can successfully be differentiated by these networks. When dealing with time series problems, we must have a good idea of what the size of the longest sequence in the dataset will be. The reason for this is simply that, in TDNNs, a fixed input window size has to be selected based on the longest sequence length. If this prior information is not known a priori, the sequence of course cannot be stored in the time window, thus making processing of the sequence impossible. The number of input to hidden layer weight increases with increasing window size. The increased number of parameters increases the time complexity of the learning algorithm.

The input layer of a TDNN consists of a sliding window whose weight vector is shared amongst other inputs. The output of the activation units is computed by

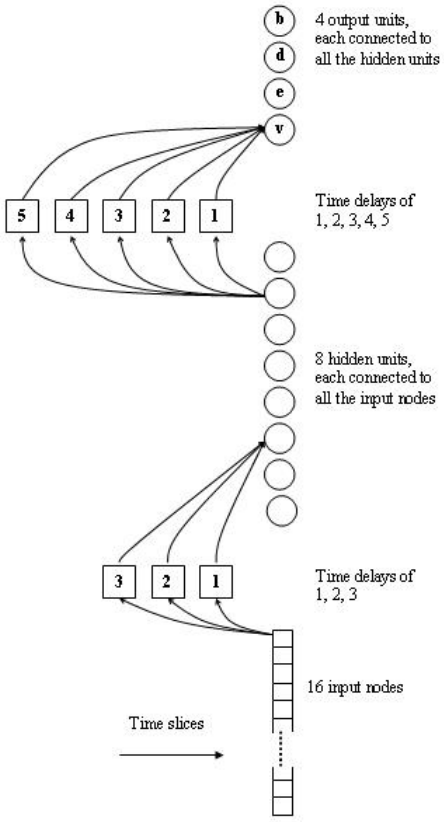

Figure 3.2: **A Time Delay Neural Network (TDNN) Representation:** TDNN is a multilayer feedforward network whose hidden and output neurons are replicated across time

taking a weighted sum of the nodes residing in the input window over a time period and then applying a squashing function to it. TDNNs are trained with the conventional error backpropagation learning algorithm.

[94] used a TDNN with two hidden layers for the recognition of three isolated words: "'bee"', "'dee"', and "'gee"'. In performance evaluation with the use of test data from three speakers, the TDNN achieved an average recognition score of 98.5 percent.

TDNN also classifies spatio-temporal patterns and provides robustness to noise and graceful degradation [62]. However, a limitation of the TDNN as originally posed in [94] is its inability to learn or adapt the values of the time delays. Time delays

are fixed initially and remain the same throughout training. As a result, the system may have poor performance due to the inflexibility of time delays and due to a mismatch between the choice of time delay values and the temporal location of the salient information in the input patterns. In addition, the system performance may vary depending on the range of the time delay values.

To overcome this limitation, [61] proposed a model referred to as Adaptive Time Delay Neural Network (ATDNN). This network adapts its time delay values as well as its weight during training, to better accommodate changing temporal patterns, and to provide more flexibility for optimization tasks. The ATDNN allows arbitrary placement of time delays along interconnections and adapts those time delays independently of one another. Furthermore, time-windows are not used as in [12], [94] but instead classification relies on a set of individual time delay values associated with each interconnection.

3.5 Finite Impulse Response and Infinite Impulse Response Neural Networks

Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) neural networks are temporal neural networks developed by [7] specifically for the task of nonlinear time series prediction. They are based on the traditional FFNN architecture with each regular static synaptic weight replaced by a FIR/IIR linear filter. (See Figures 3.3 and 3.4 for FIR and IIR filters, respectively). FIR networks contain time delays and they do not have recurrent connections. However, the IIR networks have connections that are locally recurrent.

It should be noted that temporal neural network may use FIR filters, IIR filters, or both. This type of network is still globally feedforward in nature, in that it has a global feedforward structure, with possibly local recurrent features (for IIR synapses). Thus, in the FIR filter case, it has local feedforward global feedforward

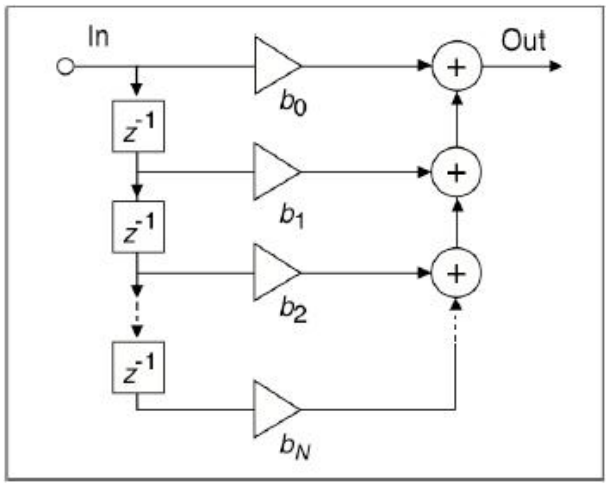

Figure 3.3: **A FIR Digital Filter:** The filter contains weighted tapped delay lines and do not have recurrent connections. The unit delay operator z^{-1} represents the input at a given time step

architecture, while in the IIR synapse case, it has a local recurrrent global feedforward architecture. Figure 3.5 shows a neuron structure with both FIR and IIR filters.

The FIR filter allows an input excitation of finite duration, which results in the output activation of the filter also being of finite duration. It was noted in [100] that FIR networks are functionally equivalent to TDNNs. However, neural networks containing IIR filter, also referred to as recurrent temporal neural networks [29], allow an input presented at any time to continue to influence the output indefinitely. Consequently, training in such network is achieved through feedback of the desired response to the network input after one unit delay in place of the actual output. Training is continued until the network response meets the required tolerance under feedforward conditions with actual output fed back.

A FIR filter produces an output, $y(k)$, which corresponds to the weighted sum of the current and past delayed values of the inputs, $x(k)$.

$$y(k) = \sum_{n=0}^{T} w(n)x(k-n) \tag{3.1}$$

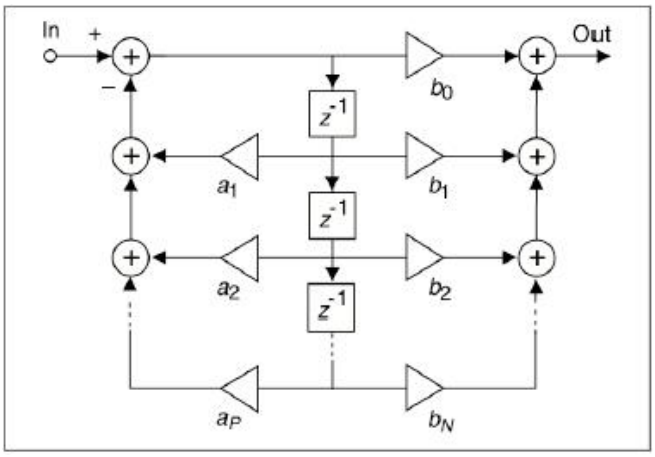

Figure 3.4: **An IIR Digital Filter:** The filter contains weighted tapped delay lines and do have connections that are locally recurrent. The unit delay operator z^{-1} represents the input at a given time step

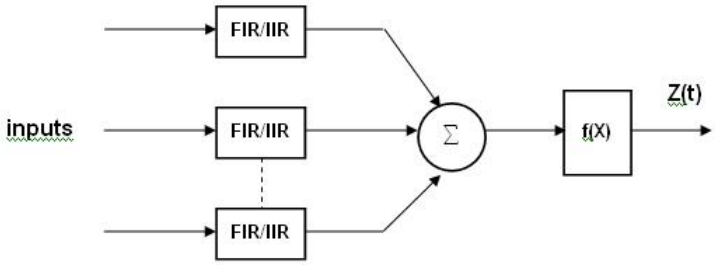

Figure 3.5: **A Neuron Structure with both FIR and IIR Filters**

This is then passed through a sigmoid function which results in the activation of the neuron.

$$y(k) = f(y(k))$$

The IIR filter has the form:

$$y(k) = \sum_{n=0} Ta(n)x(k-n) + \sum_{m=0} Mb(m)y(k-m) \tag{3.2}$$

Figures 3.3 and 3.4 illustrate FIR and IIR filters which contain weighted tapped delay lines. The unit delay operator z^{-1} represents the input at a given time step. The learning algorithm for these networks is known as temporal backpropagation.

(We refer the reader to [7] and [100] for its derivation).

[7] tested the performance of both FIR and IIR network on a time series generated by the following function:

$$y(t) = \sin\{\pi[\frac{\beta_1(q^{-1})}{1 - \alpha_1(q^{-1}) - \alpha_2(q^{-2})}x(t)]\} \tag{3.3}$$

where $x(t)$ is a zero mean white noise source, low-pass filtered with a cut-off frequency of 7 rad/sec, with $\alpha_1 = 0.8227$, $\alpha_2 = -0.9025$, and $\beta_1 = 0.99$. [7] notes that these parameters highlight the dynamics of the system and its nonlinearity. This problem stems from nonlinear control systems which occurs in a wide range of applications used in engineering and science. Some examples include nonlinear circuits, mechanical systems, robotics, chemical processes, flight control, jet engine control, evolutionary systems and biological systems. The time series generated by Equation 3.3 approximates a particular real-world control system.

The FIR and IIR networks were trained by [7] on $5X10^6$ data points generated from Equation 3.3. The test set consisted of unseen generated data of 1000 points. [7] reported testing mean square errors of 0.0664 and $1.2X10^{-5}$ for the FIR and IIR networks respectively.

[7] concludes that the IIR network achieves a lower error rate thus making it a more efficient model than a FIR network for this given task. The reason given is that networks which have local feedback connections, i.e. IIR networks, perform better than those with only local feedforward connections, i.e. FIR networks.

3.6 Bi-Directional Neural Networks

A bi-directional neural network model consists of two mutually connected sub networks performing direct and inverse signal transformations bi-directionally [99]. The model not only deals with the conventional future prediction task, but it also deals with the past prediction, an additional task from the viewpoint of the conventional approach (see Figure 3.6). To apply this model to time series prediction tasks, one sub network is trained with a conventional future prediction task and the other is trained with an additional task for past prediction. Since the coupling effects between the future and past prediction subsystems promote the model's signal processing ability, bi-directionalization of the computing architecture makes it possible to improve its performance [97]. In their work, [97] gave empirical evidence for this claim. Its prediction score is found to be better than with traditional uni-directional method [99].

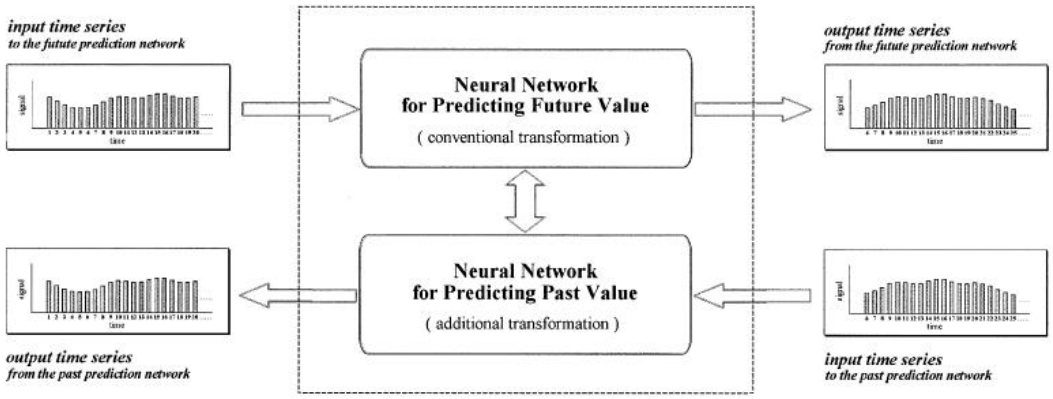

Figure 3.6: **Outline of the signal flow in the bi-directional computing architecture for time series prediction**

A bi-directional computation is able to improve future value prediction by adding values that are predicted from the past [95]. The reason for this is that past values are related to those that occur in the future [101]. The bi-directional neural network thus makes use of past and future values.

Figure 3.7 shows the structure of the bi-directional neural network model. In this figure, as described by [96], the circles represent single neuron layers without internal connections, and the arrows represent weights between adjacent neuron layers. The upper half of the signal processing subsystem is for future prediction, and the lower half of the signal processing subsystem is for past prediction. Each of them consists of a four-layered network. The output signals for each layer in the future prediction system are denoted as $y_i^{[0]}$, $y_i^{[1]}$, $y_i^{[2]}$, $y_i^{[3]}$ ($i = 1, 2, .., n_l$). The state transition rules are defined as follows. Note that n_l ($l = 1, 2, 3, 4$) is the number of neurons in the l-th layer.

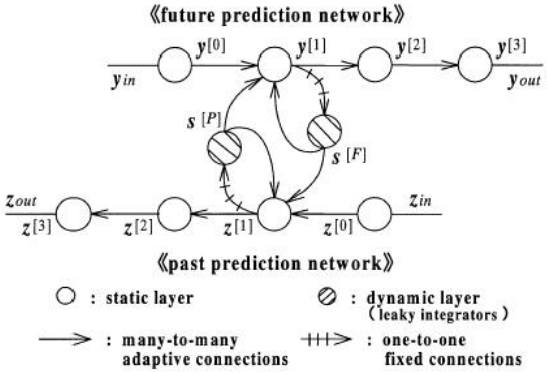

Figure 3.7: **A Bi-directional Neural Network Model:** The model consists of two mutually connected sub networks performing direct and inverse signal transformations bi-directionally

$$y_i^{[0]} = [y_{in}]_i, \tag{3.4}$$

$$y_i^{[1]} = f_1(\sum_j w_{ij}^{[1]} y_j^{[0]} + \sum_j w_{ij}^{[F]} s_j^{[F]} + \sum_j w_{ij}^{[P]} s_j^{[P]}), \tag{3.5}$$

$$\tau \frac{ds_j^{[F]}}{dt} + s_j^{[F]} = y_j^{[1]}, \tag{3.6}$$

$$y_i^{[2]} = f_2(\sum_j w_{ij}^{[2]} y_j^{[1]}), \tag{3.7}$$

$$[y_{out}]_i = y_i^{[3]} = f_3(\sum_j w_{ij}^{[3]} y_j^{[2]}), \tag{3.8}$$

where y_{in} and y_{out} represent the input signal and output signal in the future prediction module, and τ represents the time constant of a dynamic neuron with first-order decay property. This dynamic neuron plays the role of preserving past information in the form of an "internal state". The contents of the dynamic neurons are fed back to both the future and past prediction modules to train the desired signal transformation. In other words, as can be seen in Equation (3.4), both the internal states $s^{[F]}$ for the future prediction network and $s^{[P]}$ for the past prediction network are used for training of the future prediction module. Note that the first layer is a simple buffer; the second layer and the third layer are the nonlinear neuron layers of the normal sigmoid type; the fourth layer is the linear neuron layer to exclude limits on output values.

$$f_1(x) = f_2(x) = \frac{1}{1 + exp(-x)}, \tag{3.9}$$

$$f_3(x) = x. \tag{3.10}$$

In the past prediction module, similar state transition rules are defined for output signal in each neuron layer as $z_i^{[0]}$, $z_i^{[1]}$, $z_i^{[2]}$, $z_i^{[3]}$ $(i = 1, 2, .., n_l)$

During training, the future prediction network weights are updated based on real-time recurrent learning algorithm. The function for minimizing the error is defined as:

$$e_f = \sum_t \sum_i [y_{out}(t)]_i - d_i^{[F]}(t)^2. \tag{3.11}$$

where $d^{[F]}$ is simply the desired teacher signal and t is the time for applying the teacher signal. The error and the weight updates for the past prediction network are computed in the same way.

The bi-directional network – as well as uni-directional network – have been applied to sunspots[1] data in [95], [98] which is one of the most popular data sets often used for time series prediction tasks. The data set consisted of 280 years (A.D. 1700-1979) of normalised annual data.

3.7 Recurrent Neural Networks

Recurrent neural networks come in three different architectures [87] (see Figure 3.8). In Elman networks, feedback connections exist between hidden neurons; these hidden neurons are used to learn a representation of a dynamical system's hidden states being modelled. In Jordan networks, feedback connections in the output layer are fed back into the hidden layer. In fully recurrent neural networks, connections exist between all the network's neurons. These feedback connections enable these networks to create a memory of past events that occurred numerous time steps ago.

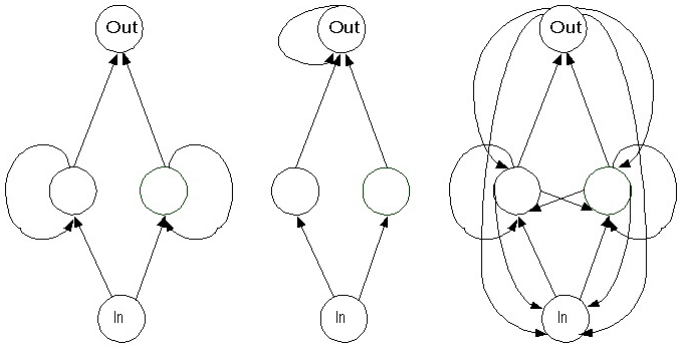

Figure 3.8: **Elman RNN, Jordan RNN and a Fully RNN**

Gradient descent learning is the most commonly used learning algorithm for RNNs. For an excellent introduction to gradient learning in RNNs (see [103]). The aim of gradient descent learning is to find the best possible set of weights in the weight space that produce the minimum margin of error. Learning in recurrent networks

[1]Sunspots are dark spots found on the surface of the sun, which are really areas of intense magnetic energy which tend to cool down the area residing within it, thus resulting in a darker appearance compared with the surrounding solar atmosphere.

is accomplished by finding the minimum of an error function E over all sequences which measures the difference between desired target outputs t_k and actual output a_k,

$$E(t) = \frac{1}{2\sum(t_k(t) - a_k(t))^2} \tag{3.12}$$

The error at a time t is calculated for a particular pattern; thus, $E(t)$ represents the sum of all the errors over all the patterns residing in the dataset. The weight are then updated according to the following rule:

$$\Delta w = -\eta E(t) \tag{3.13}$$

where η represents the learning rate constant which determines the step size in the gradient descent search. With a small learning rate, a network will take a considerable period of time to converge to the desired solution if one exists. Too large a learning rate may result in divergence; if the learning rate parameter is increased, the settling time of the network also increases which is the result of overshooting the solution. After the error signals have been calculated, they are added together and contribute to one big change for each weight. This is known as batch learning [42]. An alternative approach is on-line learning which allows the weights to be updated after each patern is presented to the network.

Potential applications of RNNs are time series prediction (e.g., of financial series), time series production (e.g., motor control in non-Markovian environments) and time series classification or labelling (e.g., rhythm detection in music and speech).

Common gradient descent based learning algorithms for recurrent networks include Backpropagation Through Time (BPTT) and Real-Time Recurrent Learning (RTRL).

3.7.1 Back-Propagation Through Time

This learning algorithm, first proposed by [78], is based on the conventional error backpropagation algorithm. The back-propagation through time (BPTT) learning algorithm computes the error gradient on a RNN that is unfolded in time. This is accomplished by creating a copy of the network for each time step. The weights are then shared amongst these copies.

The total error of the unfolded network is defined by:

$$E = \sum_{t_1}^{t_n} \sum_{j} \frac{1}{2(e_i^t)^2} \tag{3.14}$$

where e_i^t = error of node i at time $t = d_i^t - a_i^t$

e_i^t will $= 0$ if d_i^t is not specified.

The learning procedure computes weight updates as follows:

1. The forward pass for the given data is performed, and the error for each time step t is computed.

2. The error is back-propagated in order to calculate the local gradients for time step t.

$$\delta_j^t = -\frac{\delta E}{\delta I_i^t} = g'(I_i^t) e_i^t \; for \; t_1 = t_n \tag{3.15}$$

otherwise

$$\delta_i^t = g'(I_i^t)(e_i^t + \sum_j w_{ij} \delta_j^{t+1}) \tag{3.16}$$

where $g(I_i^t)$ represents the squashing function.

3. The weight change is then computed:

$$\Delta w_{ij} = -\frac{\alpha \delta E}{\delta w_i^j} = \alpha \delta \sum_{t_1}^{t_{n-1}} \delta_i^{t+1} \xi_j^t \tag{3.17}$$

where ξ_j^t represents the input to node j at time step t.

3.7.2 Real-Time Recurrent Learning

Real-time recurrent learning (RTRL) [42],[43],[102] is another gradient descent learning approach for training RNNs. This learning algorithm calculates the derivatives of states and outputs with respect to all weights in the network. This means that the network is not unfolded in time; instead, adjustments are made to the synaptic weights of the network in real time, that is, while the network continues to perform its signal processing function [102]

We define input units as: $I = x_k(t)$, where $0 \leq k \leq m$, hidden or output units as: $U = y_k(t)$, where $0 \leq k \leq n$ and arbitrary units are indexed by: $z_k(t) = x_k(t)$ if $k \in I$ or $y_k(t)$ if $k \in U$.

Let W represents the weight matrix which contains n rows and $n + m$ columns and w_{ij} will represent a weight from unit i to unit j.

The network activation for a given unit is:

$$net_k(t) = \sum_{I \in U \in J} w_{kI} z_i(t) \tag{3.18}$$

where t denotes a given time step.

The network activation is then passed through a squashing function:

$$y_k(t+1) = f_k(net_k(t))$$

A teacher signal may not be assigned for each input signal, i.e. a target is provided only for the last input in the sequence. An error defined over the output units needs to be time dependent. The reason for this is that if no target exists at a particular time step, an error produced at the output layer will be undefined or zero.

The output unit error is therefore defined as:

$$e_k(t) = d_k(t) - y_k(t) \text{ for } k \in T(t) \text{ or } 0 \text{ elsewhere}$$

where $T(t)$ is simply the set of indices in U where there exists a teacher signal $d_k(t)$. The cost or error function for a given time step is defined as:

$$E(t) = \frac{1}{2\sum_{k \in U} e_k(t)^2} \tag{3.19}$$

This error function needs to be minimized over all past steps of the network.

$$E_{total}(t_0, t_1) = \sum_{t_0+1}^{t} 1E(t) \tag{3.20}$$

The total error is now the sum of the current error and the error of the previous time steps. It then follows that E_{total} is the sum of the gradient for the preceding time steps and the current time step.

$$\nabla_w E_{total}(t_0, t+1) = \nabla_w E_{total}(t_0, t) + \nabla_w E(t+1) \tag{3.21}$$

where ∇_w is simply the gradient of w. For every sequence that is presented to the network we can compute the weight change Δ_w.

$$\Delta w_{ij}(t) = -\frac{\mu \partial E(t)}{\partial w_{ij}} \tag{3.22}$$

So each weight within the network is adjusted by:

$$\sum_{t=t_0+1}^{t_1} \Delta w_{ij}(t) \tag{3.23}$$

3.8 The Vanishing Gradient Problem

According to [10], a task will exhibit long-term dependencies if the computation of a teacher signal at a given time step depends on the input signal presented at a much earlier instance. This means that current activation states within the network influence states in the distant future.

RNNs are appropriate tools for modelling short sequences. However, training is unlikely to converge when sequences have long-term dependencies [10]. [10] further notes that the vanishing gradient problem is really the main reason why gradient descent learning is not powerful enough to discover the temporal relationship that exists between current and past inputs.

[46] analysed the problem which these networks suffer from and explains it as follows: Using the conventional BPTT algorithm devised by [102], the premise is based on the fact that we initially have a fully connected RNN whose hidden and output unit indices range from 1 to n. We note that the local error flow for arbitrary unit u at a given instant will be back-propagated for q time steps to unit v. This then results in scaling[2] the error by the following component:

$$\frac{\partial \vartheta_v(t-q)}{\partial \vartheta_u(t)} = f'_v(net_v(t-1))w_{uv} for q = 1 \tag{3.24}$$

and

$$\partial \vartheta_u(t) = f'_v(net_v(t-q)) \sum_{t=1}^{n} \frac{\partial \vartheta_l(l-q+1)}{\partial \vartheta_u(t) w_{lv}} for q > 1. \tag{3.25}$$

Thus, with l_q = v and l_o = u, we have:

$$\frac{\partial \vartheta_v(t-q)}{\partial \vartheta_u(t)} = \sum_{t_1=1}^{n} \cdots \sum_{t_{q-1}=1}^{n} \prod_{m=1}^{q} f'_{l_m}(net_{lm}(t-m)) w_{l_m l_{m-1}} \tag{3.26}$$

where $\prod_{m=1}^{q} f'_{l_m}(net_{lm}(t-m)) w_{l_m l_{m-1}}$ results in the total error flowing back in time.

[2] Adjusting the weights by a small amount at a time so as to reduce the error of the network.

Thus, if the absolute value $\left|f'_{l_m}(net_{lm}(t-m))w_{l_m l_{m-1}}\right| > 1.0$, then the error increases without bound and conflicting signals arriving at unit u will result in network instability and oscillating weight magnitude.

Additionally, if the absolute value

$$\left|f'_{l_m}(net_{lm}(t-m))w_{l_m l_{m-1}}\right| < 1.0 \tag{3.27}$$

the error tends to vanish.

Since f_{lm} represents a sigmoid function, the upper bound of f'_{lm} is 0.25. If $y^{l_{m-1}}$ is kept constant and $\neq 0$, then $\left|f'_{l_m}(net_{lm}w_{l_m l_{m-1}}\right|$ will have upper bound values where

$$w_{l_m l_{m-1}} = \frac{1}{y^{l_{m-1}} \coth(1/2 net_{lm})} \tag{3.28}$$

tends to zero for $\left|w_{l_m l_{m-1}}\right| \to \infty$ and is smaller than 1.0 for $\left|w_{l_m l_{m-1}}\right| < 4.0$. With a conventional sigmoid transfer function, the error flow will therefore diminish when the weights have absolute values below 4.0. This occurs mostly in the initial stages of the training phase.

[46] further notes that this local error flow results in the global error flow diminishing as well. It can be seen from the above analysis that the gradient descent learning technique is inadequate to deal with this problem. We will now review some solutions that various researchers have devised, most of which is described in [46] and [45].

3.8.1 Time Constants

[72] proposed the idea of time constants. A time constant affects the changes in a networks unit activations. [86] proposed an alternative approach in which the activation of a feedback unit is updated by additions of a past activation value with the current network input.

3.8.2 Ring's Approach

[76] determined that when conflicting error signals enter a unit in a network, particular error signals promote in increasing the activity of the unit by adding a higher order unit which will influence the appropriate connections. The dilemma that is confronted by this approach was that bridging gaps of n time steps may involve the addition of n units.

3.8.3 Searching Without Gradients

Network weights are randomly initialised until the resulting network is able to classify all the training patterns correctly. It has been shown by [47] that simple weight guessing solves several popular tasks faster than RNN learning algorithms.

Other proposed methods include probabilistic target propagation and adaptive sequence chunkers, which can be found in [45].

We will now direct our focus on a gradient based-method which [46] devised.

3.8.4 Long-Short Term Memory

[45] developed this model after theoretically analysing the long-term dependency problem. This model is a gradient descent based method which truncates the networks gradient. This model is described in detail in Chapter 4.

3.9 Summary

Neural networks have been successfully applied to time series modelling, i.e. time series analysis and prediction. In this chapter, we have defined what time series modelling is. We then introduced the theory of neural networks and discussed how various architectures have been applied to numerous problem domains. We then extended the idea to networks that contain feedback connections as well as learning algorithms adapted to deal with it. Furthermore, we highlighted a shortcoming of

recurrent neural networks which was evident when dealing with extremely long time series. Finally, we concluded by defining the long-term dependency problem as well as a few solutions discussed in literature. This then motivates and explains why we will be using the modelling approach discussed in Chapter 4.

Chapter 4

Long Short-Term Memory Recurrent Neural Networks

4.1 Introduction

In the previous chapter, we introduced the theory of neural networks and discussed how various architectures have been applied to numerous problem domains. The problem of RNNs in dealing with extremely long time series was mentioned and few solutions were discussed. In this chapter, we look at one of the prominent solutions – Long Short-Term Memory (LSTM) recurrent neural network.

Long Short-Term Memory (LSTM) recurrent neural network, first introduced by [46], is a gradient-based architecture specifically developed for modelling time series with long-term dependencies. This work aims to show the potentials of LSTM RNN in long time series modelling and its capabilities to discriminate different types of temporal sequences and group them according to a variety of features, it is therefore considered necessary to make reference to LSTM.

Previous successes in the real world applications of recurrent networks were limited due to practical problems of long time lag between relevant events. Difficulty arises in training recurrent neural networks from examples because their parameters often settle in sub-optional solutions which only take into account short-term dependencies and not long-term dependencies [37]. [46] proposed a novel RNN architecture and

learning algorithm to overcome the problem of long-term dependencies. The LSTM RNN architecture allows error to be back-propagated through time, and further than any other method that exists, with the exception of the echo state approach to training RNNs [50]. LSTM enforces constant error flow over extremely long temporally extended patterns.

We introduce traditional LSTM RNNs [46]. We then describe LSTM RNNs with forget gates and LSTM with peephole connections [38]; both are improvements on the traditional LSTM RNNs. Learning in LSTM RNNs is explained before we conclude with a highlight of some application domain of the architecture.

4.2 Traditional LSTM RNNs

A traditional LSTM RNN consists of an input layer, a recurrent hidden layer and an output layer (Figure 4.1). It is similar in structure to a conventional fully RNN in Figure 3.8, except that the hidden layer is replaced by a memory block layer. Each memory block contains one or more memory cells and a pair of adaptive, multiplicative gating units which gate input and output to all cells in the block. Memory blocks allow cells to share the same gates (provided the task permit this), thus reducing the number of adaptive parameters. The memory cell in Figure 4.2 has at its core a recurrently self-connected linear unit called *Constant Error Carousel* (CEC). The CEC has a weighted connection of 1.0 fed back to itself. The activation of the CEC is called the cell *state*. When the gating units of a cell produce an activation close to zero, no erroneous input enters that cell and consequently, does not contribute to the state of the cell. Thus, CEC produces an activation which reflects the state of the cell at a given instant. The CEC solves the vanishing error problem: in the absence of new input or error signals to the cell, the CEC's local error back flow remains constant, neither growing nor decaying. It thus maintain its activation state over time.

In [46], empirical evidence shows that LSTM can learn temporal patterns with long-term dependencies which traditional RNNs find difficult to learn.

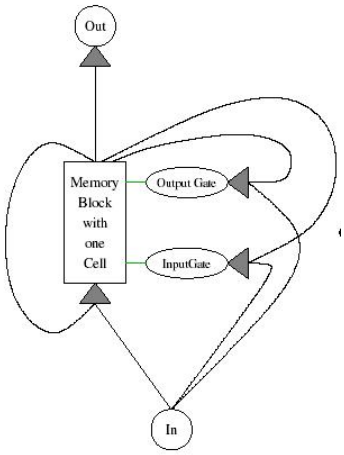

Figure 4.1: **A LSTM RNN:** The network is similar in structure to a fully RNN except that the hidden layer is replaced by a memory block

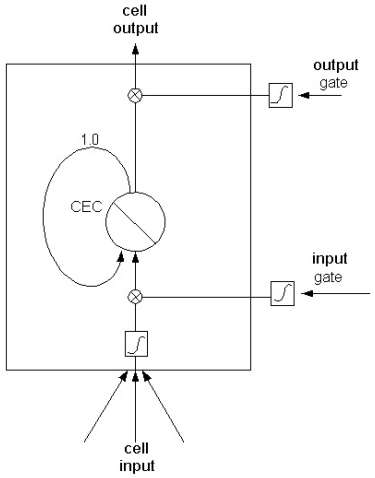

Figure 4.2: **A Memory Block with One Cell**

4.2.1 Forward Propagation

The cell state, s_c, is updated based on its current state and three sources of input: net_c is input to the cell itself while net_{in} and net_{out} are inputs to the input and output gates respectively [38]. In [46], only discrete-time steps are considered. A memory block is denoted by j and v denotes a memory cell within block j.

Activation of the Input Gate

The input to the cell is passed through a sigmoid function and is then multiplied by the activation of the input gate. Its main function is to regulate the flow of the input layer activation to the cell.

$$net_{in_j}(t) = \sum_m w_{in_j m} y^m(t-1); y^{in_j}(t) = f_{in_j}(net_{in_j}(t)). \tag{4.1}$$

Activation of the Output Gate

The output of the cell also passes through a sigmoid function and is then multiplied by the activation of the output gate. The output gate thus, regulates the flow of the output layer activation from the cell.

$$net_{out_j}(t) = \sum_m w_{out_j m} y^m(t-1); y^{out_j}(t) = f_{out_j}(net_{out_j}(t)). \tag{4.2}$$

The gating units make use of a logistic sigmoid function with range [0,1].

$$f(x) = \frac{1}{1+e^{-x}}. \tag{4.3}$$

The input to the cell itself is computed as follows:

$$net_{c_j^v}(t) = \sum_m w_{c_j^v m} y^m(t-1), \tag{4.4}$$

This activation is squashed by g, a centered logistic sigmoid function with range [-2,2].

$$g(x) = \frac{4}{1+e^{-x}} - 2. \tag{4.5}$$

The state of the memory cell s_c at a given instant, t, is computed as follows:

$$s_{c_j^v}(0) = 0; s_{c_j^v}(t) = s_{c_j^v}(t-1) + y^{in_j}(t) g(net_{c_j^v}(t)) \tag{4.6}$$

for t $\succ$ 0.

The output from the cell is computed as follows:

$$yc_j^v(t) = y^{out_j}(t) h(s_{c_j^v}(t)), \tag{4.7}$$

where h is a sigmoid function with range [-1,1]:

$$h(x) = \frac{2}{1+e^{-x}} - 1. \tag{4.8}$$

Finally, the output of the network is computed as follows:

$$net_k(t) = \sum_m w_{km} y^m(t-1), y^k(t) = f_k(net_k(t)), \tag{4.9}$$

where f_k represents the sigmoid function expressed in Equation 4.3.

This concludes the traditional LSTM's forward propagation.

4.3 LSTM RNNs With Forget Gates

[46] demonstrates that LSTM can solve numerous tasks not solvable by previous learning algorithms for RNNs. However, LSTM RNNs fail to determine how to correctly handle long time series such as those that emanate from dynamical systems [39]. If any training pattern has no clear beginning and end, the internal state values of a given memory cell could grow indefinitely, and eventually cause the network to break down. The begin and end markers of a cell allow the cells state to be reset. [39] proposed a solution to this problem by modifying the traditional LSTM RNN to forget the unit activations which represents the short term memory within the network. Forget gates are introduced to circumvent this problem.

The solution enables LSTM memory blocks to learn to reset itself once their contents are out of date and hence useless. By resets, we do not only mean immediate resets to zero but also gradual resets corresponding to slowly fading cell states. More specifically, the traditional LSTM's CEC self recurrent connection of 1.0 is replaced by a multiplicative forget gate activation y^{φ} See Figure 4.3.

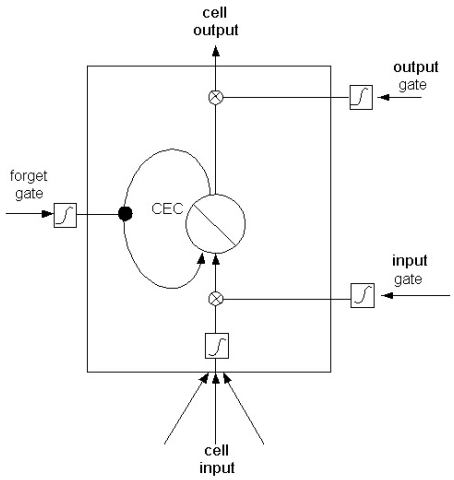

Figure 4.3: **A Memory Block with Forget Gates**

The forget gate activation, y^{φ}, is calculated like the activations of the other gates – (see Equations 4.1 and 4.2):

$$net_{\varphi_j}(t) = \sum_m w_{\varphi_j m} y^m(t-1); y^{\varphi_j}(t) = f_{\varphi_j}(net_{\varphi_j}(t)). \tag{4.10}$$

where f_{φ_j} is a sigmoid function as in Equation 4.3.

The cell state now changes slightly, which now includes the forget gate activation:

$$s_{c_j^v}(0) = 0; s_{c_j^v}(t) = y^{\varphi_j}(t) s_{c_j^v}(t-1) + y^{in_j}(t) g(net_{c_j^v}(t)), \tag{4.11}$$

for t $\succ$ 0.

4.4 LSTM RNNs With Peephole Connections

In LSTM RNN with forget gates, each gate receives connections from the input units and the output of all cells. But there exists no explicit connection from the CEC which its controls. This results in essential information being lost since the forget gate can only directly observe the output of a cell. This may harm network performance by not being able to inspect the CEC. Weighted peephole connections from the CEC to all the gates residing within the same memory block are, therefore, introduced [38],[40]. These peephole connections (Figure 4.4) shield the CEC from unwanted information during the forward and backward passes. Peephole connections are able to utilize the cells contents when decisions need to be made.

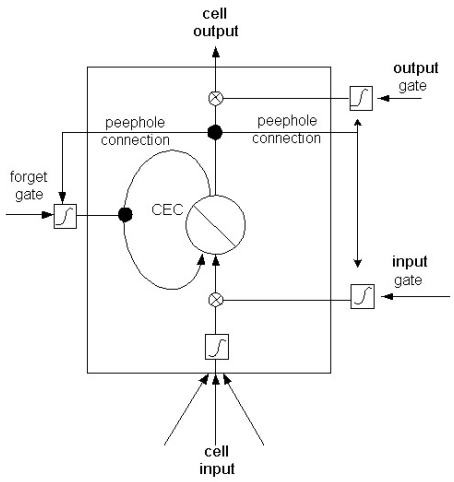

Figure 4.4: **A Memory Block with Peephole Connections**

The input and forget gate activation with peephole connections are as follows:

$$net_{in_j}(t) = \sum_m w_{in_j m} y^m(t-1) + \sum_{v=1}^{s_j} w_{in_j} c_j^v s_{c_j^v}(t-1), y^{in_j}(t) = f_{in_j}(net_{in_j}(t)) \quad (4.12)$$

$$net_{\varphi_j}(t) = \sum_m w_{\varphi_j m} y^m(t-1) + \sum_{v=1}^{s_j} w_{\varphi_j} c_j^v s_{c_j^v}(t-1), y^{\varphi_j}(t) = f_{\varphi_j}(net_{\varphi_j}(t)) \quad (4.13)$$

The output gate activation with peephole connection is computed as:

$$net_{out_j}(t) = \sum_m w_{out_j m} y^m(t-1) + \sum_{v=1}^{s_j} w_{out_j} c_j^v s_{c_j^v}(t-1), \tag{4.14}$$

$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t)). \tag{4.15}$$

The cell output is computed as:

$$yc_j^v(t) = y^{out_j}(t) s_{c_j^v}(t). \tag{4.16}$$

4.5 Learning In LSTM RNNs

Learning in LSTM RNNs is based on a combination of truncated BPTT and a modified version of RTRL which was derived in [46]. This learning algorithm includes forget gates and peephole connections which has been extended in [38] to deal with continuous time-series and precise timing problems.

Essentially, errors arriving at the net inputs of memory blocks and their gates do not get propagated back further in time, although they do serve to change the incoming weights. In essence, once an error signal arrives at a memory cell output, it gets scaled by the output gate and the output nonlinearity h; then it enters the memory cell's linear CEC, where it can flow back indefinitely without ever being changed. This is why LSTM can bridge arbitrary time lags between input events and target signals.

During the training phase, the multiplicative gates learn to open and close, thus allowing the next set of input enter the cell. The sum of square errors is defined as follows:

$$E(t) = \frac{1}{2} \sum_k e_k(t)^2; e_k(t) := t^k(t) - y^k(t) \tag{4.17}$$

where t^k represents the desired teacher signal and y_k represents the actual output of the network.

The learning procedure is as follows:

Step 1

The errors are backpropagated by engaging the following procedure:

Compute the derivative of the output units:

$$\delta_k(t) = f_k^{'}(net_k(t))e_k(t) \tag{4.18}$$

For all memory block j, compute the derivatives of the output gates:

$$\delta_{out_j} = f_{out_j}^{'}(net_{out_j}(t))(\sum_{v=1}^{s_j} h(s_{c_j^v}(t)) \sum_k w_{kc_j^v}^v \delta_k(t)) \tag{4.19}$$

For the v^{th} cells in the j^{th} block compute the cell state error:

$$e_{s_{c_j}^v}(t) = y^{out_j}(t)h^{'}(s_{c_j{}^v}(t))(\sum_k w_{k_{c_j}^v} \delta_k(t)) \tag{4.20}$$

Step 2

Update the weights within the network:

Output unit weight updates:

$$\Delta w_{km}(t) = \alpha \delta_k(t) y^m \tag{4.21}$$

For all memory blocks j

Output gate weight updates:

$$\Delta w_{out,m} = \alpha \delta_{out} y^m; \Delta w_{out,c_j^v} = \delta_{out} s_{c_j^v}; \tag{4.22}$$

Input gate weight updates:

$$\Delta w_{in,m} = \alpha \sum_{v=1}^{s_j} e_{s_{c_j^v}} ds_{in,m}^{jv} \tag{4.23}$$

For all peephole connections, $v^{'}$

$$\Delta w_{in,m} = \alpha \sum_{v=1}^{s_j} e_{s_{c_j^v}} ds_{in,c_j^{v'}}^{jv} \tag{4.24}$$

Forget gate weight updates:

$$\Delta w_{in,c_j^{v'}} = \alpha \sum_{v=1}^{s_j} e_{s_{c_j^v}} ds_{\varphi,m}^{jv} \tag{4.25}$$

For all peephole connections, $v^{'}$

$$\Delta w_{\varphi,c_j^v} = \alpha \sum_{v=1}^{s_j} e_{s_{c_j^v}} ds_{\varphi,c_j^{v'}}^{jv} \tag{4.26}$$

Update the cells for the v^{th} cells in the j^{th} block:

$$\Delta w_{c_j^v,m} = \alpha e_{s_{c_j^v}} ds_{cm}^{jv} \tag{4.27}$$

LSTM learning algorithm is local in both space and time. It has a computational complexity per time step and weights of $O(1)$, that means $O(n^2)$ where n represents the number of hidden units. This is in essence determined by the network topology.

It is important to note that learning in LSTM RNNs as described above is supervised learning. Recent efforts have been made to exploit the computational ability of LSTM RNNs for unsupervised learning [53]. We are particularly interested in unsupervised learning of LSTM RNNs, where a good task-independent representation for a given set of data must be found.

As noted above, LSTM RNNs are typically trained, in a supervised learning fashion, based on truncated BPTT and a modified version of RTRL so as to minimize an objective function (also called error or loss function) that measures the network's performance. In supervised learning problems, there exist an explicit target for each

output node and input pattern. In unsupervised learning, however, there is no target available to tell the network what to do. In the absence of a teacher, what should the network's objective be? The progress of these unsupervised networks must be evaluated in a task-independent manner, using general principles of efficient coding [80]. This is the domain of information theory, and we can therefore derive objective functions for unsupervised learning from information-theoretic arguments.

[53] report on certain information theoretic objective functions for unsupervised learning that can be plugged into the LSTM RNN for unsupervised learning task. We now describe information theoretic models in the following section.

4.6 Information Theoretic Models

The concept of information theory first developed by [83] is a deep mathematical theory that is concerned with the very essence of communication process [42]. The theory provides a framework for the study of fundamental issues – such as the efficiency of information representation, and the limitation involved in the reliable transmission of information over a communication channel. Models derived from the theory are referred to as information theoretic models. [80] reports on the neural network implementation of an information-theoretical model in an unsupervised learning fashion using parametric modelling, probabilistic networks and nonparametric estimation. For a review of information-theoretic models see [42]. [26] also discusses some information theory models suitable for data mining.

Two information theoretic models for unsupervised learning are binary information gain optimization (BINGO) and nonparametric entropy optimization (NEO). We now describe these models in the following sections.

4.6.1 Binary Information Gain Optimization (BINGO)

BINGO, developed by [81], is a parametric unsupervised learning algorithm which clusters unlabelled data with linear adaptive discriminants stressing the gaps between clusters [81, 80]. The method maximizes the information gained from observing output of a single-layer network of logistic nodes, interpreting their activity as stochastic binary variables. The resulting weight update formula (see [81, 80] for a complete derivation) for node i is given by

$$\Delta w_i \; \alpha \; f^{'}(y_i)x(y_i - \hat{y}_i), \tag{4.28}$$

where $f^{'}(y_i)$ is the derivative of the logistic squashing function f, x the presynaptic input, and $\hat{y}_i$ the difference between actual and estimated network output. A linear second-order estimator is used for multiple outputs:

$$\hat{y} = y + (Q_y - 2I)(y - \bar{y}), \tag{4.29}$$

where $\bar{y}$ denotes the average of y over the data, and Q_y its autocorrelation. The binary information gain is maximal (namely, 1 bit/node) when a network's outputs are uncorrelated, and approach '1' (resp. '0') for half the data points. Thus BINGO seeks to identify independent dichomoties in the data.

4.6.2 Nonparametric Entropy Optimisation (NEO)

NEO, developed by [93], – in contrast to parametric unsupervised learning techniques – is a differential learning rule that optimizes signal entropy by way of kernel density estimation; thus, no additional assumptions about a density's smoothness or function form are necessary [80], [93]. The nonparametric *Parzen window* density estimate $\hat{p}(y)$is given by:

$$\hat{p}(y) = \frac{1}{\mid T \mid} \sum_{y_j \epsilon T} K_\sigma(y - y_j), \tag{4.30}$$

where T is a sample of points y_j and K_σ is a kernel function, in our case an isotropic Gaussian with variance σ^2. The kernel width σ that best regularizies the density

estimate can be found by maximizing the log-likelihood:

$$\hat{L} = \sum_{y_i \epsilon S} log \sum_{y_j \epsilon T} K_\sigma(y_i - y_j) - \mid S \mid log \mid T \mid, \tag{4.31}$$

whose derivative with respect to the kernel width is given by:

$$\frac{\partial}{\partial \sigma} \hat{L} = \sum_{y_i \epsilon S} \frac{\sum_{y_j \epsilon T} \frac{\partial}{\partial \sigma} K_\sigma(y_i - y_j)}{\sum_{y_j \epsilon T} K_\sigma(y_i - y_j)}. \tag{4.32}$$

The maximum likelihood kernel makes a second S derived from $p(y)$ most likely under the estimated density $\hat{p}(y)$ computed from the sample T. [1] A nonparametric estimate of the empirical entropy (given optimal kernel shape) of a neural network's output y can then be calculated as:

$$\hat{H}(y) = -\frac{1}{\mid S \mid} \sum_{y_i \epsilon S} log \sum_{y_j \epsilon T} K_\sigma(y_i - y_j) + log \mid T \mid, \tag{4.33}$$

and minimized by adjusting the neural network's weights w:

$$\frac{\partial}{\partial w} \hat{H}(y) = -\frac{1}{\mid S \mid} \sum_{y_i \epsilon S} \frac{\sum_{y_j \epsilon T} \frac{\partial}{\partial w} K_\sigma(y_i - y_j)}{\sum_{y_j \epsilon T} K_\sigma(y_i - y_j)}. \tag{4.34}$$

Low $\hat{H}(y)$ is achieved by clustering the data. See [80, 93] for further details.

In this thesis, we train LSTM recurrent neural networks to maximize NEO (Section 5.4.2).

4.7 Applications of LSTM RNNs

4.7.1 Blues Improvisation

[28] showed that LSTM RNNs are able to learn a form of blues music and are able to compose novel melodies with the same style. [28] notes that the compositions of music have a distinct global temporal structure in the form of nested periodicities. Music therefore has some notes that are more distinctive than others. A LSTM

[1] To make efficient use of available training data, we let y_j in the inner sums of Equations 4.32 and 4.34 range over $T = S \mid \{y_i\}$ for each $y_i \epsilon S$ in the outer sum [80]

RNN is able to successfully learn to predict notes at time $t + 1$ by making use of input data at times $\leq t$.

4.7.2 Automatic Speech Recognition

Since traditional RNNs suffer from the vanishing gradient problem (see Section 3.8), they are unable to learn correlations between inputs and errors which span long time intervals, [27]. This then leads to general failure to find long-term dependencies spanning several phones [2] [27]. [27] notes that traditional RNNs are not able to discover transition probabilities among sequences of words at a very slow timescale. The authors note that even at faster timescales, time warping and co-articulation effects tend to stretch phones which in turn blur their boundaries. LSTM seeks to address these problems that traditional RNNs face. A LSTM RNN maps every frame of an acoustic speech signal onto a set of fixed phone targets. The training involves using a collection of hand-labelled data. Two LSTM RNNs are used: the first network estimates the frame-level phone probability; the second network computes a mapping of the phone predictions into words; i.e. when the network is trained, it predicts sequences of words from sequences of phones which has been obtained from the first network. Results indicate that LSTM RNNs perform well at the frame-level phone prediction.

4.7.3 Named Entity Recognition

Involves identifying atomic elements of information in text, such as names, locations and monetary values, etc. [41] trained a LSTM RNN on English and German atomic elements. A self-organising map for sequences is used to generate representations for the lexical items presented to the network [41]. The network is trained to output a vector which represents a particular tag, i.e. for the tag O a vector representation of 0100000 is produced. Promising results were yielded. The reader can consult [41] for the results.

[2] A small unit of speech sound that assists to distinguish one word from another.

4.8 Summary

Long Short-Term Memory RNNs are extensions of RNNs. They are able to overcome the long-term dependency problem which traditional RNNs suffer from. This chapter provided a short tutorial on how LSTM RNNs work and briefly highlighted some tasks that LSTM has been applied to. This will be useful in understanding the next chapter which applies LSTM RNNs to the profiling of calls made by users over a period of time in a mobile telecommunication network.

Chapter 5

Call Pattern Analysis in Mobile Telecommunication Networks

5.1 Introduction

In this chapter, we present a brief overview of data mining. We then present our experiments based on the unsupervised learning modeling approaches that we applied for profiling of calls made by users over a period of time in a mobile telecommunications network. We discuss and conclude with findings from our experiments.

5.2 Data Mining

Knowledge discovery in databases (KDD) deals with data integration techniques and the discovery, interpretation, and visualization of patterns in large collections of data. An integral part of the KDD process is data mining. It entails the search for valuable information in large volumes of data through the exploration and analysis, by automatic or semi-automatic means, of the large quantities of data in order to discover meaningful patterns and rules [23]. Data mining originated from database technology, statistics, machine learning (AI), visualization and traditional techniques. Data mining tasks are generally of two forms:

1. Predictive Tasks – use some variables to predict unknown or future values of other variables.

2. Descriptive Tasks – find human-interpretable patterns that describe the data.

This thesis focuses on the descriptive data mining tasks. [67] discusses the foundation as well as the practical side of intelligent agents and their theory and applications for data mining and information retrieval in large databases.

Neural networks are of particular interest as data mining techniques for time series, because they offer a means of efficiently modeling large and complex problems in which there may be hundreds of predictor variables that have many interactions.

5.3 Self-Organizing Maps

The self-organizing map (SOM), developed by Kohonen [56], is a neural network model for the analysis and visualization of high dimensional data. It projects the nonlinear statistical relationships between high-dimensional data into simple topological relationships on a regular, typically two-dimensional grid of nodes. The SOM thereby compacts information while preserving the most important topological and/or metric relationships of the primary data elements on the two-dimensional plane. SOMs have been successfully applied in the development of adaptive devices for various telecommunication applications [55]. See [52, 73] for a bibliography on published papers.

The basic SOM model is a set of prototype vectors, with a defined neighbourhood relation. This neighbourhood relation defines a structured lattice, which may be linear, rectangular or hexagonal arrangement of map units. The SOM is formed through an unsupervised, competitive learning process. This process is initiated when a winner unit is searched, which minimizes the Euclidean distance measure between data samples x and the map units m_i. This unit is described as the best-matching node, signified by the subscript c:

$$\begin{aligned} \| x - m_c \| = \min_i \{\| x - m_i \|\}, \quad or \\ c = \arg\min_i \{\| x - m_i \|\}. \end{aligned} \tag{5.1}$$

Then, the map units are updated in the *topological* neighborhood of the winner unit, which is defined in terms of the lattice structure. The update step can be performed by applying

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{5.2}$$

where t is an integer, the discrete-time coordinate, $h_{ci}(t)$ is the so-called *neighbourhood kernel*; it is the function defined over the lattice points. The average width and form of h_{ci} defines the "stiffness" of the "elastic surface" to be fitted to the data points. In addition, the last term in the square brackets is proportional to the gradient of the squared Euclidean distance $d(x, m_i) = \| x - m_i \|^2$. The learning rate $\alpha(t) \in [0, 1]$ must be a decreasing function of time and the neighborhood function $h^c(t, i)$ is a non-increasing function around the winner unit defined in the topological lattice of map units. A good candidate is a Gaussian around the winner unit defined in terms of the coordinates $\mathbf{r}$ in the lattice of neurons

$$h^c(t, i) = exp(-\frac{\| r^i - r^c \|^2}{2\sigma(t)^2}). \tag{5.3}$$

Some other neighborhood functions are discussed in [56]. During training, the learning rate and the width of the neighborhood function are decreased, typically in a linear fashion. The map then tends to converge to a stationary distribution, which approximates the probability density of the data.

After the input samples have been presented, and the codebook vectors have converged, the map is calibrated. Calibration of the map is done to locate images of different input data items on it. In practical applications, it may be self-evident to note how a particular input data set ought to be interpreted and labeled. By

providing a number of typical, manually analysed data sets, searching for the best matches on the map, and labeling the map units correspondingly, the map becomes calibrated. Since the mapping is assumed to be continuous along a hypothetical "elastic" surface, the closest reference vectors approximate the unknown input data. A number of ways of improving the performance of the SOM algorithm and a number of variants of the SOM are presented in [56].

Self-organizing maps may be visualized by using a unified distance matrix representation [91], where the clustering of the SOM is visualized by calculating distances between the map units locally and representing these with gray levels. Another choice for visualization is Sammon's mapping [79] which projects the high-dimensional map units on a plane by minimizing the global distortion of inter point distances when applying the mapping. However, in this work, we used a Kohonen Extension Map developed by [54].

5.4 Experiments

We report on our experiments using self-organizing maps and LSTM recurrent neural networks to demonstrate the unsupervised learning potentials of the two neural network architectures for call profiling over a period of time in a mobile telecommunication network. Experiment 1 is a report on our published work [4, 6], which investigates the capabilities of SOM while Experiment 2 reports on unsupervised training of the LSTM architecture.

5.4.1 Experiment 1: Self Organizing Map Model

We applied self-organizing maps to unsupervised classification of call data for prepaid service subscribers from a real mobile telecommunication network. The data set contains the call data of 500 *masked* subscribers over a period of 6 months. The data set was first *normalized*. Subsequently, we extracted from the data set Mobile Originating Calls (MOC). These are calls that were initiated by the subscribers. Within the 6 months period, a total of 227,318 calls originated from the 500 subscribers.

The final training records after normalization contained the following fields

1. Subscriber number (MSISDN)
2. Other party called
3. Cell ID in use by the subscriber
4. Area code for the location of the subscriber
5. Date and time the call was made
6. Duration of the call

The subscriber number was not used in training but to identify the SOM of each subscriber.

Feature Extraction

We generated a feature vector for each subscriber. However, there were symbolic data present in our training data. These include the other party called, cell ID in use by the subscriber and the area code for the location of the subscriber. In order to convert these to numeric values, we constructed a frequency table for each of the symbolic data per subscriber. Each symbol in the frequency table for each subscriber was then ranked based on the individual frequencies. This ranking was then used as the corresponding numeric value for the symbol.

The field for date and time was also transformed. This was converted to peak (7am to 8pm) and off-peak (8pm to 7am) periods. Peak period were represented by 1, and off peak represented by 2. Thus we have a five dimensional feature vector. The number of training examples available for each subscriber varies. The subscriber with the maximum number of training examples had 7007 and the subscriber with the minimum number of examples had 3.

Construction of the Maps

The maps for each subscriber were generated using the standard SOM algorithm [54]. A sample of the input features are shown in Table 5.1. After training, a number of outputs are generated. The corresponding weights in a trained SOM are shown in Table 5.2. A graphical view of the map is depicted in Figure 5.1.

C1	C2	C3	C4	C5	C6	C7	C8	C9
1	3	6	6	5	4	2	4	5
4	3	6	6	5	2	5	4	1
2	3	3	3	3	1	3	2	3
1	1	1	1	1	1	1	1	1
11	22	110	259	18	119	39	199	70

Table 5.1: **A set of 9 feature vectors for a user transactions in 6 months**: The calls made are labeled C1-C9 for traceability.

	X1	X2	X3	X4	X5
Y1	4.896766	4.755345	4.193862	5.005420	5.490612
	1.103238	1.305285	3.128322	5.005420	5.490612
	2.793532	2.541002	2.048763	2.502710	2.745306
	1.000000	1.000000	1.000000	1.000000	1.000000
	75.058495	84.413177	162.387711	229.162582	243.718369
Y2	3.915803	4.298299	4.281914	4.501860	5.010276
	2.442353	1.917373	2.908486	4.439574	5.010276
	2.980540	2.836195	2.158985	2.219787	2.505138
	1.000000	1.000000	1.000000	1.000000	1.000000
	59.373951	68.149834	152.800949	211.564301	229.308273
Y3	3.122192	3.497991	3.919303	3.922408	4.042710
	3.584075	3.116735	3.086745	3.322197	2.941889
	3.000000	2.901704	2.122239	1.683334	1.470946
	1.000000	1.000000	1.000000	1.000000	1.000000
	40.156727	49.095196	131.155838	162.034576	153.066940
Y4	2.653057	3.057090	3.330011	4.479967	4.837338
	4.011194	4.153870	4.072011	3.789629	3.674676
	2.765450	2.791516	2.671162	1.954079	1.837338
	1.000000	1.000000	1.000000	1.000000	1.000000
	22.448343	23.540915	43.343700	106.291168	115.231987
Y5	2.601324	3.252775	3.506822	4.996550	5.340620
	3.995059	4.223461	4.364476	4.770777	4.681239
	2.540669	2.699617	3.000000	2.440937	2.340619
	1.000000	1.000000	1.000000	1.000000	1.000000
	15.986089	17.336380	25.421503	104.627922	112.967209

Table 5.2: **Trained Weight Vectors for Input features in Table 5.1**

Results

A traditional Kohonen map groups similar input vectors together. Distance implies difference, but nearness does not imply resemblance. However, from the map shown in Figure 5.1, this is made visible on the map by calculating the square difference between neighbouring units of the trained map. Consequently, this value is used to colour the edge separating the units. Hence dark lines indicate strong difference and light lines indicate strong resemblance.

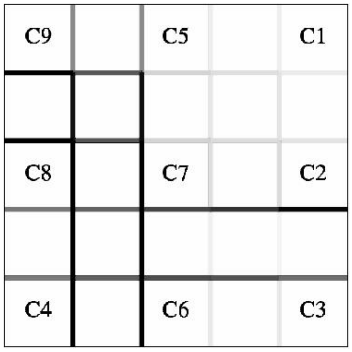

Figure 5.1: **An extended Kohonen Map showing difference and resemblance between map units**

Interpretation of Results

We need to distinguish clusters on the map which indicate normal and abnormal call pattern behaviour, respectively. Generally, clusters that tend to the corners of the map, especially with large distances (dark lines) seem to indicate abnormal behaviour; for instance, C4 and C8 in Figure 5.1. On inspection of the original data, it was discovered that these calls actually showed behaviour that was unusual, e.g. unusually long call duration. However, the only way we can validate our assumption is through a feedback procedure in collaboration with the fraud analysts to further investigate the knowledge extracted. The feedback will be used to label the clusters appropriately.

Discussion

Though we were able to represent the call profile of a mobile phone subscriber using the SOM, it must be pointed out that information on the temporal nature of the call transactions was lost in the process.

Further, the fact that different map sizes were generated for the different subscribers present a challenge when it comes to comparing the call profiles of different subscribers.

Different maps (with same sizes) are generated for the same subscriber with different runs of the algorithm. This is due to the stochastic nature of the SOM. This means that the accuracy of the map depends on the number of iterations of the SOM. Nonetheless, the algorithm preserves the topological property of the map. Observations that are close to each other in the input space (at least locally) remain close to each other in the SOM (see Figure 5.2).

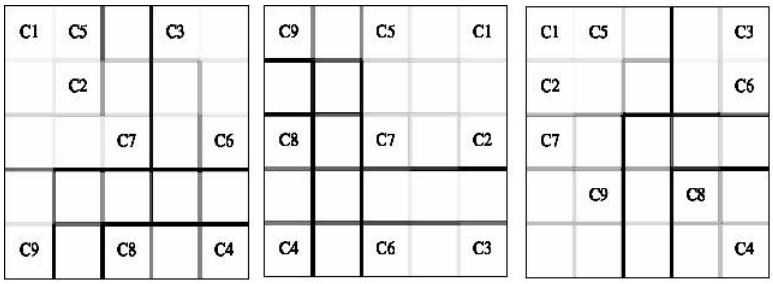

Figure 5.2: **Extended Kohonen Maps for 3 runs on the same input vector of size 9 showing the preservation of the topological property of the maps**

Also, an important question that arises about the map is its reliability. From our result, the SOM algorithm can be used for getting insight into the call data and for the initial search of potential dependencies. In general, the findings need to be cross-validated with other principled statistical methods, in order to assess the confidence of the conclusions and to reject those that are not statistically significant. The works of [59, 54, 13] offer suggestions in this regard.

5.4.2 Experiment 2: Unsupervised Training of LSTM Recurrent Neural Networks

We applied LSTM recurrent neural networks trained with NEO objective function (Section 4.6.2) to unsupervised discrimination of the same call data used as in experiment 1. The subscriber number was also not used in training but to identify the network outputs for each subscriber.

Feature Extraction

A feature vector was generated for each subscriber as in experiment 1. The field for date and time was transformed into discrete time steps to represent the temporal order of the sequence. Thus we have a five dimensional feature vector. The feature vectors for each subscriber were saved in a text file where it can be read by the LSTM RNN model. Each vector was labeled for traceability. An example of a subscriber input feature vector with 9 call transactions in 6 months is shown in Table 5.3.

C1	C2	C3	C4	C5	C6	C7	C8	C9
1	3	6	6	5	4	2	4	5
4	3	6	6	5	2	5	4	1
2	3	3	3	3	1	3	2	3
1	2	3	4	5	6	7	8	9
11	22	110	259	18	119	39	199	70

Table 5.3: **A set of 9 feature vectors for a user call transactions in 6 months with discrete time steps to reflect the temporal order of the sequence**: The calls made are labeled C1-C9 for traceability.

Network Architecture and Training

The LSTM RNN trained with NEO algorithm (Section 4.6.2) consisted of one single-cell memory block (this appeared sufficient to learn the task). The single linear output unit produced just one real value for each sequence. After training the network clusters the output at the end of each sequence, according to the different features between the sequences.

In our experiment, we used LSTM network model with forget gates (Section 4.3). The network was trained until further training did not noticeably improve the result. An average of 10 runs was made for the training with the result of each run falling within the 90% confidence interval.

Results

After training, the network's outputs form clusters of related call patterns. The clusters becomes denser based on the number of calls for each subscriber. Tables 5.4, 5.5, 5.6 and 5.7 show examples of the outputs produced by the network for 4 subscribers with 9, 13, 22 and 30 call transactions respectively in the 6 months period. See Figure 5.3 for Table 5.4, Figure 5.4 for Table 5.5, Figure 5.5 for Table 5.6 and Figure 5.6 for Table 5.7.

C1	0.99589163
C2	0.982752837
C3	0.995229964
C4	0.930946741
C5	0.985136358
C6	0.970881254
C7	0.980531615
C8	0.957798918
C9	0.992584693

Table 5.4: **Trained network output for a subscriber with 9 call transactions**

C1	0.990109443
C2	0.975707756
C3	0.990109443
C4	0.991368568
C5	0.88085985
C6	0.991364486
C7	0.880814611
C8	0.981987861
C9	0.966746684
C10	0.886485079
C11	0.989465226
C12	0.989029876
C13	0.990109443

Table 5.5: **Trained network output for a subscriber with 13 call transactions**

C1	0.999972757
C2	0.999841905
C3	0.999879332
C4	0.999983157
C5	0.994692222
C6	0.999983299
C7	0.999970841
C8	0.998391782
C9	0.999924911
C10	0.999950676
C11	0.999904136
C12	0.998965471
C13	0.999960762
C14	0.999948665
C15	0.997908185
C16	0.985477198
C17	0.981955605
C18	0.906226842
C19	0.999720082
C20	0.999835145
C21	0.999983299
C22	0.999147025

Table 5.6: **Trained network output for a subscriber with 22 call transactions**

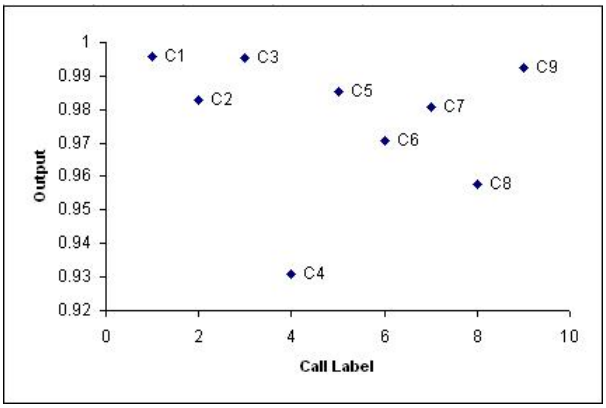

Figure 5.3: **Trained network output for a subscriber with 9 call transactions**

C1	0.999878127
C2	0.999945895
C3	0.999675456
C4	0.99982513
C5	0.999689383
C6	0.9997588
C7	0.994916759
C8	0.997774856
C9	0.999952796
C10	0.999915842
C11	0.999860702
C12	0.999945895
C13	0.981674747
C14	0.967777243
C15	0.999945895
C16	0.998871244
C17	0.999886581
C18	0.999922937
C19	0.880155866
C20	0.999876606
C21	0.996018245
C22	0.99946733
C23	0.999477189
C24	0.999817685
C25	0.999698839
C26	0.999895133
C27	0.999901345
C28	0.999681177
C29	0.994916759
C30	0.999895133

Table 5.7: **Trained network output for a subscriber with 30 call transactions**

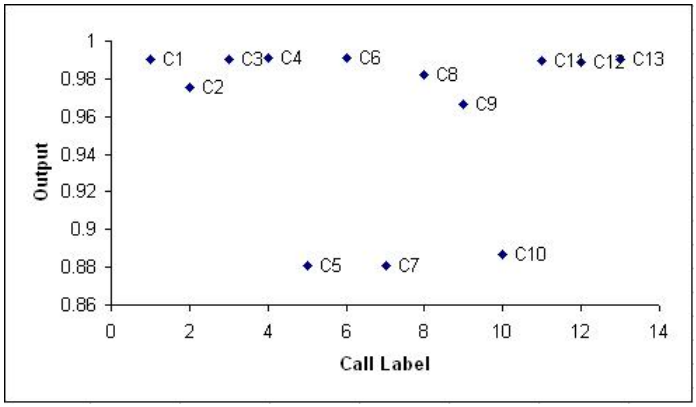

Figure 5.4: **Trained network output for a subscriber with 13 call transactions**

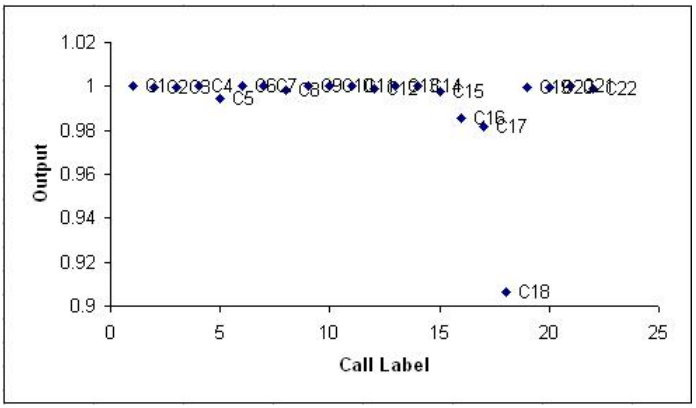

Figure 5.5: **Trained network output for a subscriber with 22 call transactions**

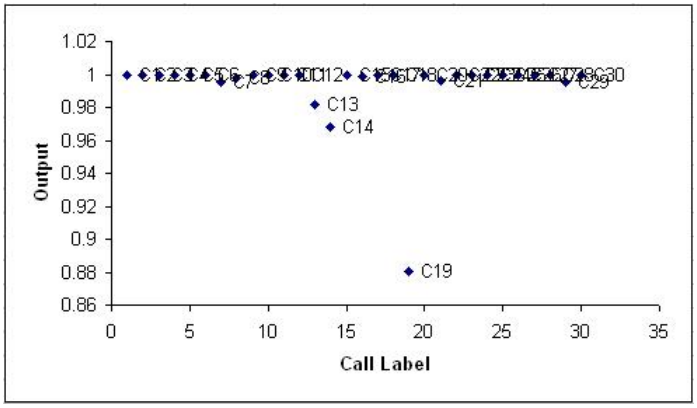

Figure 5.6: **Trained network output for a subscriber with 30 call transactions**

Interpretation of Results

We noticed that certain call patterns are prominent, especially those that tend more towards the horizontal axis. For instance C4 in Figure 5.3, C5, C7 and C10 in Figure 5.4, C18 in Figure 5.5 and C19 in Figure 5.6. On inspection of the original data, it was discovered that these calls actually showed behaviour that were unusual, e.g. unusually long/short call duration, repeated calls made to a particular destination, and calls made from a particular location.

Tables 5.8, 5.9, 5.10 and 5.11 show examples of the clusters discovered by the LSTM

network in Figures 5.3, 5.4, 5.5 and 5.6 with the salient features [1] that were identified for these clusters on inspection of the original data.

Cluster	Call Label	Salient Features	Comment
1	C4	4	This call had the longest call duration
2	C8	3 and 4	
3	C6	3 and 4	
4	C2, C5, C7	3	
5	C1, C3, C9	2 and 3	

Table 5.8: **Clusters Identified by the LSTM network for Figure 5.3**

Cluster	Call Label	Salient Features	Comment
1	C5, C7, C10	3	These calls were made from the same location
2	C9	1, 2, 3 and 4	
3	C2, C8	2 and 3	
4	C1, C3, C4, C6, C11, C12, C13	2 and 3	

Table 5.9: **Clusters Identified by the LSTM network for Figure 5.4**

Cluster	Call Label	Salient Features	Comment
1	C18	1, 2, 3 and 4	This call differs from others on all its call features
2	C16, C17	3 and 4	
3	C5	1 and 4	
4	C1, C2, C3, C4, C6, C7, C8, C9, C10, C11, C12, C13, C14, C19, C20, C21, C22	3 and 4	

Table 5.10: **Clusters Identified by the LSTM network for Figure 5.5**

[1] 1 is for Other party called, 2 is for Cell ID in use by the subscriber, 3 is for Area code for the location of the subscriber, 4 is for duration of call

Cluster	Call Label	Salient Features	Comment
1	C19	4	This call has an usually short call duration
2	C14	1, 2 and 3	
3	C13	1, 2, 3 and 4	
4	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C15, C16, C17, C18, C19, C20, C21, C22, C23, C24, C25, C26, C27, C28, C29, C30	1, 2 and 3	C21 has the longest call duration

Table 5.11: **Clusters Identified by the LSTM network for Figure 5.6**

In order to cross validate our assumption, a feedback procedure in collaboration with the fraud analysts is necessary to further investage the knowledge extracted. The feedback will be used to label the clusters appropriately.

Discussion

It was observed from training that the results were more difficult to obtain as the number of call patterns increased for the subscribers. This suggests that the correct discriminants were harder to find for increased number of call patterns.

In unsupervised models, learning becomes possible because there is redundancy [2] in the input data stream [8]. Since most naturally occurring phenomena are not random, any redundancy that occurs in the distribution of the data set implies that this set does not fill the data space uniformly. Thus, redundancy in the data suggests the existence of structure in the data. Consequently, by using LSTM RNN model to represent the call profile of the mobile subscribers in an unsupervised learning fashion, we attempted to preserve the temporal nature of the sequence of call transactions, hence conserving the redundancy within the data set more accurately than the SOM model.

We also observed that, as the number of training sequences increased per subscriber, a better discrimination was noticeable.

[2] Redundancy is the property of the input stimuli that distinguishes them from random noise

In addition, because the number of training examples available for each subscriber varies, that presents a challenge when it comes to comparing the call profiles of different subscribers.

5.5 Assessment of Models

While SOM shows information on neighbouring units of the trained map, it provides less information on salient features between call patterns as LSTM does.

SOM help to get insight into the call data and for initial search for potential dependencies but LSTM, while preserving the temporal order of the sequence, brings out the salient features in related call patterns.

LSTM provides a better clustering of the call patterns more than SOM and its output is available in a better human-interpretable fashion.

In accordance with descriptive data mining tasks, LSTM finds human-interpretable patterns that describe the data more effectively than SOM.

The longest training patterns for a subscriber trained by the LSTM RNN model was 7007. [10] state that traditional RNNs are notoriously difficult to train, especially when the interval between relevant events in the input sequence exceed about 10 time steps. [46] also show that LSTM has been shown to bridge minimal time lags in excess of 1000 discrete time steps. Consequently, our finding suggest LSTM RNN model as an instant appeal to the problem of call profiling of user transactions with long time series, over SOM and other traditional RNNs.

5.6 Summary

We have presented the results of our experiments, which have enabled us to deduce a number of conclusions based on applying SOM and LSTM RNNs – in an unsupervised learning fashion – to the task of modelling subscriber call transactions time series data in a mobile telecommunication network.

Chapter 6

Conclusions and Directions for Future Research

6.1 Conclusion

In [8], it was noted that unsupervised neural networks can mainly be used in exploratory data analysis. They act on unlabelled data in order to extract an efficient internal representation of the structure implicit in the data distribution.

In accordance, the main focus of this thesis has been to investigate the unsupervised learning potentials of two neural networks - Self-Organizing Maps (SOM) and Long Short-Term Memory (LSTM) recurrent neural networks for the profiling of calls made by users over a period of time in a real mobile telecommunication network in order to conduct a descriptive data mining on the users call patterns.

Our investigation shows the learning ability of both neural networks to discriminate user call patterns with the LSTM recurrent neural network algorithm providing a better discrimination of call patterns than the SOM algorithm in terms of long time series modelling.

To our knowledge, LSTM RNN has not been used as a basis where unsupervised learning in a dynamic model fashion is applied to unlabelled call data.

In chapter 5, we have presented the results of our experiments. These results have enabled us to draw a number of conclusions of relevance to unsupervising learning capabilities of SOM and LSTM RNN as part of the fraud detection process in mobile telecommunication networks. We have shown that LSTM RNN is able to model a subscriber's call transactions over a period of time with a relatively high degree of accuracy. Also, this study provided us with a first time look at the application of LSTM RNNs, in an unsupervised learning fashion for call pattern analysis in mobile telecommunication networks.

6.2 Directions for Future Research

6.2.1 LSTM RNN Trained With Other Objective Functions

It would certainly be useful and interesting to see LSTM RNN trained with other unsupervised learning objective function applied for call pattern analysis in mobile telecommunication network. [53] reported on the training of LSTM RNN with a unsupervised learning objective function – BINGO (Section 4.6.1)

Other objective functions for unsupervised learning can also be used, resulting in promising techniques specially for those tasks involving unsupervised detection of input sequence features spanning long time periods.

6.2.2 LSTM RNN for Predictive Data Mining Tasks

The ideas presented in this study may be used for clustering call patterns in order to label them as normal or abnormal. The labeled data set could then be used to learn a time-series classifier, for instance a LSTM recurrent neural network model, in a supervised fashion. Our investigation suggests that LSTM RNN would do better than other temporal classifiers if we had labelled data. We intend to pursue these tracks in future work [3, 5]

Finally, combining the advantages of our approaches with a recurrent neural network model should result in promising techniques for numerous real-world tasks involving detection of input sequence features spanning extended time periods.

Bibliography

[1] Phone fraud: The battle rages on. *Communication News*, 33(8):8, 1996.

[2] Media releases. In *Communications of Vibrant Solutions*, Available at http://www.vibrantsolutions.com/20020422-Cerebrus.asp, April 2002.

[3] O.A. Abidogun and C. W. Omlin. Intelligent fraud detection - work in progress. In *Proceedings of the 7th Southern African Telecommunication Networks & Applications Conference (SATNAC 2003)*, page 10, Southern Cape, South Africa, September 2003.

[4] O.A. Abidogun and C. W. Omlin. Call profiling using self–organizing maps. In *Proceedings of the 4th International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA 2004)*, pages 371–379, Gold Coast, Australia, July 2004.

[5] O.A. Abidogun and C. W. Omlin. Intelligent fraud detection. In *Proceedings of the 1st African Conference on the Digital Common (Idlelo 2004)*, page 19, University of the Western Cape, Bellville, Cape Town, South Africa, January 2004.

[6] O.A. Abidogun and C. W. Omlin. A self organizing maps model for outlier detection in call data from mobile telecommunication networks. In *Proceedings of the 8th Southern African Telecommunication Networks and Applications Conference (SATNAC 2004)*, page 4, South Western Cape, South Africa, September 2004.

[7] A. D. Back and A. C. Tsoi. FIR and IIR synapses, a new neural network architecture for time series modelling. *Neural Computation*, 3:375–385, 1991.

[8] G. A. Barreto, A. F. R. Araujo, and S. C. Kremer. A taxonomy for spatiotemporal connectionist networks revisited: The unsupervised case. *Neural Computation*, 15(6):1255–1320, June 2003.

[9] P. Barson, S. Field, N. Davey, G. McAskie, and R. Frank. The detection of fraud in mobile phone networks. *Neural Network World*, 6(4):477–484, 1996.

[10] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[11] V. Blavette. Application of intelligent techniques to telecommunication fraud detection. In *European Institute for Research and Strategic Studies in Telecommunications*, Public Project 2000, page 1, Available at http://www.eurescom.de/public/projects/p1000-series/P1007/default, May 2001.

[12] U. Bodenhausen and A. Waibel. The tempo2 algorithm: Adjusting time-delays by supervised learning. In J. E. Moody, R. P. Lippmann, and D. S. Touretzky, editors, *Proceedings of Advances in Neural Information Processing Systems*, Vol. 3, pages 155–161, Denver, USA, 1991. Morgan Kaufmann.

[13] E. Bodt, M. Cottrel, and M. Verleysen. Are they really neighbour? A statistical analysis of the SOM algorithm output. In *Proceedings of International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 35–40, Florida, USA, 2001.

[14] K. Boehm, W. Broll, and M. Sokolewicz. Dynamic gesture recognition using neural networks; a fundamental for advanced interation construction. In *Proceedings of SPIE Conference on Electronic Imaging Science and Technology*, San Jose, California, USA, 1994.

[15] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Institute of Mathematical Statistics*, 17(3):235–255, 2002.

[16] P. Burge and J. Shawe-Taylor. Frameworks for fraud detection in mobile telecommunications networks. In *Proceedings of the 4th Annual Mobile and Personal Communications Seminar*, University of Limerick, Limerick, Ireland, 1996.

[17] P. Burge and J. Shawe-Taylor. Detecting cellular fraud using adaptive prototypes. In *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 9–13. AAAI Press, 1997.

[18] P. Burge, J. Shawe-Taylor, Y. Moreau, H. Verrelst, C. Stormann, and P. Gosset. Brutus - a hybrid detection tool. In *Proceedings of ACTS Mobile Telecommunications Summit*, Aalborg, Denmark, 1997.

[19] M. Collins. Telecommunications crime - part1. *Computers and Security*, (18):577–586, 1999.

[20] M. Collins. Telecommunications crime - part2. *Computers and Security*, (18):683–692, 1999.

[21] M. Collins. Telecommunications crime - part3. *Computers and Security*, (19):141–148, 1999.

[22] J. T. Connor, L. R. Brothers, and J. Alspector. Neural network detection of fraudulent calling card patterns. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications*, Vol. 2, pages 363–370. Laurence Erlbaum Associates, 1995.

[23] Two Crows Corporation. *Introduction to Data Mining and Knowledge Discovery).* Maryland, USA, 1999.

[24] K. C. Cox, S. G. Eick, G. J. Wills, and R. J. Brachman. Visual data mining: Recognizing telephone calling fraud. *Journal of Data mining and Knowledge Discovery*, 1(2):225–231, 1997.

[25] A. B. Davis and S. K. Goyal. Management of cellular fraud: Knowledge-based detection, classification and prevention. In *Proceedings of the 13th International Conference on Artificial Intelligence, Expert Systems and Natural Language*, Vol. 2, pages 155–164, Avignon, France, 1993.

[26] R. Drossu and Z. Obradovic. Data mining techniques for designing neural network time series predictors. In I. Cloete and J. M. Zurada, editors, *Knowledge-Based Neurocomputing*, chapter 10, pages 325–356. MIT Press, Massachusetts, USA.

[27] D. Eck, A. Graves, and J. Schmidhuber. A new approach to continuous speech recognition using LSTM recurrent neural networks. Technical report, IDSIA (Istituto Dalle Molle di Studi sullIntelligenza Artificiale) Dalle Molle Institute for Artificial Intelligence, IDSIA-14-03, 2003.

[28] D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. *Neural Networks for Signal Processing*, pages 747–756, 2002.

[29] R. T. Edwards. An overview of temporal backpropagation. Standford University, California, USA, 1991. Available at http://bach.ece.jhu.edu/ tim/research/tdnn/tdnn.ps.

[30] K. Ezawa, M. Singh, and S. Norton. Learning goal oriented bayesian networks for telecommunications risk management. In *Proceedings of the 13th International Conference on Machine Learning*, pages 139–147. Morgan Kaufmann, 1996.

[31] K. J. Ezawa. Fraud and uncollectible debt detection using a bayesian network based learning system: A rare binary outcome with mixed data structures. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 157–166. Morgan Kaufmann, 1995.

[32] K. J. Ezawa and S. W. Norton. Constructing bayesian networks to predict uncollectible telecommunications accounts. *Journal of IEEE Expert*, 11(5):45–51, 1996.

[33] T. Fawcett and F. Provost. Combining data mining and machine learning for effective user profiling. In E. Simoudis, J. Han, and U. Fayyad, editors, *Proceedings of the 13th International Conference on Machine Learning*, pages 8–13, 1996.

[34] T. Fawcett and F. Provost. Adaptive fraud detection. *Journal of Data mining and Knowledge Discovery*, 1(3):291–316, 1997.

[35] S. Field and P. Hobson. Techniques for telecommunications fraud management. In J. Alspector, R. Goodman, and T. X. Brown, editors, *Proceedings of International Workshop on Applications of Neural Networks to Telecommunications*, Vol. 3, pages 107–115, New Jersey, USA, 1997. Lawrence Erlbaum.

[36] R. J. Frank, S. P. Hunt, and N. Davey. Applications of neural networks to telecommunications systems. Department of Computer Science, University of Hertfordshire, Hatfield, Herts, United Kingdom.

[37] P. Frasconi, M. Gori, and Y. Bengio. Recurrent neural networks for adaptive temporal processing. In *Proceedings of the 6th Italian Workshop on Parallel Architectures and Neural Networks*, Italy, 1993.

[38] F. Gers. *Long Short-Term Memory in Recurrent Neural Networks*. PhD thesis, Federal Polytechnic School of Lausanne, Department of Computer Science, Lausanne, Switzerland, 2001.

[39] F. Gers and J. Schmidhuber. Learning to forget: Continual prediction with LSTM. *Neural Networks*, 12(10):2451–2471, 2000.

[40] F. A. Gers, N.N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, (3):115–143, 2002.

[41] J. Hammerton. Named entity recognition with long short-term memory. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 172–175, Edmonton, Canada, 2003.

[42] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, USA, second edition, 1999.

[43] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computations*. Addison-Wesley, California, USA, 1991.

[44] P. Hoath. Telecoms fraud, the gory details. *Computer Fraud and Security*, 20(1):10–14, 1998.

[45] S. Hochreiter, Y. Bengio, and J. Schmidhuber. Gradient flow in recurrent nets: The difficulty of learning long term-dependencies. In John F. Kolen and Stefan C. Kremer, editors, *A Field Guide to Dynamical Recurrent Networks*, chapter 14, pages 231–234. IEEE Press, New York, USA.

[46] S. Hochreiter and J. Schmidhuber. Long short-term memory. *IEEE Transactions on Neural Networks*, 9(8):1735–1780, 1997.

[47] S. Hochreiter and J. Schmidhuber. LSTM can solve hard long time lag problems. In *Proceedings of Advances in Neural Information Processing Systems*, Vol. 6, pages 473–479. Cambridge, 1997.

[48] J. Hollmen. *User Profiling and Classification for Fraud Detection in Mobile Communication Networks*. PhD thesis, Helsinki University of Technology, Department of Cognitive and Computer Science and Engineering, 2000. Espoo, Finland.

[49] P. Howard and P. Gosset. D20 - project final report and results of trials. In *ASPeCT: Advanced Security for Personal Communications Technologies*, Report AC095/VOD/W31/DS/P/20/E, 1998.

[50] H. Jaeger. The echo state approach to analysing and training recurrent neural network. Technical Report 43 pages, German National Research Center for Information Technology, GMD Report 148, 2001.

[51] M. Johnson. Cause and effect of telecoms fraud. *Telecommunication (International Edition)*, 30(12):80–84, 1996.

[52] S. Kaski, J. Kangas, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1981-1997. In *Neural Computing Surveys*, Vol. 1, pages 102–350, 1998.

[53] M. Klapper-Rybicka, N. N. Schraudolph, and J. Schmidhuber. Unsupervised learning in recurrent neural networks. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2001)*, 2001.

[54] P. Kleiweg. An extended Kohonen map. In *University of Groningen*, Available at http://www.let.rug.nl/~kleiweg/kohonen/kohonen.html#soft, 2001.

[55] T. Kohonen. The self-organizing map. In E. Sanchez-Sinencio and C. Lau, editors, *Proceedings of the IEEE*, Vol. 78, No. 9, pages 1464–1480 [1990]. IEEE Press, 1992.

[56] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, Germany, 2001.

[57] J.F. Kolen and S.C. Kremer (Eds.). *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, New York, USA, 2001.

[58] H. Kvarnstrom, E. Lundin, and E. Jonsson. Combining fraud and intrusion detection – meeting new requirements –. In *Proceedings of the 5th Nordic Workshop on Secure IT systems (NordSec2000)*, Reykjavik, Iceland, October 2000.

[59] J. Lampinen and T. Kostiainen. Self-organizing map in data analysis – notes on overfitting and overinterpretation. In *Proceedings of ESANN*, pages 239–244, Bruges, Belgium, April 2000.

[60] K. Lang and G. Hinton. The development of the time-delay neural network architecture for speech recognition. Technical report cmu-cs-88-152, Carnegie Mellon University, Department of Computer Science, Pittsburgh, Pennsylvania, 1988.

[61] D. T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Adaptive time-delay neural networks for temporal correlation and prediction. *SPIE Intelligent Robots and Computer Vision XI: Biological, Neural Net, and 3D Methods*, 1826(12):170–181, November 1992.

[62] D. T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Trajectory recognition with a time delay neural network. In *Proceedings of the International Joint Conference on Neural Networks*, Vol. 3, pages 197–202, New York, USA, 1992. IEEE.

[63] E. Lundin, H. Kvarnstrom, and E. Jonsson. Generation of high quality test data for evaluation of fraud detection systems. In *Proceedings of the 6th Nordic Workshop on Secure IT systems (NordSec2001)*, Copenhagen, Denmark, 2001.

[64] C. Mason. Cellular/pcs carriers continue to weather losses from fraud. *America's Network*, 103(2):18, 1999.

[65] R. Michalecki. Toll fraud: Multimillion-dollar telecomm problem. *Communication News*, 31(2):34, 1994.

[66] T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, USA, 1997.

[67] M. Mohammadian. *Intelligent Agents for Data Mining and Information Retrieval*. Idea Group, Pennsylvania, USA, 2004.

[68] Y. Moreau, E. Lerouge, H. Verrelst, J. Vandewalle, C. Stormann, and P. Burge. A hybrid system for fraud detection in mobile communications. In *Proceedings of European Symposium on Artificial Neural Networks (ESANN 1999)*, pages 447–454, Bruges, Belgium, April 1999.

[69] Y. Moreau, B. Preenel, P. Burge, J. Shawe-Taylor, C. Stormann, and C. Cooke. Novel techniques for fraud detection in mobile telecommunication networks. In *Proceedings of ACTS Mobile Telecommunications Summit*, Granada, Spain, 1996.

[70] Y. Moreau and J. Vandewalle. Fraud detection in mobile communications networks using supervised neural networks. In *Proceedings of SNN97, Europes Best Neural Networks Practice*. World Scientific, 1997.

[71] Y. Moreau, H. Verrelst, and J. Vandewalle. Detection of mobile phone fraud using supervised neural networks: A first prototype. In *Proceedings of International Conference on Artificial Neural Networks (ICANN97)*, pages 1065–1070, 1997.

[72] M. Mozer. Induction of multiscale temporal structure. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Proceedings of Advances in Neural Information Processing Systems*, Vol. 4, pages 275–282, 1992.

[73] M. Oja, S. Kaski, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1998-2001 addendum. In *Helsinki University of Technology, Neural Networks Research Centre*, Available at http://www.cse.ucsc.edu/NCS/VOL3/vol3_1.pdf, 2001.

[74] S.N. Pang, D. Kim, and S.Y. Bang. Fraud detection using support vector machine ensemble. In *ICORNIP2001*, Shanghai, China, 2001.

[75] T. Parker. The twists and turns of fraud. *Telephony (Supplement Issue)*, 231:18–21, 1996.

[76] M. B. Ring. Learning sequential tasks by incrementing adding higher orders. In *Proceedings of Advances in Neural Information Processing Systems*, Vol. 5, pages 115–122. Morgan Kaufmann, 1993.

[77] D.E. Rumelhart. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition : Foundations (Parallel Distributed Processing)*. MIT Press, Massachusetts, USA, 1986.

[78] D.E. Rumelhart, J.L. Mcclelland, and The PDP research group (Eds.). *Parallel Distributed Processing)*. MIT Press, Massachusetts, USA, 1986.

[79] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, May 1969.

[80] N. N. Schraudolph. *Optimization of Entropy with Neural Networks*. PhD thesis, University of California, Department of Cognitive and Computer Science, 1995. California, USA.

[81] N. N. Schraudolph and T. J. Sejnowski. Unsupervised discrimination of clustered data via optimization of binary information gain. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *In Proceedings of Advances in Neural Information Processing Systems*, number 5, pages 499–506, California, USA, 1993. Morgan Kaufmann.

[82] R. A. Shaffer. Good guys, bad guys and digital. *Forbes*, 154(4):122, 1994.

[83] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[84] D. O'Shea. Beating the bugs: Telecom fraud. *Telephony*, 232(3):24, 1997.

[85] R. Shortland and R. Scarfe. Data mining applications in BT. *Journal of BT Technology*, 12(4):17–22, 1994.

[86] G. Sun, H. Chen, and Y. Lee. Time warping invariant neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, Vol. 4, California, USA, 1993.

[87] C. Tiflin. Long short-term memory recurrent neural networks for signature verification. MSc thesis, University of the Western Cape, Department of Computer Science, Bellville, South Africa, 2003.

[88] K. G. Troitzsch. Introduction to time series analysis. University of Koblenz-Landau, Koblenz, Germany, 2001. Available at http://www.uni-koblenz.de/ kgt/SICSS/TimeSeries/mtes.pdf.

[89] A. Tsoi, D. So, and A. Sergejew. Classification of electroencephalogram using Artificial Neural Networks. *Advances in Neural Information Processing*, (6):1151–1158, 1994.

[90] J. L. Tyson. Cellular fraud swells phone bill to book size. *Christian Science Monitor*, 87(121):8, 1995.

[91] A. Ultsch and H. Siemon. Kohonen's self-organizing maps for exploratory data analysis. In *Proceedings of International Neural Networks Conference (INNC 90)*, pages 305–308, Dordrecht, Netherlands, 1990. Kluwer.

[92] A. Vahed. *Knowledge-Driven Training of Recurrent Neural Networks with Application to Time Series Modelling*. PhD thesis, University of the Western Cape, Department of Computer Science, Bellville, South Africa, 2003.

[93] P. A. Viola, N. N. Schraudolph, and T. J. Sejnowski. Emperical entropy manipulation for real-world problems. In M. C. Mozer D. S. Touretzky and M. E. Hasselmo, editors, *In Proceedings of Advances in Neural Information Processing Systems*, number 8, pages 851–857, Massachusetts, USA, 1996. MIT Press.

[94] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

[95] H. Wak and J. Zurada. Time series prediction by a neural network model based on the bi-directional computation style networks. In *Proceedings of International Conference on Neural Networks*, pages 2225–2230, 2000.

[96] H. Wakuya, R. Futami, and N. Hoshimiya. A bi-directional neural network model for generation and recognition of temporal patterns. *IEICE Transactions Inf. and Syst. (Japanese Edition)*, J77-D-II:236–243, 1994.

[97] H. Wakuya and K. Shida. Time series prediction with a neural network model based on bi-directional computation style: An analytical study and its estimation on acquired signal transformation. *IEE Transactions Japan (Japanese Edition)*, 122-C:1794–1802, 2002.

[98] H. Wakuya and J. M. Zurada. Time series prediction by a bi-directional computing architecture using future-past information integration: Application

to the sunspots number prediction. Technical report nc99-107, IEICE, (in Japanese), 2000.

[99] H. Wakuya and J. M. Zurada. Bi-directional computing architecture for time series prediction. *Neural Networks*, pages 1307–1321, May 2001.

[100] A. E. Wan. *Finite Impulse Response Neural Networks for Autoregressive Time Series Prediction*. PhD thesis, Standford University, Department of Electrical Engineering, California, USA, 1993.

[101] A. Weigend, B. Huberman, and D. Rumelhart. Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1:193–209, 1990.

[102] R. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[103] R. J. Williams and D. Zisper. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin and D. E. Rumelhart, editors, *Backpropagation: Theory, Architectures and Applications*, pages 433–486, New Jersey, USA, 1992. Lawrence Erlbaum.

[104] B. Yuhas. Toll-fraud detection. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications (IWANNT93)*, pages 239–244, 1993.