A New Approach to Securing Passwords using a Probabilistic Neural Network Based on Biometric Keystroke Dynamics

Steven Richard Shorrock

A thesis submitted for the degree of
Doctor of Philosophy
in the Faculty of Engineering
University of Newcastle upon Tyne, UK

The Department of Electrical and Electronic Engineering

January, 2003

Abstract

Passwords are a common means of identifying an individual user on a computer system. However, they are only as secure as the computer user is vigilant in keeping them confidential. This thesis presents new methods for the strengthening of password security by employing the biometric feature of keystroke dynamics. Keystroke dynamics refers to the unique rhythm generated when keys are pressed as a person types on a computer keyboard. The aim is to make the positive identification of a computer user more robust by analysing the *way* in which a password is typed and not just the content of *what* is typed.

Two new methods for implementing a keystroke dynamic system utilising neural networks are presented. The probabilistic neural network is shown to perform well and be more suited to the application than traditional backpropagation method. An improvement of 6% in the false acceptance and false rejection errors is observed along with a significant decrease in training time. A novel time sequenced method using a cascade forward neural network is demonstrated. This is a totally new approach to the subject of keystroke dynamics and is shown to be a very promising method

The problems encountered in the acquisition of keystroke dynamics which, are often ignored in other research in this area, are explored, including timing considerations and keyboard handling. The features inherent in keystroke data are explored and a statistical technique for dealing with the problem of outlier datum is implemented.

Table of Contents

Abstract ii
Table of Contents iii
List of tables and figures v
Acknowledgements vii
Publications Arising from this Research viii
Acronyms ix
Chapter 1 Introduction 1
1.1 Security of I.T. Systems 1
1.2 Identification 2
1.3 Another Password 4
1.4 Biometrics 5
1.5 Method 7
1.6 Aims of this thesis 7
1.7 Contributions of this thesis 8
1.8 Structure of thesis 9
Chapter 2 Keystroke Dynamics 10
2.1 Review of Keystroke Dynamics Publications 10
2.2 Verification vs. Identification 17
2.3 Continuous and Login Monitoring 19
2.4 Pattern Recognition 22
2.5 Evaluating Performance 22
2.6 Enrolment 24
2.7 Applications 25
Chapter 3 Data Capture 27
3.1 Accessing the Keyboard 27
3.2 Timing on a Computer 29
3.3 Experimental Environment 29
3.4 Measurable Keystroke Features 30
3.5 Variance of Behaviour Over Time 31
3.6 Data Sets 32
3.6.1 Unbiased Data 33
3.6.2 Biased Data 34
3.7 Dealing with outlier data 34
3.8 Data Pre-processing 36
3.9 Feature Extraction 38
3.10 Summary 39
Chapter 4 Statistical Methods 40
4.1 Introduction 40
4.2 Euclidean Distance 41
4.2.1 Unbiased Data 42
4.2.2 Biased Data 44
4.3 Manhattan Distance 48
4.3.1 Unbiased data 48
4.3.2 Biased data 48
4.4 Discussion 53
4.5 Summary 54
Chapter 5 Backpropagation Neural Network 55
5.1 Introduction 55
5.2 Identification and BPNN 55

5.3 Verification and BPNN 59
5.3.1 Unbiased Data 60
5.3.2 Biased Data 65
5.4 Summary 66
Chapter 6 Probabilistic Neural Network 67
6.1 Introduction 67
6.2 Verification and PNN 71
6.2.1 Unbiased Data 71
6.2.2 Biased Data 76
6.3 Summary 77
Chapter 7 Cascade Forward Neural Network 79
7.1 Introduction 79
7.2 Verification and CFNN 81
7.2.1 Unbiased Data 81
7.2.2 Biased Data 84
7.3 Comparison of Methods 87
7.4 Summary 90
Chapter 8 Conclusions 91
8.1 Further Work 92
References 94
Appendices 101
8.2 Appendix A - Keystroke Capture Program 101
8.3 Appendix B - Graphs of Statistical Results 110
8.3.1 Euclidean Unbiased Data 110
8.3.2 Euclidean Biased Data 113
8.3.3 Manhattan Unbiased 114
8.3.4 Manhattan Biased 117

List of tables and figures

FIGURE 2.1 FALSE ACCEPTANCE VS. FALSE REJECTION 23

FIGURE 2.2 OVERVIEW OF BIOMETRIC IDENTIFICATION SYSTEM 25

FIGURE 3.1 STRUCTURE OF TIME INTERVALS FOR UNBIASED DATA 34

FIGURE 3.2 TYPICAL OUTLIER DATUM 35

FIGURE 3.3 STANDARD DEVIATION OF USER 3 FOR UNBIASED DATA 39

FIGURE 4.1 AVERAGE ERROR USER ERROR CURVE UNBIASED 42

FIGURE 4.2 HIGH ERROR USER ERROR CURVE UNBIASED 43

FIGURE 4.3 LOW ERROR USER ERROR CURVE UNBIASED 44

FIGURE 4.4 OPTIMUM THRESHOLD ERROR LEVELS ACROSS USERS UNBIASED 45

FIGURE 4.6 AVERAGE ERROR USER ERROR CURVE BIASED DATA 46

FIGURE 4.7 BAD USER ERROR CURVE BIASED DATA 46

FIGURE 4.8 LOW ERROR USER ERROR CURVE BIASED DATA 47

FIGURE 4.9 OPTIMUM THRESHOLD ERROR LEVELS ACROSS USERS BIASED 47

FIGURE 4.11 AVERAGE ERROR USER ERROR CURVE UNBIASED DATA 49

FIGURE 4.12 BAD USER ERROR CURVE UNBIASED DATA 49

FIGURE 4.13 LOW ERROR USER ERROR CURVE UNBIASED DATA 50

FIGURE 4.14 OPTIMUM THRESHOLD ERROR LEVELS ACROSS USERS UNBIASED 50

FIGURE 4.16 MANHATTAN AVERAGE ERROR USER BIASED DATA 51

FIGURE 4.17 MANHATTAN LOW ERROR USER BIASED DATA 52

FIGURE 4.18 MANHATTAN HIGH ERROR USER BIASED DATA 52

FIGURE 4.19 MANHATTAN OPTIMUM THRESHOLD ERROR LEVELS ACROSS USERS FOR BIASED DATA 53

FIGURE 5.1 CORRECT CLASSIFICATION 57

FIGURE 5.2 INCORRECT CLASSIFICATION 58

FIGURE 5.3 CORRECT CLASSIFICATION WITH OTHER ACTIVATION 58

FIGURE 5.4 UNCHARACTERISTIC DATUM 59

FIGURE 5.5 RESULTS FOR UNBIASED TRAINING DATA ON BPNN 61

FIGURE 5.7 RESULTS FOR UNBIASED TEST DATA ON BPNN..... 62

FIGURE 5.9 RESULTS FOR UNBIASED TRAINING DATA FOR BPNN WITH PRE-PROCESSING. 63

FIGURE 5.10 RESULTS FOR UNBIASED TEST DATA FOR BPNN WITH PRE-PROCESSING 64

FIGURE 5.12 RESULTS FOR BIASED TEST DATA FOR BPNN 65

FIGURE 6.1 STRUCTURE OF PNN 68

FIGURE 6.2 RESULTS FOR UNBIASED TEST DATA FOR PNN USING EUCLIDEAN DISTANCE. 72

FIGURE 6.4 RESULTS FOR UNBIASED TEST DATA FOR PNN USING EUCLIDEAN DISTANCE WITH PRE-PROCESSING..... 73

FIGURE 6.6 RESULTS FOR UNBIASED TEST DATA FOR PNN USING MANHATTAN DISTANCE74

FIGURE 6.8 RESULTS FOR UNBIASED TEST DATA FOR PNN USING MANHATTAN DISTANCE WITH PRE-PROCESSING..... 75

FIGURE 6.10 RESULTS FOR BIASED TEST DATA FOR PNN USING EUCLIDEAN DISTANCE..... 76

FIGURE 6.12 RESULTS FOR BIASED TEST DATA FOR PNN USING MANHATTAN DISTANCE.. 77

FIGURE 7.1 RESULTS FOR CFNN VERIFICATION WITH UNBIASED DATA..... 82

FIGURE 7.2 EXAMPLE OF CFNN OUTPUT FOR THE CORRECT USER AND AN IMPOSTOR..... 84

FIGURE 7.4 RESULTS FOR CFNN VERIFICATION WITH BIASED DATA 85

Acknowledgements

The author would like to express his gratitude to everyone in the Department of Electrical and Electronic Engineering who have helped with the progress of this research. In particular, the encouragement and guidance received from my supervisors Dr. D. Atkinson and Dr. S. Dlay. The author would also like to thank A. Giannopoulos for his valuable insights & work with the cascade forward neural network.

Publications Arising from this Research

Dlay S, Atkinson D, Shorrock S. (2000). "Probabilistic Neural Network for Biometric Computer User Verification" *Proceedings of PREP2000, Nottingham, 11-13 April 2000.*

Dlay S, Atkinson D, Shorrock S and Giannopoulos A. (2000). "Biometric verification of computer users with probabilistic and cascade forward neural networks" *4th WSES Multi-Conference on Circuits, Systems, Communications and Computers (CSCC'2000), Vouliagmeni (Athens), Greece, July 10-15, 2000* 267-273.

Acronyms

- **ATM** - Automatic Teller Machine
- **BPNN** – Back propagation Neural Network
- **Biometric** - a measurable feature of the human body
- **CFNN** - Cascade Forward Neural Network
- **FA** - False Acceptance Error
- **FR** - False Rejection Error
- **IT** - Information Technology
- **Keystroke Dynamics** - this describes the rhythm of keypresses associated with an individual persons behaviour
- **MSE** - Mean Square Error
- **PC** - Personal Computer
- **PIN** - Personal Identification Number
- **PNN** – Probabilistic Neural Network
- **User** - this refers to an individual person in the context of '*using*' a biometric system

Chapter 1

Introduction

1.1 Security of I.T. Systems

In our increasingly connected world of information systems, security is becoming a serious issue. The vulnerability of information increases with greater accessibility. The conflict between distributed access to information and the security of that information is a problem which may never be completely solved. For many years the security of sensitive information has relied on physical barriers; papers locked in filing cabinets or even safes. The same physical barrier protection remains today on high security computer installations which are not networked. Stand alone personal computers are used with removable hard drives (often encrypted) which are locked securely in a safe when not in use. While this method of securing information is not totally secure, as it relies on the integrity of the users with access, it keeps the risk of a breach in security to a minimum by eliminating all but one path of access to the information.

Such physical barriers are not possible in a networked environment and this environment is set to grow even further in the next few years. With the coming of wireless networking, a whole new range of devices will have a presence on the internet. These will range from washing machines to third generation (3G) mobile phones and personal digital assistants (PDA's). As more sensitive information is made accessible,

such as bank accounts, personal email, addresses and schedules, there will be a heightened awareness and a greater need for security. The opportunities that trading across the internet offers will only be realised if there is a confidence amongst consumers to know who they are dealing with and conversely, a confidence amongst retailers that their customers are genuine.

Computers are becoming more secure with the addition of firewalls, strong encryption and more secure logon procedures for users, to name a few. However, with the continuing threat of viruses such as "Trojan Horses" (programs which secretly allow others access to your information) and key loggers (programs which foil encryption by intercepting keystrokes at the keyboard) it is clear there is still much to be done. Whilst there are various methods under development to increase security, they all rely on one thing - identity.

1.2 Identification

The ability for an individual to be identified with confidence is a problem in which there is a great deal of interest. Knowledge of a persons identity is generally obtained by using three distinct methods:

1. What a person has - Tokens
2. What a person knows - Knowledge
3. What a person is - Biometrics

The first method of tokens relies on what a person has. Keys are a common type of token, used to grant us access to our houses and cars. They require nothing else but the physical object itself to be of use. The disadvantage of token identification is the need to carry objects and the ease of which they can fall into the wrong hands by being lost, copied or stolen. The second method relies on the knowledge of an individual. PIN

numbers are an example of knowledge based identification. They have the advantage of not requiring a physical object to be carried around but are prone to being forgotten, guessed or discovered (often written on a piece of paper). The third method, biometrics, is probably the oldest and most common method that humans and animals use. We are continuously identifying people by recognising faces. More recently other features such as signatures, fingerprints, retina patterns and hand geometry have be used. However, automated security systems utilising biometric features are rare in comparison to the other two methods. This is mainly due to the expense of developing and installing such systems.

These methods can be used separately, but are often combined. For example, Credit Cards incorporate all three methods: tokens, knowledge and biometrics. Although only two of these methods are used at any one time. This is due to suitability of the identification method in the two scenarios in which the credit card is used. At a cash machine the card (token) must be presented, but also a PIN number remembered (knowledge). It is easy for a cash machine to check a number but much harder to analyse a signature. During a till transaction the card (token) must be produced but the person must also provide a signature (biometric). Again, it is easy for a human to analyse a signature, but it would be difficult and expensive to supply a machine to check PIN numbers in every outlet where the card is accepted.

Of all these methods of identifying an individual person, the password has become dominant in the realm of information systems. It is often only a simple password that is required to access the private key data that allows strongly encrypted data to be revealed. The password has become dominant as it is suited to the environment in which it is used. It has the advantage over other methods of not requiring any extra hardware and being easy to implement. But this proliferation of

passwords has led to some problems with the security that the password provides.

1.3 Another Password

Over the last few years there has been an explosion in the use of passwords, helped in part by the growth of the internet. Nowadays it seems every website you visit requires one. With so many passwords to remember, people are becoming careless and the security that the password once provided is slowly becoming eroded. Passwords are a knowledge based identification method and as such have always been prone to the problems associated with these systems. The three main disadvantages are:

1. Forgotten
2. Guessed
3. Discovered

The first disadvantage is that passwords are prone to being forgotten. Many people use obvious phrases for their passwords (such as their first name, dogs name, favourite band, etc...) in an attempt to make them easier to remember. Unfortunately this makes passwords easier to 'guess', increasing problems with the second disadvantage. Most system administrators attempt to force users to choose strong passwords (which are harder to guess) by setting limits on the length of password and combination of letters or numbers that it may contain. However, these stronger passwords are harder to remember and some users promptly forget them. So many users take to writing passwords down, often leaving them in an insecure place. It can be seen from this example that each of these factors depends on the others. A password that is harder to guess, may be easier to forget - requiring that it is written down, which, in turn increases the risk of it becoming discovered.

One of the biggest problems with password security is complacency. With so many passwords being required from the user, many simply use the same password for every system. If this password was to become compromised all the users information on other systems would become vulnerable.

The password will probably become outdated at some point in the future as other methods of identification are introduced (fingerprint recognition systems are already being incorporated on high end laptop computers). However, the widespread use of these alternative systems may not happen for some considerable time and the password may always be useful for some applications. With all these demands being placed on the integrity of the password, it can be seen there is scope for considerable improvement.

This thesis attempts to offer a solution to the problem by combining the knowledge of the password with a biometric feature. This is comparable to the way a credit card increases security by utilising more than one method to identify an individual.

1.4 Biometrics

The term 'Biometrics' is used to describe measurable features of the human body. A biometric feature becomes a useful tool to identify a person when it exhibits a certain degree of uniqueness to that person. Other qualities, such as stability over a period of time and ease of acquiring measurements are important. These biometric features can be split into two groups:

1. Physical Biometrics

2. Behavioural Biometrics

Physical biometrics describe features such as fingerprints, iris patterns, retina

patterns, DNA codes, faces, hand geometry, weight and height. Any measurable feature of the body which is a physical entity is assigned to this group. Features such as DNA and fingerprints have been used successfully for sometime - but problems in acquiring the measurements have so far kept their use specialised. Iris patterns hold a great deal of promise [22]. They show some of the best qualities of uniqueness, are very stable over time (protected within the eye) and can be acquired easily and cheaply with a CCD camera, although poor lighting and focus can affect acquisition.

The main disadvantage of many of these techniques, apart from the cost of hardware, is the active role which any user of the system must take. They require a person to learn how to use a new piece of equipment. There is also a time cost associated with the use of these systems. For example, on more primitive iris capture devices, the users themselves are required to ensure their iris image is in focus by moving their eye forwards or backwards until the correct focal length is obtained. This procedure will only take a small amount of time, but if it is required to be repeated frequently, or if there is a large number of people to be processed, it will become an issue.

Behavioural biometrics describe features such as signatures and voice patterns. Any measurable behaviour that a person exhibits is placed into this group. Behavioural biometric techniques can be unique to an individual and stable over time. There has been some considerable work done on signatures and voice recognition. Devices such as mobile phones, user interfaces in some cars and computer software are available and can recognise a user's voice. A lesser known biometric feature is that of typing behaviour. This biometric technique utilises the fact that individual people, when typing on a computer keyboard, do so with a rhythm that is unique to them. This rhythm can be extracted as a pattern of time intervals measured when keys are pressed. The

individuality of these typing rhythms can be exploited in the creation of a secure method for identifying individual persons.

1.5 Method

The term given to this technique of extracting a behavioural biometric from typing rhythms is *keystroke dynamics*. The task of identifying a person from their keystroke dynamics is a pattern recognition problem. Early reported methods for keystroke dynamic systems utilised statistically based pattern recognition techniques [1]. Recently, better results for classifying individuals have been obtained with neural network based methods [5]. Neural networks are becoming valuable tools for solving a wide variety of pattern recognition problems. This thesis expands on these neural network techniques by applying the probabilistic neural network and the cascade forward neural network to the problem. The probabilistic neural network was selected as a candidate for its known ability to generalise well and fit to the optimal Bayesian solution given enough data. Other advantages such as its parallel structure and quick learning were important factors, allowing users to be added and removed with minimal re-training. The cascade forward neural network was implemented with a novel interpolation method which is based on the CFNN's ability to fit to a time-sequenced series of samples.

1.6 Aims of this thesis

This thesis presents a new method of securing passwords by incorporating a biometric feature in addition to the knowledge of the password. The aim is to investigate the monitoring of a person's typing behaviour in order to increase the security of passwords on a computer system. It will be shown that by analysing the

pattern generated by the timing of keypresses on a keyboard it is possible to improve the security of passwords.

The field of keystroke dynamics is not as well developed as some other biometric techniques. This is due to other methods being favoured such as fingerprint recognition, as applications for these (e.g. policing, passport control) were historically of greater significance than securing computer access. However, as the use of computing has grown so much, interest in keystroke dynamics as a viable solution has increased. An exploration of the few methods reported for the implementation of keystroke dynamics is undertaken. The possible variations in the application of keystroke dynamics are discussed. A new approach utilising the probabilistic neural network (PNN) is compared with statistical techniques and feed-forward neural networks trained with the back propagation algorithm (BPNN). The thesis attempts to show that the use of keystroke dynamics is a viable proposition.

1.7 Contributions of this thesis

- This thesis presents a new method of keystroke dynamics utilising the probabilistic neural network (PNN).
- A comparison of the PNN with current methods available is described.
- A new method of improving false rejection performance by pre-processing keystroke data is implemented.
- A novel method using the cascade forward neural network (CFNN)
- An examination of keystroke patterns and features.

1.8 Structure of thesis

The thesis is structured as follows: Chapter 2 provides a background to the field of keystroke dynamics. The previous work in the field is covered briefly followed by an introduction to the ways in which keystroke dynamics can be implemented and the techniques applied. Chapter 3 discusses the problems associated with the acquisition of keystroke timing data and the features of that acquired data are described. Chapter 4 introduces simple statistical techniques for the implementation of keystroke dynamics and explores the use of two different vector distance measures. Chapter 5 deals with a traditional backpropagation approach to the problem and demonstrates the uniqueness of keystroke timing data with an identification based experiment. Chapter 6 details a new approach to keystroke dynamics utilising a probabilistic neural network. This is shown to perform better for biased data and train faster than the BPNN. Its other benefits such as output consistency and confidence of decision are explored. Chapter 7 is a novel approach which uses a windowed time sequenced method of interpreting the keystroke data with a cascade forward neural network. A comparison of the other method is given. Finally, Chapter 8 concludes.

Chapter 2

Keystroke Dynamics

2.1 Review of Keystroke Dynamics Publications

Keystroke Dynamics has not received the level of attention that other biometric techniques such as fingerprint and voice recognition have commanded. Consequently, there is not a great deal of published work available on the subject. The major contributions arise from a few sources with an active interest in this area. However, as there is no publicly available 'benchmark' data set, all these sources utilise their own sets of data. This makes it difficult to directly compare results from different sources. This is a fact that must be taken into consideration for the remainder of this section, which provides a summary of the current works in the field.

One of the early significant journal publications on keystroke dynamics was published in 1985 [1]. In this work the authors hypothesised that the time intervals between successive keystrokes were related to the behaviour of the brain, and that bursts of keystrokes occurred between latency for thought. For this reason they limited the number of the keystrokes to be analysed in one burst as 6. No experimentation was given as to the origin of this number and it could also be argued that such pauses for thought are a behavioural trait worthy of measurement. The method was concerned with the identification of users. Individual users provided a sample of 1400 words as a reference. The system then compiled the times between different digraphs (the latency between two characters) and produced the reference template consisting of mean times

for each digraph. Timing resolution was only 1ms and latencies over 750ms were discarded from the samples set. For identification to occur, the system required the user to provide a typing sample of 300 characters in length. This requirement for such a large number of keystrokes would make the system more disruptive to the user than a regular logon. The paper concludes that the keystroke dynamics metric is unsuitable for identification but could be suitable for verification.

A patent filed in 1986 [2] details a keystroke dynamics system where the users names and a short 'login id' are used for identification purposes. This use of a shorter textual input makes the system far more acceptable for a real application. The classifier is based on the Mahalanobis distance measure, as this gives a statistical measure of the similarity between an individual's input and the reference profile, weighted by the overall consistency. The Mahalanobis distance function, given a test vector V_0 is:

$$MD = (\overline{V} - V_0)\hat{C}^{-1}(\overline{V} - V_0) \tag{2.1}$$

where the reference vector is represented by $\overline{V}$ and the inverse of the empirical covariance matrix is given by $\hat{C}^{-1}$.

This essentially means that more consistent typists are granted less margin for error. An inconsistent typist would have a larger range over which intervals would be deemed to have originated from them. This enables the system to tighten the security for consistent typists, whilst allowing inconsistent typists a lower FR rate. This work also deals with outliers, hypothesising that they are generated by sporadic external events that distract the user, such as a door slamming or a telephone ringing. These are removed with two methods; spectral analysis and fourth moment filtering. The work quotes the probability of an unauthorised user gaining access as 0.0001, which seems to

be highly optimistic given findings in other research. It also quotes a FR error of 50% but does not provide an explanation as to how these figures are produced. The system brings in the concept of requiring multiple attempts by the correct user in order to log on – these are not evaluated individually but as a group – hence, a correct user would move steadily towards an acceptable MD level whereas an impostor would move steadily away for each access attempt. No results are provided to show that this is the case.

In 1988 one of the more prolific authors on the subject submitted a PhD dissertation [3]. The work focused on both identification and verification. The identification required the users to type the phrase 'UNIVERSITY OF MISSOURI COLUMBIA'. The verification method utilised the users names. This work brought in a 'shuffling' technique, where the user supplied two samples for classification and the best matching latencies from each sample were combined to make a new sample. Although this had the effect of requiring a longer login sequence, it proved to be useful in eliminating sporadic outlier values. The classification was implemented with a Bayesian statistical approach and the K-means minimum distance measure. Results were affected by one particularly inconsistent user, who was easily imitated by others. The timing resolution of 1ms was fairly low. The results of this work can be reviewed in [6,7,8].

In 1988 the authors of [2] followed up their work with the publication of [4]. This is built on their previous work and introduces uni, di and tetragraph analysis of the sample text. However, the method remained the same, and still required a sample of 300 characters. Applying the system to a continuously monitored environment was studies in [9]. There was no new data available for this study and results relied on a 'simulated' data set. Recognition was performed by analysing the most frequently occurring digraphs. The system took around 100 keystrokes to detect that an impostor had replaced the correct user.

A significant patent was filed in 1989 [5]. In this work, the timing resolution is given as 1ms, it introduces the possibility of using pressure sensitive keyboards to add an extra dimension to the system. The effectiveness of the system is described by the use of probability curves, where the area under the intersection of two probability distributions gives an indication of the effectiveness of the system. As this is a patent describing the method only, no results are given as to the performance of the system. The features used for the system were not explicitly stated, instead a range of possible features were described. These included the average keystroke latency, the average time to enter a common word (such as 'the'), the standard deviation of the keystroke latencies and the minimum/maximum limits of the keystroke latencies. As some of these features may be more reliable than others, a method of weighting these features was described. The variance of the feature was used to determine its consistency, and hence its reliability. Applications of the system to areas other than securing computer access are given, such as ATM and telephone keypads. The patent described how a users typing behaviour could change over time. The system was able to compensate for this by adding an element of learning to the typing template. Small changes in the users behaviour would, after a number of occurrences, be included in the template by modifying the corresponding feature.

The foundations of a later patent [10] are described in [11]. In this study the neural techniques of the ADALINE and BPNN are explored. The paper attempts to eliminate outlier data using a Kohonen clustering neural network. The timing resolution is given as 1ms. It concludes that while the ADALINE has the advantage of always finding the optimum solution, the non-linear pattern separation of the BPNN is better suited to the application. The short login strings are shown to provide enough information to implement an effective keystroke dynamics system. An improvement in

results was observed when using the Kohonen network.

A higher timing resolution of 0.1ms was introduced by [12] & [13]. An interesting outcome of the work is the degradation in performance observed when normalising the keystroke timing vectors. Normalisation is often performed on raw data in pattern recognition applications, as some neural networks require data in the range -1 to 1 (especially when using sigmoid transfer functions). However, a key feature in keystroke dynamics is the overall latency of the pattern. Normalisation preserves the pattern in the vector, but removes the information relating to the overall latency, thus reducing the differences between a slow and fast typist. Hence, poorer results were observed when normalisation was used. Three networks were implemented as classifiers; the BPNN, sum-of-products and a hybrid sum-of-products. The BPNN produced the best classification results, but required more training than the other two approaches.

The methods of [9] were improved upon by introducing a weighting on the captured digraphs [14]. Outlier values were dealt with by simply excluding all digraphs above 500ms.

A comprehensive test of common pattern recognition techniques including, cosine measure, BPNN, LVQ, RNN, HSOP, RBFN and ART-2 are described in [15]. The default keyboard handler was replaced to give accuracies of 0.1ms and the ability to capture keyup and keydown events for the first time. Better results were observed when using both the keyup and keydown event for classification. Fuzzy logic approaches were covered in more detail in [16]. Here, keystroke digraphs were weighted according to the separation of the keys on the keyboard. The digraph average times were then applied to a set of fuzzy rules and the decision made on the centre of gravity of the result. There was no reasoning given as to the derivation of the fuzzy regions.

A report on continuous monitoring, centred on the more difficult problem of identification, is detailed in [17]. However, more than 50% of the collected data in the study was discarded for being too inconsistent. Out of the 42 users in the set, 11 were eliminated from the study for inconsistency. This highlights the difficulty of dealing with inconsistent users in an identification-based system. Classification was achieved using the Euclidean distance measure, a weighted probability score method and a non-weighted probability score method. The weighted probabilistic method gave the best results with a 90% correct identification of users (at the expense of excluding many of the samples from the experiment).

A comparison of the fuzzy ART-2 method, the CPN and BPNN networks is described in [18]. The best results were achieved with the BPNN, but it took longer to train than the other methods, which achieved good results with less training overhead. This publication lead onto a significant paper detailing a comprehensive test of many different methods [19]. The results show neural network methods give a good level of performance compared to statistical techniques. The highest performing statistical method was the potential function method, closely followed by the Baye's rule classifier.

The use of real user logins as a subject for verification was covered in [20]. Three classification methods were applied to the data; a minimum distance measure, a non-linear method (utilising the standard deviation) and an inductive learning method. The data capture method used both keyup and keydown events, timed with a resolution of 0.1 ms. Results were promising, considering the low number of keystrokes in a typical login string. The average length of login string was 6.4 characters. Even with this low number of keystrokes, results were comparable to those obtained in [19].

A Java Applet written for verification using keystroke dynamics on websites is described by [21]. In this work the auto-associator neural network is compared to the k-means nearest neighbour approach. The Java Applet is used to acquire the keystroke latency times. The resolution of the timer for this method is given as 1 ms and both keyup and keydown events are utilised. However, there is no description of the exact method of obtaining the keystroke timing data. In a platform independent, high-level language such as Java, accessing the keyboard at a low level in order to obtain accurate readings is necessary. A concern that the collected samples would be corrupted by the underlying operating system buffering the keystrokes is not dealt with by the publication. The results obtained are very promising with an error rate quoted as 1% for the auto-associator network. It is unclear whether this relates to the FAR or FRR or both. This low error rate was achieved through considerable manipulation of the data set. Of the samples collected, up to 50% were discarded for each user. The system required a large training set, up to 325 samples from each user, far more than could be expected of users in a 'real' system.

The classification methods described by previous works have shown neural networks to be strong candidates for this application. Statistical methods have faired less well, with the exception of probabilistic based approaches, such as Baye's rule [19]. It was therefore conceived that a combination of these methods might produce better results. The problem of eliminating outlier data in the typing samples has been highlighted in previous work [19]. Consequently, a method of pre-processing the sample data set is implemented in this work.

The practise of eliminating whole samples or even individuals from the data set to improve results will not be applied in this work. Some precious work report results as a single combined figure for error. This does not give the whole picture. Therefore, all results in this work shall be given as FAR/FRR error pairs for each particular case.

It is helpful when looking at results to visualise the spread of the FAR/FRR errors for each case. In this work, results are displayed for the statistical classifiers in a new form (Figure 4.4). This enables the reader to see the spread of the errors that make up the error for the system as a whole. The FAR/FRR error is shown in the y-axis, with the x-axis referring to the threshold level employed in the system.

The previous works have shown that the problem of identification is not easily solved with the use of keystroke dynamics. It is probable that this is inherent in the nature of the biometric, as it is more prone to change over time and similarity across individuals than physical biometrics such as fingerprints and DNA. This thesis will therefore be concerned with the problem of verification.

2.2 Verification vs. Identification

Biometric security systems, including keystroke dynamics, operate in one of two modes:

1. Identification
2. Verification

Identification is the ultimate test of a biometric. The identity of a person must be determined from their given biometric. The system must search a database of known users and select the person to which the sample belongs. It must also be able to determine if the person is a stranger to the system; in this case it should fail to find a match amongst the set of known users. This is a task that we perform everyday when we recognise somebody's face. We take the biometric of a face and match it to a name that we have remembered. We can also tell strangers apart from known individuals with ease. However, the fact that we, as humans, find this an easy task, does not mean it is a

trivial problem. In order for a system to work as a recogniser, it must be able to confidently find a persons identity amongst a user base that may contain thousands, if not millions of users. Even the human brain would not be able to cope with such an amount of information. In fact, whilst we are good at recognising whether a face belongs to a stranger, we are not so good at remembering names. Some studies have suggested we have difficulty remembering more than one hundred names. This gives some idea of the enormous diversity that a biometric must contain if it is to be truly unique to an individual.

It takes a special type of biometric to be suitable for identification purposes. It must have the properties of being highly individual to a specific person and stable over a period of time. For example, if the chances of someone having the same biometric as another individual are 10 million to 1, in a country such as the UK a search through the population of some 60 million may result in the person being identified as six individuals. Biometrics proved suitable for identification are the fingerprint, DNA code, retina pattern and the iris pattern. The iris pattern is proving to be a very promising method, details can be found in [22].

The task of verification is less demanding on the uniqueness of a biometric. In this role we are seeking to check that an individual is who they claim to be. Biometrics which are occurring similarly in only 1 in 1000 of population may be fairly effective for verification (but useless for identification). There is a reference in [5] to a claim that keystroke dynamics are as unique as the fingerprint, however, this is not backed up with any evidence. Indeed, to prove the uniqueness of keystroke dynamics would be an extremely difficult task, as typing data would need to be collected from a vast number of persons. Whilst this claim may not be proven, it does suggest keystroke dynamics may exhibit a fairly strong degree of uniqueness.

The main focus of the work in this thesis will be orientated towards verification. Identification will be covered briefly for a closed environment of a limited number of users.

2.3 Continuous and Login Monitoring

There are two approaches for improving security on computer systems using keystroke dynamics:

1. Continuous Monitoring
2. Login Monitoring

Both of these methods can operate in either identification or verification mode. Continuous monitoring involves a continual evaluation of an individual's typing behaviour. In this scenario the text that the user is keying in is unknown to the system. Therefore, the information regarding the behaviour of the typist must be built from features which are inherent in their general typing behaviour. These features must be consistent no matter what is typed. This approach can be used for continuous monitoring of the typist during the duration of their session on a PC. The aim is to constantly monitor the use of the PC for unauthorised access. It is of most use when a PC is being used for word processing or other application that requires the regular use of the keyboard. A level of security is provided such that, if the authorised user left the PC unattended, an unauthorised user would be detected whilst using the keyboard. The keystroke monitoring system would not recognise the typing rhythm of the unauthorised user and subsequently deny access to the system. The difficulty with monitoring keystrokes continuously is developing a method of taking the large amount of data acquired and finding consistent features. This has been done in some methods by simply

looking for commonly occurring words (such as 'the', 'and', 'where' and 'there') and comparing the time patterns for these words with corresponding data of known patterns for the user. Other methods measure the time intervals between pairs of characters, for instance, the time between 'A' and 'S' or the time between 'T' and 'H'. These are then compiled and compared with a reference profile of previously generated data for the user. These time delays are often compiled into a digraph chart [1]. A digraph is the term given to a time interval between two characters. Although these digraphs reduce the amount of data which needs to be stored to represent a user's typing rhythm profile when compared to the method of commonly used words, the amounts of data associated with generating templates for both these methods are substantial. The data storage requirements are not the only disadvantages with this approach. Continuously analysing the timing data from a user will put the PC's processor under more load than usual. This may affect the performance of the computer system. Also, the continuous monitoring system only secures access via the keyboard, allowing an impostor to navigate the computer via the mouse or other input device. This degrades the effectiveness of the continuous monitoring approach on reinforcing security. It would seem only to have use where a PC is without an input device other than the keyboard, or on a system where the main threat to security can only be accomplished on a keyboard. Such a scenario is uncommon these days with the vast majority of PCs having a mouse. These factors may limit the potential uses of continuous keystroke monitoring.

Login monitoring ascertains the user's authenticity only once, during the logon process. It builds on the traditional login procedure found on many computer networks by asking the user to type a known username and password. As the user types, it monitors their typing style and compares it with a known profile. Security is enhanced with this method, as an extra barrier to access is encountered, even if an impostor has obtained knowledge of another users password, as the system requires the impostor to

type the username and password with a similar rhythm to the authorised user. This makes it a very difficult task for an impostor, but a familiar procedure for most computer users. It can even be performed without the knowledge of the user. Hence, no user training is required to use this system. Some physical biometric techniques, such as fingerprint recognition, require the use of new hardware and procedures. Login monitoring has the advantage of an easier implementation than the continuous monitoring system, with a smaller requirement of PC resources. The computer's processor is only used once at the logon, therefore, it will not affect the subsequent performance of the PC. The storage requirements for the reference profile are also much reduced. Once the user has been granted access to the system there are no further checks on authenticity. However, depending on the level of security required, authentication could be demanded from the user for performing certain critical tasks (such as file copy/deletion) or, after a certain period of inactivity. This would help protect the system should a user leave the PC unattended while logged in.

The techniques for continuously monitoring a users typing rhythm can also be applied to the logon procedure. Here, instead of just providing a username and password, the user must type a phrase, randomly generated by the PC and previously unseen by the user. The users typing style is then checked against a profile and access granted or denied. Some of the disadvantages with the continuous system apply here too, such as the large memory requirement of the profile. Also, the delay encountered by the user whilst typing this phrase may be unacceptable.

The focus of this thesis will be login monitoring. The use of keystroke dynamics in this guise has the potential to apply a new, higher level of security to the vast majority of PC systems.

2.4 Pattern Recognition

Pattern recognition (or discrimination) is an attempt to 'guess' the nature of a previously unseen observation [24]. The nature of these observations may be any number of differing classes, which may or may not be known. If the different classes are known, the nature of the observation may be derived from previously seen examples. If the classes are unknown, grouping of observations may be inferred from clustering. Each observation is a vector of numerical data representing features (such as pixels in a digitised image). In keystroke dynamics the observation is often a vector of time intervals between keystrokes, obtained from a person typing on a computer keyboard. For a keystroke dynamic problem, the nature of the observation is generally known and could be either an authorised user or an impostor.

2.5 Evaluating Performance

Login monitoring with verification requires that a yes/no decision is made on a pattern presented to a classifier [25]. This is a common scenario within the field of pattern recognition. In all pattern recognition tasks of this nature there are four possible outcomes - either the pattern is the target or not and the decision made is either correct or not. The four possible outcomes in keystroke dynamics are:

1. FA (False acceptance) - this is the case where an impostor is wrongly accepted by the classifier as the authorised user.

2. FR (False rejection) - when the correct user is wrongly rejected by the classifier as an impostor.

3. CA (Correct acceptance) - the authorised user is correctly accepted by the classifier.

4. CR (Correct rejection) - an impostor is correctly rejected by the classifier.

It is clear that a good classifier seeks to minimise the errors FA & FR (known as Type I and Type II errors respectively) and maximise the correct decisions of CA & CR. Often the performance of biometric recognition systems is described by these measures. However, it is not these values alone which are important, but the relationship between them. It is possible to obtain a 0% false rejection error by simply accepting every access attempt. It is therefore necessary to quote figures for all these cases when describing the performance of a biometric security system. To give a clearer picture a graph can be drawn which shows the relationship between false acceptance and false rejection.

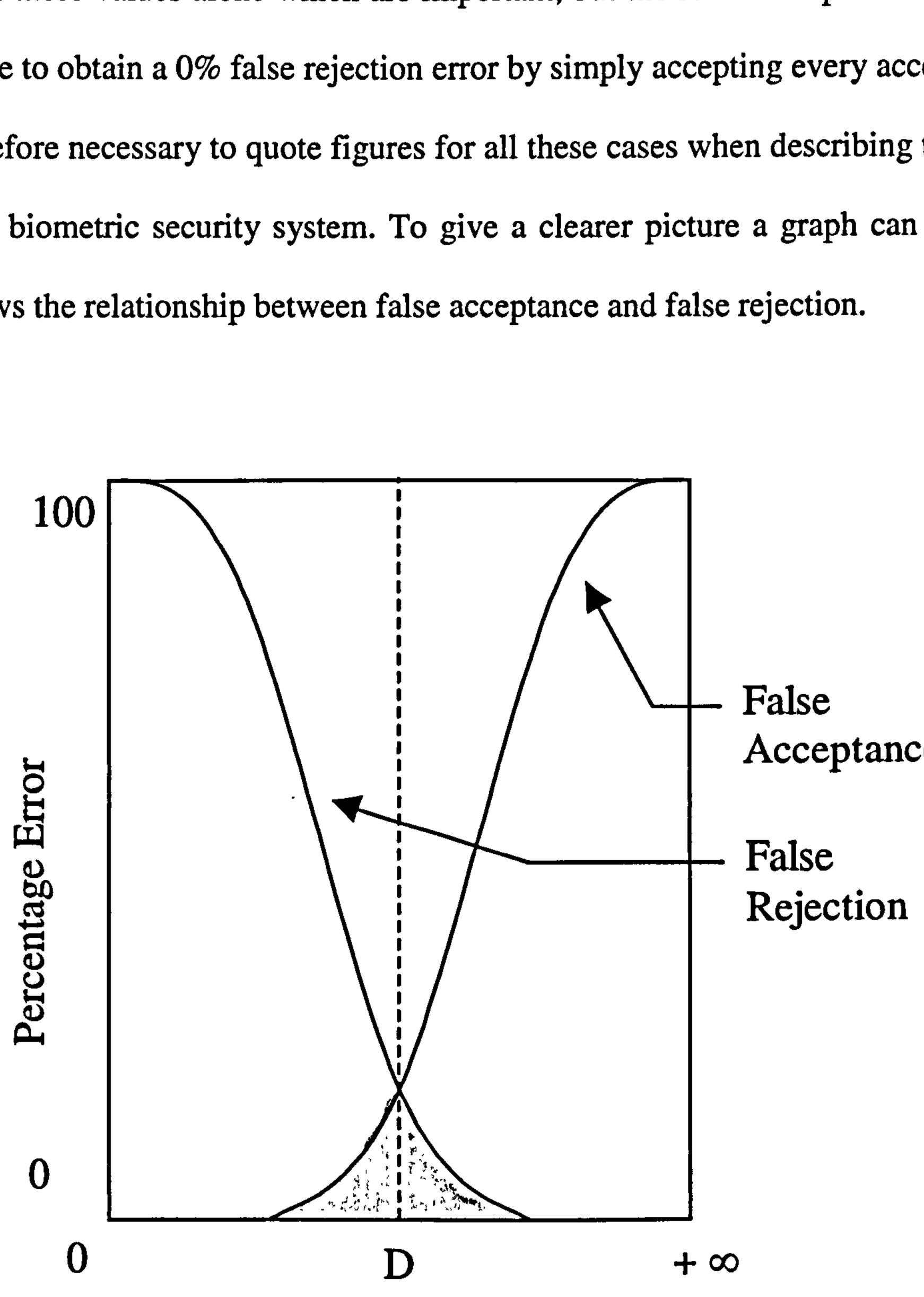


Figure 2.1 False Acceptance vs. False Rejection

In Figure 2.1 an example of an FA vs. FR graph is shown. It displays the two errors imperfectly separated by some discriminating measure. The line 'D' represents a decision threshold. All values less than 'D' are deemed to have originated from the correct users and all values greater from an impostor. Altering the threshold level 'D', alters the ratio of FA to FR. By moving D to the left of the centre, FA is reduced but FR is increased – at the limit, FA is reduced to zero simply by rejecting every sample. Correspondingly the FR error in this case is 100%.

The crossover point of these two graphs, marked 'D', can be used to give an idea of the overall performance of the system. A lower error at the crossover point 'D' indicates an improvement in performance. The shaded area represents the region of error, where misclassifications will occur. The smaller the shaded area, the better the performance. In a totally accurate system, these two areas would be perfectly separated.

2.6 Enrolment

Obtaining biometric features reliably is the first, crucial step, in any biometric system (Figure 2.2). Enrolment is the term given to the process of acquiring such features [22]. The degree of difficulty encountered during enrolment varies according to the individual properties of the biometric concerned. An example of a biometric with a difficult enrolment is iris recognition. The acquisition of iris patterns relies on capturing an image of the iris. The equipment involved must acquire an image of the iris which is properly focused [22]. During acquisition, the subject may move resulting in an image which is blurred and out of focus. If this image is of such poor quality that the features of the iris cannot be ascertained, then the enrolment of the biometric observation is deemed to have failed. Estimating the likely rate of a successful enrolment can be a difficult task.

Fortunately enrolment is not a big problem for keystroke dynamics. The patterns unique to an individual are inherent in the timings of keystrokes on the computer's keyboard. Any physical problems associated with acquiring keystrokes have been addressed by keyboard manufacturers long ago. The only perceived problems with the enrolment of keystroke dynamics is the possible corruption of the timing intervals by software. As this could be caused by the operating system or any number of applications running on a PC it may be difficult to detect if it occurs.

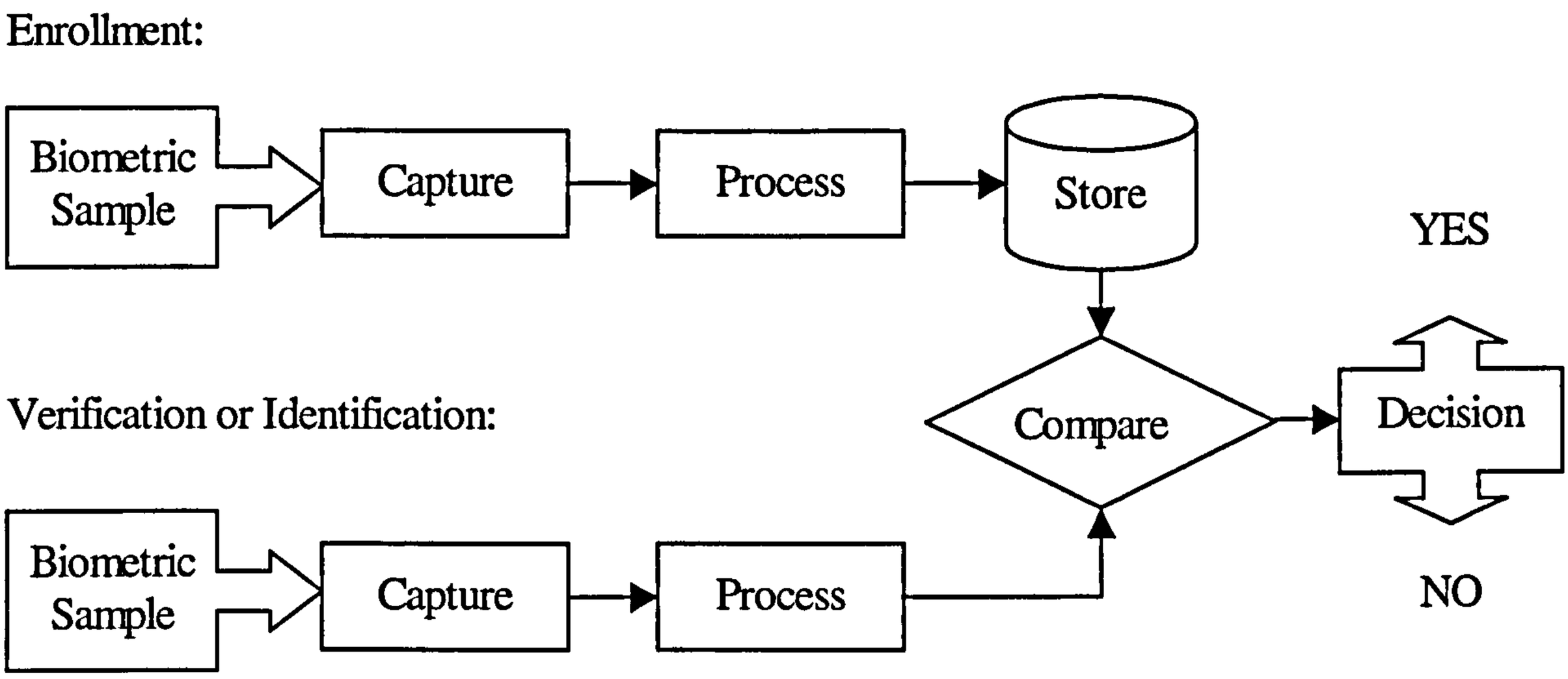


Figure 2.2 Overview of Biometric Identification System

2.7 Applications

The primary application for keystroke dynamics is seen as the personal computer. The major advantage of this technique over any physical biometric system is the speed with which it may be deployed across a network of computers. The costs associated are minimal and confined to the cost of the software. However, the application is not limited to the reinforcement of passwords on a personal computer, other uses of the system can be envisaged across a number of areas.

By monitoring the timing intervals of keypresses on a telephone keypad it is possible to extract a keystroke dynamic pattern of behaviour. This would require no modifications to any existing telephone at the acquiring end. The only requirement would be a system at the receiving end to monitor and time the DTMF tones produced by the acquiring phone. With such a system it may be possible for details such as credit card information to be keyed in to the phone and verified by the actual credit card number keystroke dynamics or by a PIN number keystroke dynamics supplied with it.

ATM cash points are a possible area in which keystroke dynamics could increase security. In a similar manner to the telephone keypad system, the timing intervals could be acquired as a person types their PIN number on the cash point keypad. The need for an extremely low FR rate would be important here, as banks would not wish to inconvenience legitimate customers. The system could be used in conjunction with a habit tracking system. A user may not be refused access if they fail the keystroke dynamic test, but the transaction they wish to make could then be highlighted for scrutiny. Any transaction which is abnormal to the customers recorded spending habits could then be blocked.

Chapter 3

Data Capture

3.1 Accessing the Keyboard

An essential part of any keystroke dynamics system is the acquisition of keystroke data from the keyboard. The information regarding the timing of keystrokes must be handled by the computer operating system in a manner which does not affect the accuracy of those measurements at the keyboard. A program or application running on a PC and performing keystroke dynamics must be able to access the keyboard at a low level. In a sophisticated high-level operating system such as Windows™, application access to peripheral devices such as the keyboard is controlled through an interface. This enables software developers to create applications without the need for their programs to handle the keyboard directly. Whilst this is advantageous to most applications, there are certain properties regarding the behaviour of this interface which are undesirable when obtaining keystroke dynamics.

A multitasking operating system must share resources out amongst the various applications running on the platform. In order to allocate them effectively, priorities are assigned to every function. Housekeeping functions of the operating system such as hard drive and memory access are given a high priority. User inputs are given a low priority and often the user will have to wait while the system is busy. The keyboard has a buffer which allows the operating system to temporarily suspend the keyboard

handling, but allows the user to continue typing. When a high priority task needs to be run, the inputs from the keyboard are passed to the keyboard buffer. This can store a limited number of keystrokes until the high priority task is completed. An example of this can be seen in a word processor which has an 'auto save' function. If a user is typing when a save needs to be performed, the characters momentarily freeze on the screen whilst the save is performed. When the save is completed the characters that have been typed whilst the screen was frozen, rapidly appear. This is the action of characters being released from the keyboard buffer and transferred to the application.

Transferring keyboard strokes to applications in Windows™ is accomplished by setting the value of a "windows message" to the character code of the depressed keys. When a key is pressed on the keyboard, the Windows™ keyboard handler sends a 'KEYDOWN' message to the active application with the code relating to the key. When a key is released, a 'KEYUP' message is sent to the active application along with the key code. If Windows™ interrupts the keyboard messages whilst it performs a higher priority task, it places the keystrokes in a buffer. Once the high priority task has finished, the buffer is flushed and all the stored keystrokes are sent to the application. Unfortunately, during this process the information about the timing of those keystrokes is lost. Therefore, in a keystroke dynamics system, another means of interfacing with the keyboard, which bypasses the default windows keyboard handler, is required.

To bypass the original keyboard handler, a new one needs to be created which can ensure that the timing data acquired at the keyboard is not disturbed by the operating system. This is possible to implement on a Windows™ PC and is actually widely used by games programmers. They also need such a low-level access to the keyboard, as the timing of keypresses is critical to the performance of a computer game. As the problem of timely access to the keyboard is overcome, the next problem is the measurement of these small time intervals that the keyboard handler generates.

3.2 Timing on a Computer

A PC performing keystroke dynamics is also required to provide an accurate means of measuring the time intervals between keystrokes. Typically, time intervals between keypresses are in the order of milliseconds, with the most experienced typists creating the shortest intervals. Every PC has a system clock on the motherboard. Most time dependant functions depend on this device. However, this BIOS clock was not designed to provide accurate timing. Although it has an accuracy of 0.001 seconds, in actual fact this timer is only updated 18.2 times per second. This makes the resolution interval only 54.925ms. As some keystrokes can be shorter in duration than this, the BIOS clock is not suitable for use as a timing device for keystroke dynamics.

Another approach is to use the computer processor unit (CPU) as a timing device. As the CPU processes many hundreds of thousands of operations per second, it provides excellent resolution as a timing device. CPU clock speeds vary between PCs, therefore, in order to maintain consistency across PCs, the CPU clock speed must be determined so that the timing interval can be converted to a standard unit. An easy way to accomplish this is to measure the number of processor clock cycles performed over a time interval with the BIOS clock. This time interval must be of a length that reduces the effect of the 18.2 samples per second resolution to negligible levels. With such calibration the time interval for a single clock cycle can be deduced.

3.3 Experimental Environment

Although it would be possible to implement accurate keystroke capture on a PC by replacing the keyboard handler of the OS with custom software, for the purposes of this research, a method of collecting keystroke dynamic data was sought which would

not be affected by any of the problems mentioned above. This would ensure that the data collected was as pure as possible in order that the results obtained were totally reliable. In order to satisfy this requirement, a system for capturing the keystroke data with a single board computer was implemented. The software to acquire the keystroke data was written entirely in assembly language. With no operating system on the device and the complete control that low-level programming enables; a stable, clean environment was created for capturing keystroke dynamics. A keyboard was the only input device connected to the system, with a monitor connected for the output. This set-up contained no hard disk, so data was transferred via serial cable onto a host PC for analysis. The program listing can be found in Appendix A - Keystroke Capture Program. The processor for the data capture was the Motorola™ MC68000. All experiments were programmed and conducted with the Matlab™ simulation package (Version 5.2 release R11). The metric used throughout the thesis to quantify the timing gaps between keystrokes is processor clock cycles, with each clock cycle representing a period of 0.125us.

3.4 Measurable Keystroke Features

The number of measurements which may be quantified from a keyboard as a person types, varies according to the type of keyboard. Possible measurements that may be useful include keystroke pressure and key acceleration. However, obtaining these measurements would require the use of a special keyboard. Such keyboards would impose a hardware cost on the provision of a keystroke recognition system. For this reason, the collection of data was implemented on a standard keyboard, where three key measurements can be captured:

1. Keydown - this occurs when a key is depressed.

2. Keyup - this occurs when a depressed key is released.

3. Keycode - the unique code which identifies which key was depressed/released

From these three measurements, two metrics can be obtained:

1. Flight time - the time between a keyup and a keydown.

2. Hold time - the time between a keydown and a keyup.

Not all methods use both of these metrics, some use just the hold or flight time. Others utilise the combined metric obtained from just adding the hold and flight times together. This gives the time interval from keydown to keydown (or keyup to keyup). [18] shows that the hold time is a better metric than the flight time for identifying individuals. Also, it is shown that using both the metrics as separate entities yields a better result than the combined metric. For example, the potential function method reported in [18] has FR and FA errors of 4.7% and 2.2% respectively for the case where just the flight time is used. The FA and FR reduce to 1.9% and 0.7% when both hold and flight time are used.

The single board computer system used in this study to collect the raw keystroke data had the ability to record a 'key-down' event only. Therefore, all the experimental data utilised the combined hold and flight time. The use of a consistent data set provided a good base from which to draw comparisons between different classification methods.

3.5 Variance of Behaviour Over Time

An unknown quantity with behavioural biometrics is the possibility that the behavioural trait will change over a period of time. There has been no evidence of this occurring within the current published research on keystroke dynamics. However, it is perhaps intuitive that it will be a problem among some users. This is also a problem with physical biometric features, for example, a cut on an individuals fingertip may

render their fingerprint unrecognisable.

Keystroke behavioural traits differ from some physical metrics as they are less likely to experience sudden, acute changes (such as the scar on a fingertip). Changes in behaviour tend to occur slowly and over a period of time. In this way it is possible to update the classifier so that it 'learns' and adapts to subtle changes in the users behaviour. This is a problem that will only require solving once a classification technique for a keystroke recognition system has been perfected. The scope of this thesis is primarily concerned with classification techniques. Therefore, changes in behaviour will not be covered within the context of this thesis.

3.6 Data Sets

When a person performs a task with a high frequency they develop a consistent behaviour. The same holds true in keystroke dynamics [1]. Two types of data were acquired in order to explore the differences in results when users type personally familiar phrases and general phrases. These two data sets are labelled biased and unbiased respectively. The combination of these two data sets provide a solid base from which to design and test the system. The unbiased data set was chosen in order to test the classifier's discrimination abilities and the biased to indicate likely performance in a real application. Each data set is divided into training and test samples. The number of training samples must be a realistic amount that a person would be willing to give in order to set-up a keystroke dynamic system. A figure of ten samples was chosen as this seemed acceptable to most users and is similar to amounts used in other experiments, notably [5]. The number of users in the unbiased data set was twenty one and the number in the biased set five. All the users were familiar with a computer keyboard and they also had varying degrees of typing ability. No mistakes were tolerated during the

capture of keystrokes. If a user made a typing error the sample was discarded and another attempt initiated.

3.6.1 Unbiased Data

The unbiased data set is used to test the classifiers ability to discriminate users based solely on their typing rhythm. As all users type the same universally known phrase, any variances and correlations emerging from the data will be purely attributed to their individual typing rhythms. Users have no prior knowledge or experience in typing this phrase, therefore, it is labelled unbiased. Some studies have used long strings of text in order to extract the keystroke dynamics. In this work the logon procedure is the focus, therefore, a phrase similar in length to the typical logon was used. Each user was requested to type the word 'username' followed by a carriage return, the word 'password' and a final carriage return. Timing was initiated as soon as the username prompt appeared and ended with the final carriage return. This resulted in a timing vector of 18 time intervals for each sample (Figure 3.1). Each of the twenty one users provided fifteen samples. These samples were later split into training and test sets, with ten samples in the training set and five in the test set.

Whilst this provides a good test for any pattern recognition method, the results it produces will not be representative of the final application. In a keystroke dynamic logon system, users will type their own particular username in order to logon. This username will become very familiar to the correct user, but unfamiliar to an impostor. Therefore, to indicate the likely performance to be expected from the classifier in the real application, the biased data set was created.

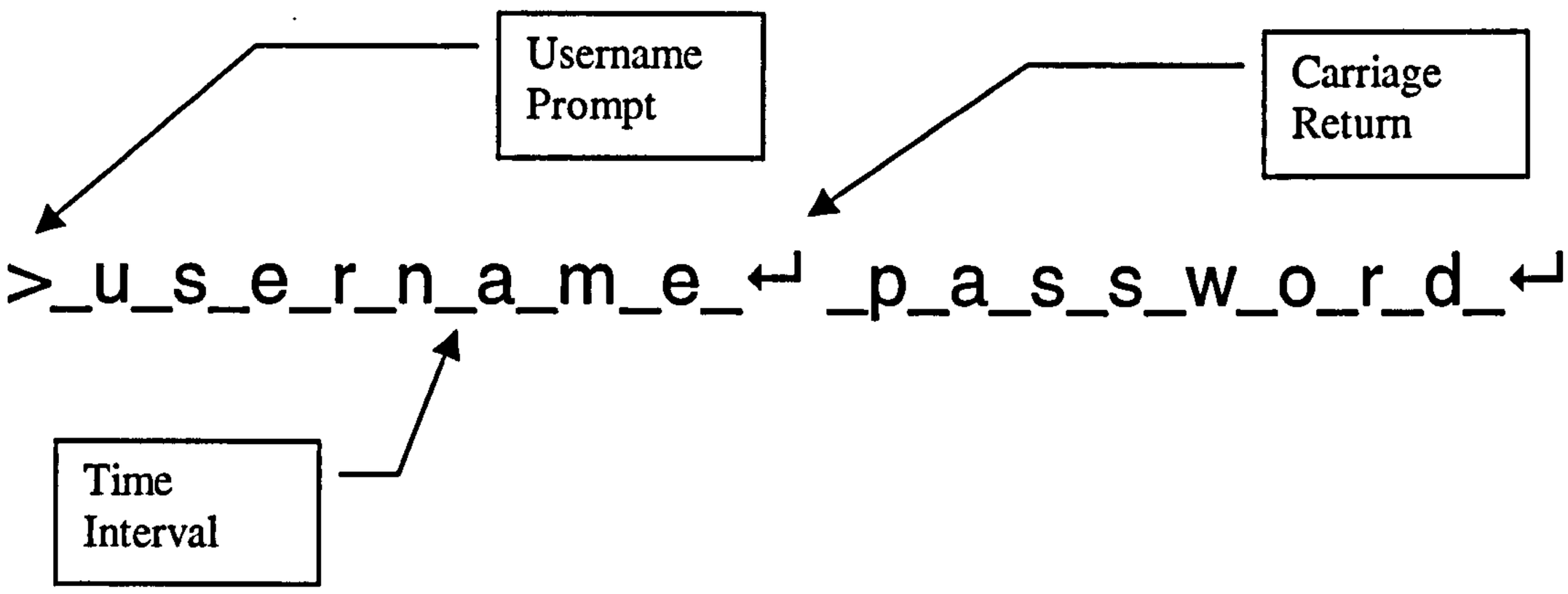


Figure 3.1 Structure of time intervals for unbiased data

3.6.2 Biased Data

The biased data provides a more realistic test for a keystroke dynamic system. In the biased data set, users were asked to type a phrase which is particularly familiar to them. To emulate a login procedure, each user was asked to type their names in full. The timing started with the prompt for their name, followed by a carriage return, the word 'password' and a final carriage return. Each user was then requested to provide attempts at typing the other user's names. This provided impostor samples for every user. The number of timing intervals generated in each sample for the extraction of the keystroke features varied with the length of the users name. The lowest number of timing intervals was 19 and the highest 25. Each of the five users provided twenty samples of their own name and ten 'impostor' samples for the other four names. These samples were later split into training and test sets, with ten samples in the training set and ten in the test set.

3.7 Dealing with outlier data

Hesitations that occur during typing, often do so at random intervals [2]. These hesitations can have an effect on the performance of the classifier. This is particularly

important in the training of neural networks. In order for a neural network to learn a pattern correctly, it relies on the data within the samples being representative of the class. Any data which are not consistent with a pattern will cause a network to learn an erroneous pattern. It is therefore important that the data being supplied to the neural network are as 'clean' as possible.

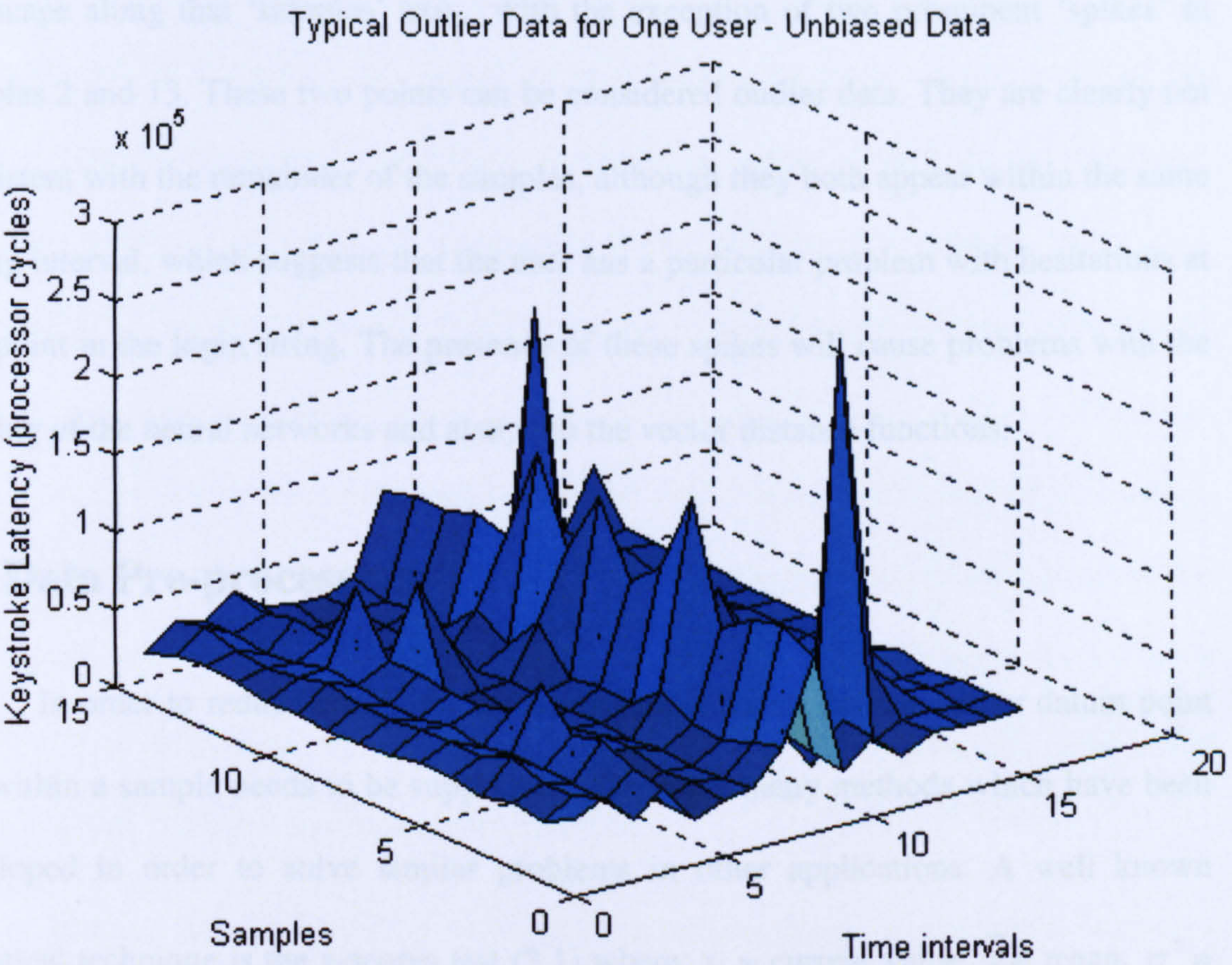


Figure 3.2 Typical Outlier Datum

An illustration of how spurious outlier data can affect the sample space can be seen in Figure 3.2. The three dimensional plot shows the typing samples for one user from the unbiased data set. Sixteen time intervals in each sample can be seen on the axis labelled 'time intervals' (the first and last samples are removed due to high variance). Fifteen samples are then arranged along the axis labelled 'samples'. The time for a

keystroke is represented on the 'keystroke latency' axis - note the unit of measurement here is processor clock cycles. The large values ($x10^5$) gives some idea of the high resolution obtained in this experimental environment.

A visual comparison can be made with the keystroke patterns along the 'samples' axis. Each sample is of a similar pattern, creating an almost uniform landscape along that 'samples' axis, with the exception of two prominent 'spikes' in samples 2 and 13. These two points can be considered outlier data. They are clearly not consistent with the remainder of the samples, although they both appear within the same typing interval, which suggests that the user has a particular problem with hesitations at that point in the login string. The presence of these spikes will cause problems with the training of the neural networks and also with the vector distance functions.

3.8 Data Pre-processing

In order to reduce errors, the effect on classification that an outlier datum point has within a sample needs to be suppressed. There are many methods which have been developed in order to solve similar problems in other applications. A well known statistical technique is the z-scores test (3.1) where: x_i = current value, $\bar{x}$ = mean, σ^2 = standard deviation.

$$z_i = \frac{x_i - \bar{x}}{\sigma^2} \tag{3.1}$$

When a user enrols on a keystroke dynamics system they provide a set of samples as a reference for the classifier. Those samples are used in order to create a reference template. From this reference template the mean value for each observed time interval is calculated. The z-score for each time interval is calculated by subtracting the mean value of that time interval from the current value, then dividing it by the standard

deviation for that interval. So in addition to the reference template mean, we also need the standard deviation for each keystroke in the vector. This effectively doubles the amount of data needing to be stored for each user in a simple statistical classifier. However, the z-score allows the identification of any points in the training and test sets which fall outside a certain deviation from the reference template. Once these outlier points have been identified, their effects need to be minimised. One method for achieving this is to replace the outlier value with a value more representative of the sample. This can be achieved by simply replacing the errant value with the equivalent from the template. However, an impostor's typing rhythm may produce many time intervals which are unrepresentative of the correct user's response. These would then be wrongly identified as outlier points. In order to reduce this effect a limit needs to be placed on the number of outlier points that can be replaced within any given sample. Also, by replacing the errant value with one from the template, the distance between the vectors would be reduced by an amount which would not normally occur as it is very rare that a time interval will be an exact match to that of the template. This is the method which is used in the subsequent chapters.

Another method would to be to apply a scaling factor to the outlier value to reduce its effects. This would enable the value to be reduced to a level which is closer to, but not the same as the value in the template. It must be noted that this will also need a limit applying to stop every single value being brought more towards that of the correct user. Employing these techniques has the effect of reducing the false rejection error, but their nature of altering the uniqueness of impostor data will increase the false acceptance error. The degree to which the false acceptance and false rejection errors are affected depends on the number of outlier points which are altered.

3.9 Feature Extraction

The first step in most pattern recognition tasks is to extract some quantifiable features from the given data. The features extracted directly influence the performance of the classifier. Ideally, the set of features extracted should provide the classifier with easily (linearly) separable clusters within the pattern space. Often, improvements in the performance of pattern recognition systems are due to improvements in the extraction of features and not the design of a better classifier. Extracting the right features from a data set can reduce the computational complexity of the problem. In applications where the observations acquired contain significant amounts of data, such as image recognition, feature extraction is essential to obtain the important artefacts from the mass of data.

In keystroke dynamics, especially applied to login monitoring, the amount of data acquired is low compared with other pattern recognition tasks such as image recognition. In this case, the task of any feature extraction will not primarily be feature reduction, but the elimination of erroneous data from the sample. When looking at the standard deviation of a typical user's typing behaviour (Figure 3.3), it is clear to see that some of the keystroke intervals occur with more consistency than others. One possible method is to apply more weighting on these intervals, so that the classifier places more importance on the intervals which have a low variance. However, if any outlier datum occurs on these points, it will have a greatly exaggerated effect due to the higher weighting. This weighted method shows promise, however, an exploration of this was beyond the scope of this thesis.

It is clear to see from Figure 3.3 that the first sample has a very large variance. This sample is the time interval between the prompt appearing on the screen and the user pressing the first key (u). As can be expected, the variance is large as the user takes varying levels of time to start typing based on their reaction to the prompt and the

distance that their finger needs to move to start typing on the keyboard. This feature is inconsistent and displays a high variance in all users, so it was removed from the data of every user before classification.

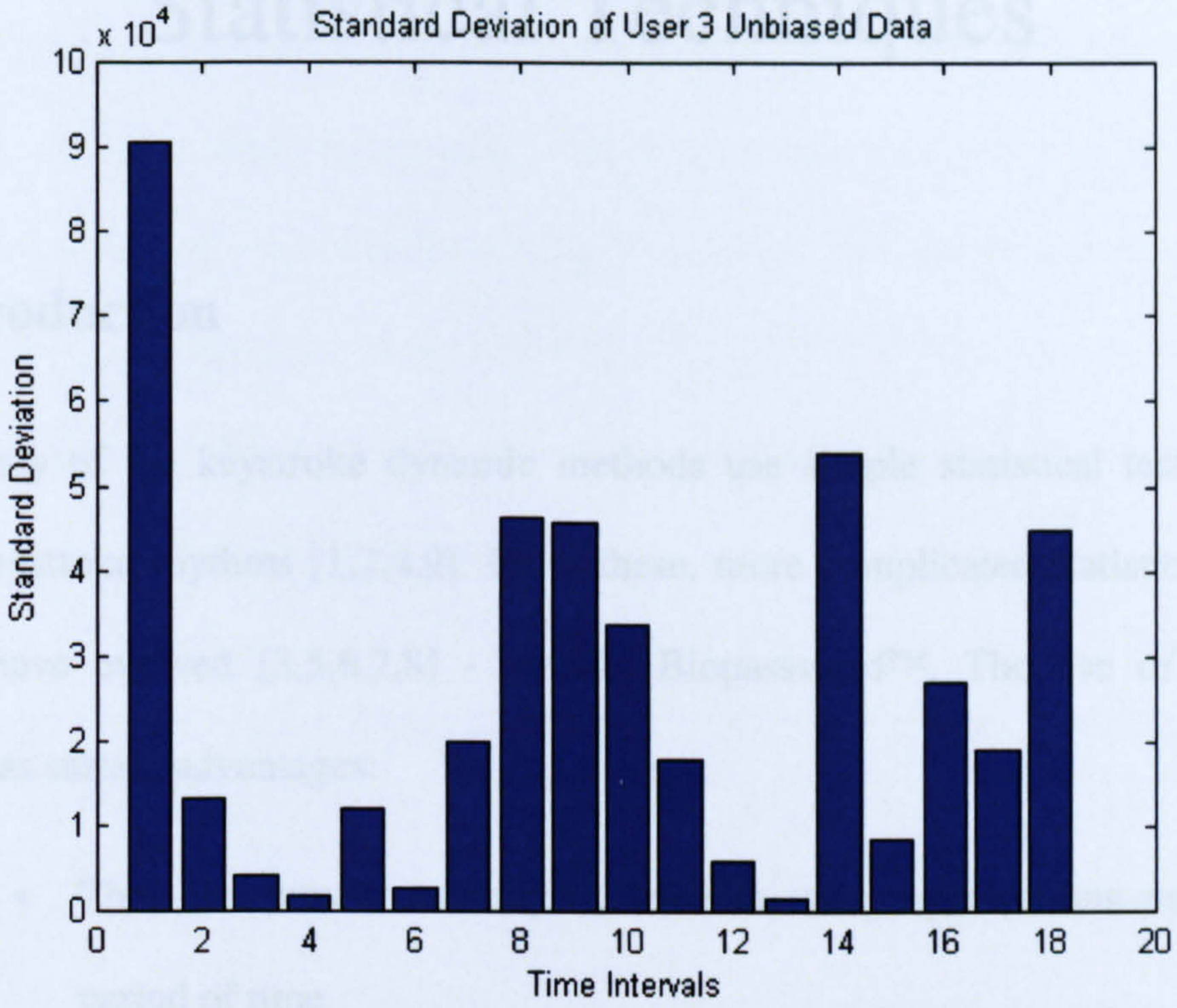


Figure 3.3 Standard Deviation of User 3 for Unbiased Data

3.10 Summary

This chapter has introduced the problems associated with the accurate capturing of keystrokes on a computer keyboard. The task of timing the keystroke intervals has been addressed and an experimental environment described. The measurable features of keystrokes have been listed along with some typical behaviour traits of keystroke dynamics. The problems that outlier data can cause have been discussed and a method to reduce their effect introduced.

Chapter 4

Statistical Techniques

4.1 Introduction

Many of the keystroke dynamic methods use simple statistical techniques to classify keystroke rhythms [1,2,4,9]. From these, more complicated statistically based methods have evolved [3,5,6,7,8] - notably Biopassword™. The use of statistical methods has certain advantages:

- They are proven techniques, used in many applications over a long period of time.
- The behaviour of statistical classifiers can be predicted.

This enables probabilities of outcomes, thresholds and limits to be accurately measured. . This is especially important in security applications, as the integrity of the system to intrusion must be known.

The vector distance measure is a fundamental tool in the field of pattern recognition. Proving particularly useful where data is consistently similar within groups and dissimilar across groups. As a starting point in this pattern recognition task, an investigation was performed to see how well simple distance functions could discriminate individual typing profiles. Two popular distance functions were implemented, the Euclidean and Manhattan (or City Block). These methods are used as a benchmark with which to compare results achieved with neural networks in the other chapters.

4.2 Euclidean Distance

The first method to be considered was based on the Euclidean distance (4.1) where, x_i = current vector of time samples of length i and y_i = reference vector of time samples of length i. This measure was selected as it is widely used in the field of pattern recognition. Samples from individuals were divided into two sets – a set used as a reference template and a set used for testing. The reference template was constructed as a mean prototype. A mean prototype refers to a class which is described by one point in the pattern space (usually the mean). Five of the samples from each user were averaged to produce a mean vector for each user.

$$d = \left(\sum_i (x_i - y_i)^2 \right)^{1/2} \tag{4.1}$$

This mean vector was then used to determine whether a test sample belonged to the individual or not. The Euclidean distance was computed between the test sample and the reference sample. If this distance was below a threshold value the sample was deemed to originate from the correct individual. Any samples which were over this threshold level were deemed to be from an impostor. Changing the level of this threshold will affect the behaviour of the verifier. A higher threshold will reduce the chance of the correct user being rejected, but increase the chance of an impostor being accepted. These two errors are False Rejection (FR - Type 1) and False Acceptance (FA - Type 2). There will be an optimum threshold level where both these errors are minimised, as described in section 2.5 - Evaluating Performance. This threshold level depends on the consistency of samples produced by the individual. Reviewing the response from each user would be too time consuming and unnecessary as most fit into three distinct categories: high error, average error and low error. In the following

sections these categories relate to the optimum levels for FA/FR errors for each user.

4.2.1 Unbiased Data

Firstly the method was implemented with the unbiased data set.

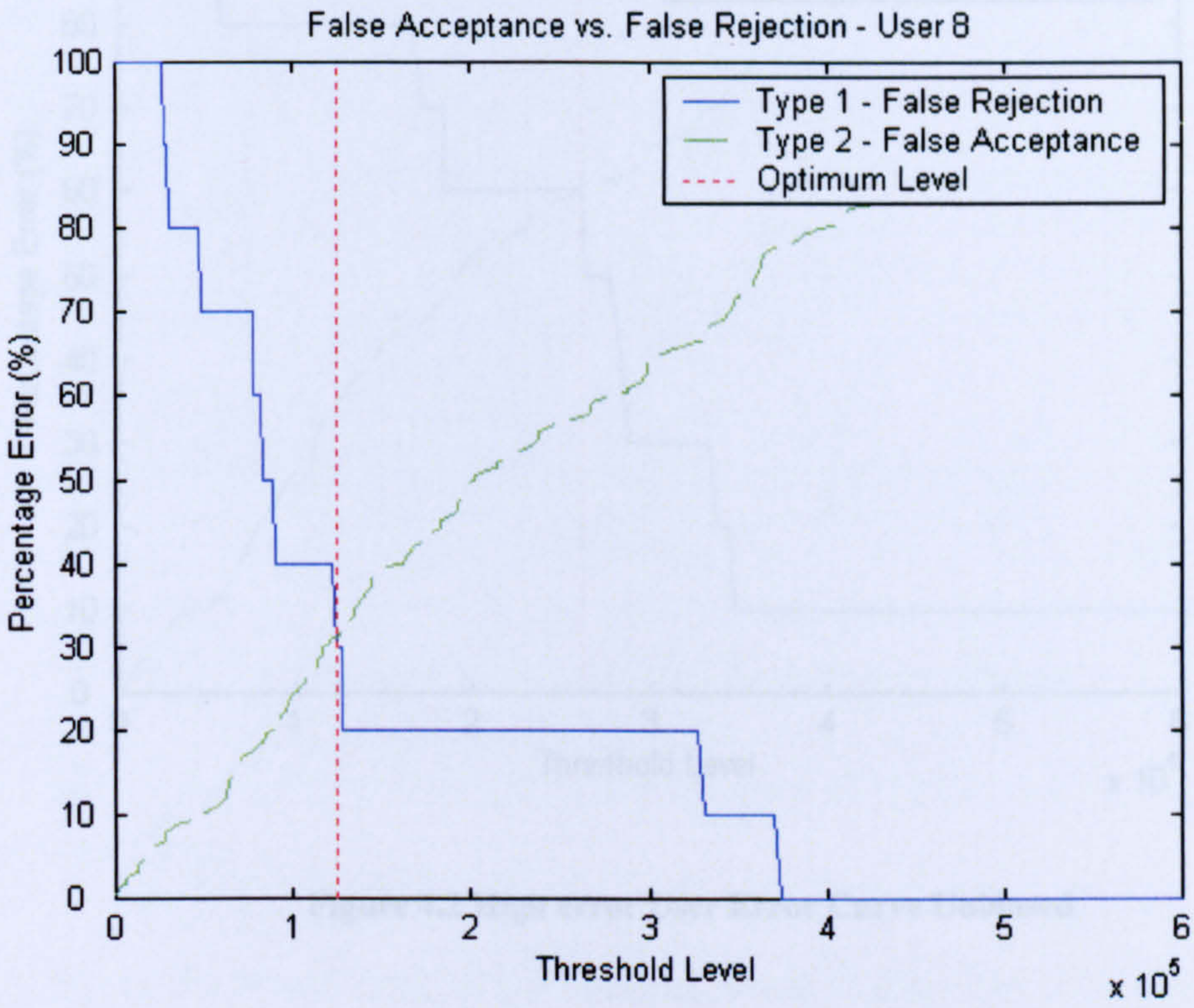


Figure 4.1 Average error User Error Curve Unbiased

The graph in Figure 4.1 shows the typical error curves associated with an average error user. This is the response which the majority of users in the data set generate. Figure 4.2 shows a high error response for a user with an inconsistent typing style. There are two users who display this behaviour. Figure 4.3 demonstrates the case for a consistent typist with a low error response from the verifier. Again there are only two users who fall into this category.

This highlights the fact that the performance in the level of security provided by keystroke dynamics is affected by the uniqueness and consistency of the typist. There

exists a large difference in the optimum FA and FR rate of nearly 60% between the best and worst users, 21 and 17.

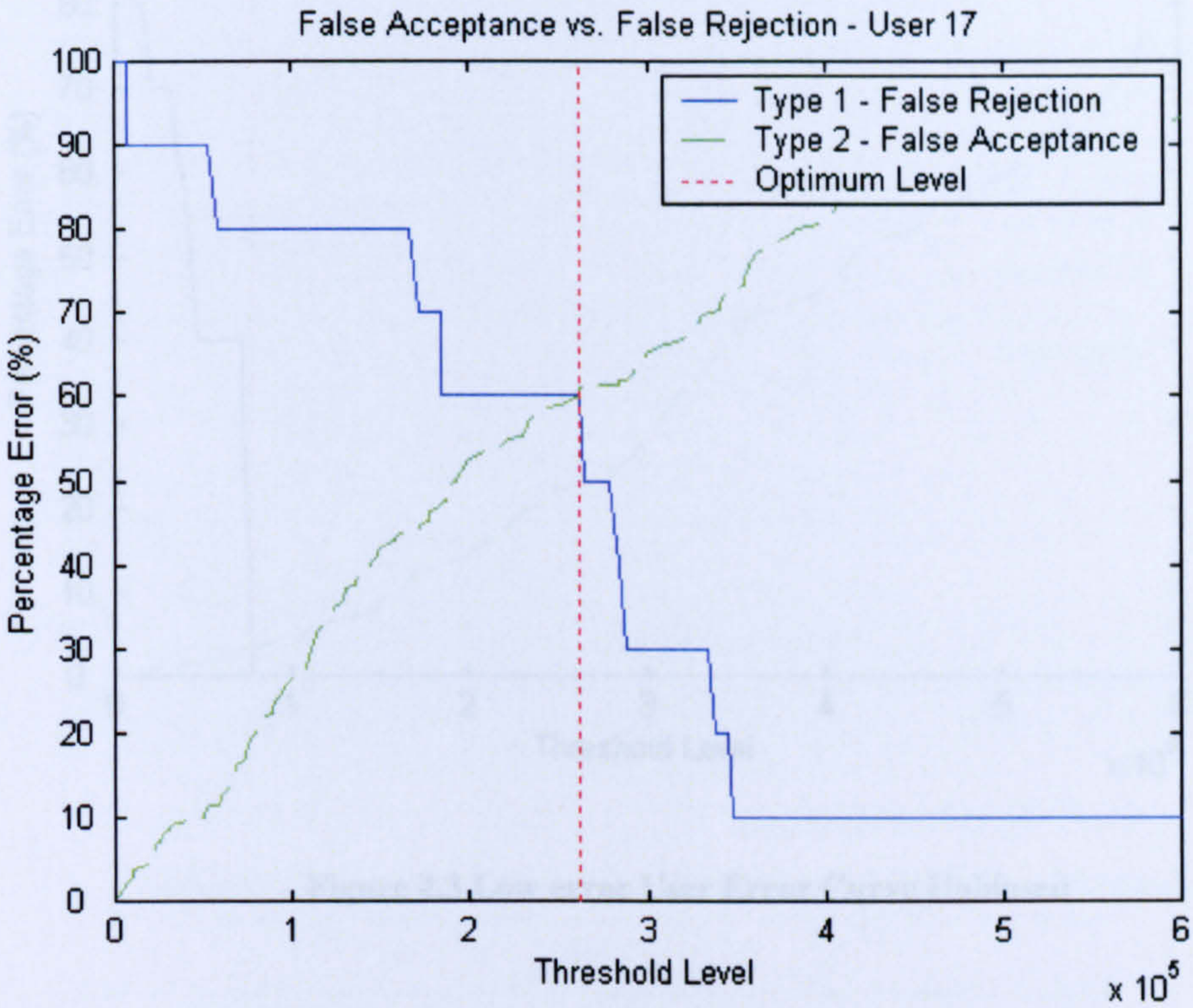


Figure 4.2 High error User Error Curve Unbiased

From Figure 4.4 each users optimum threshold errors for FA and FR can be seen plotted against that threshold level. Here the optimum errors for FA and FR shown in Figure 4.1, Figure 4.2 and Figure 4.3 are plotted against their relative threshold levels. This is also done for all the other users in the data set. A high concentration of optimum values for FA and FR can be seen around the average error and threshold. It is a general rule that the larger the threshold level, the more inconsistent the user is. This can be seen as the lowest and highest error points also correspond to the lowest and highest threshold levels. Most users are around the average errors for FR (29.5238%) and FA (31.5476%) (Table 1).

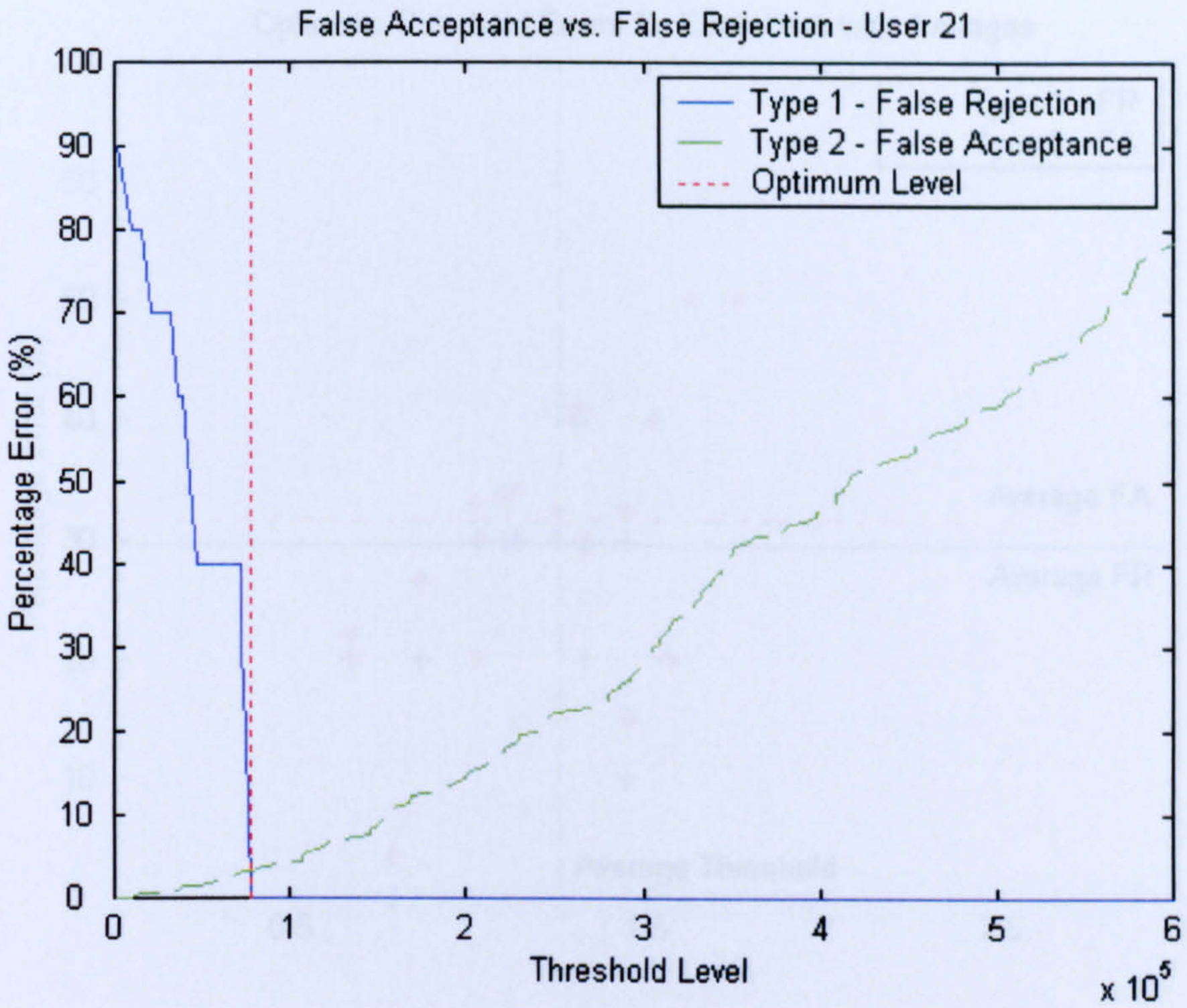


Figure 4.3 Low error User Error Curve Unbiased

4.2.2 Biased Data

Applying the biased data to this method yields slightly better results. All of the users exhibit a lower threshold level – indicating that they are more consistent when typing this type of data. Average error errors are slightly lower than for the unbiased data set with FR (25.3333%) and FA (27%).

Figure 4.5 shows the response of the average error user, with a high error user and low error user response shown by Figure 4.6 and Figure 4.7 respectively. Figure 4.8 shows the results of the optimum threshold level for all the users in the sample set. Here, the threshold level averages are lower than the unbiased set. By comparing Figure 4.8 with Figure 4.4 the reduction in average threshold is from almost 1.25 $x10^5$ to $0.5x10^5$. This suggests a more consistent typing style. The average errors have also reduced considerably from FR and FAs of 0.3 and 0.32 in Table 1 to an FR of 0.12 and an FA of 0.09 in Table 2.

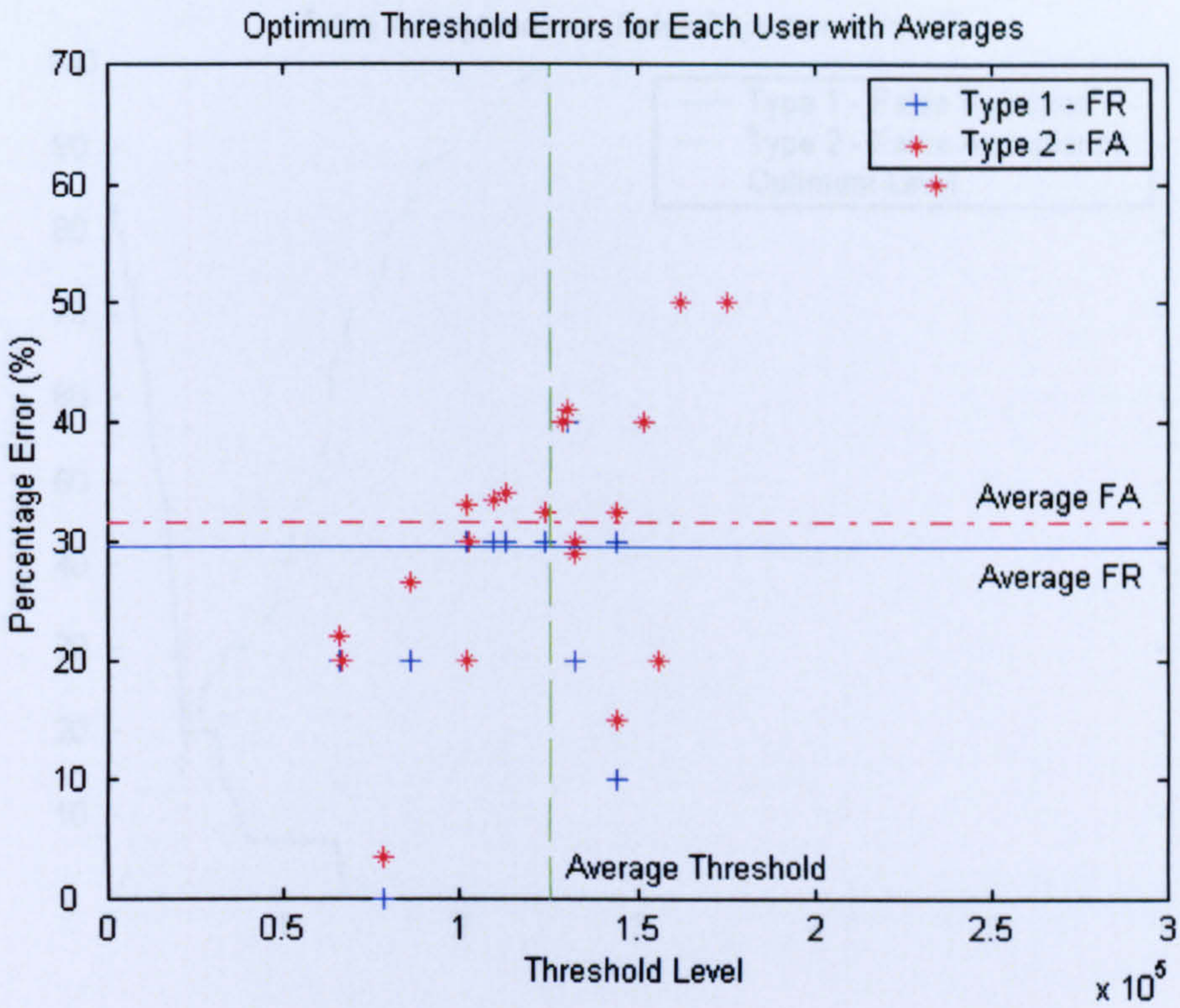


Figure 4.4 Optimum Threshold Error Levels Across Users Unbiased

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0.20	0.20	0.30	0.50	0.20	0.40	0.10	0.30	0.30	0.30
FA (%)	0.20	0.22	0.33	0.50	0.20	0.40	0.15	0.33	0.34	0.33

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0.40	0.30	0.20	0.20	0.40	0.20	0.60	0.30	0.50	0.30	0	**0.30**
FA (%)	0.40	0.30	0.27	0.29	0.41	0.20	0.60	0.30	0.50	0.34	0.04	**0.32**

Table 1 - Table of results for Euclidean Distance Unbiased

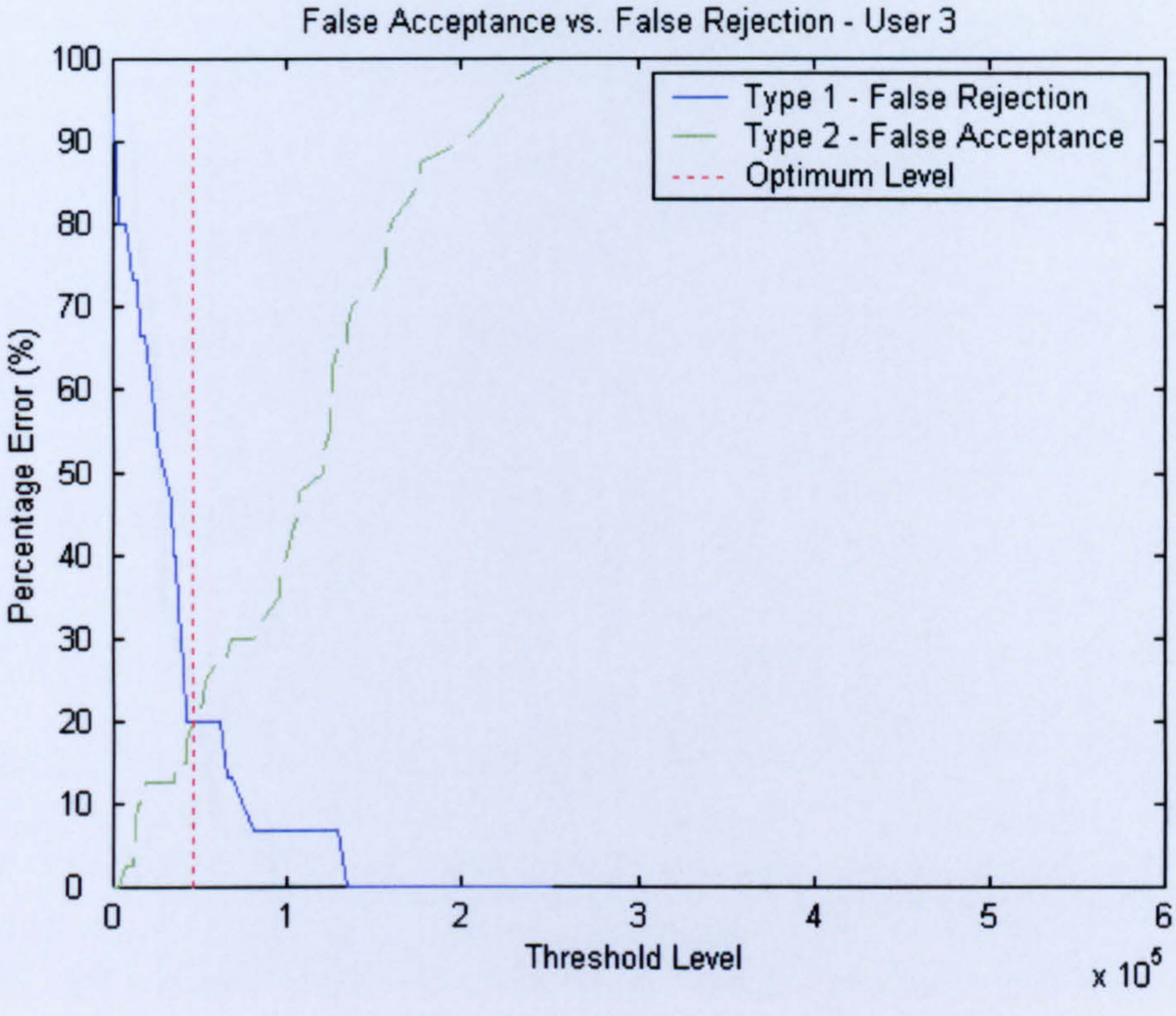


Figure 4.5 Average error User Error Curve Biased Data

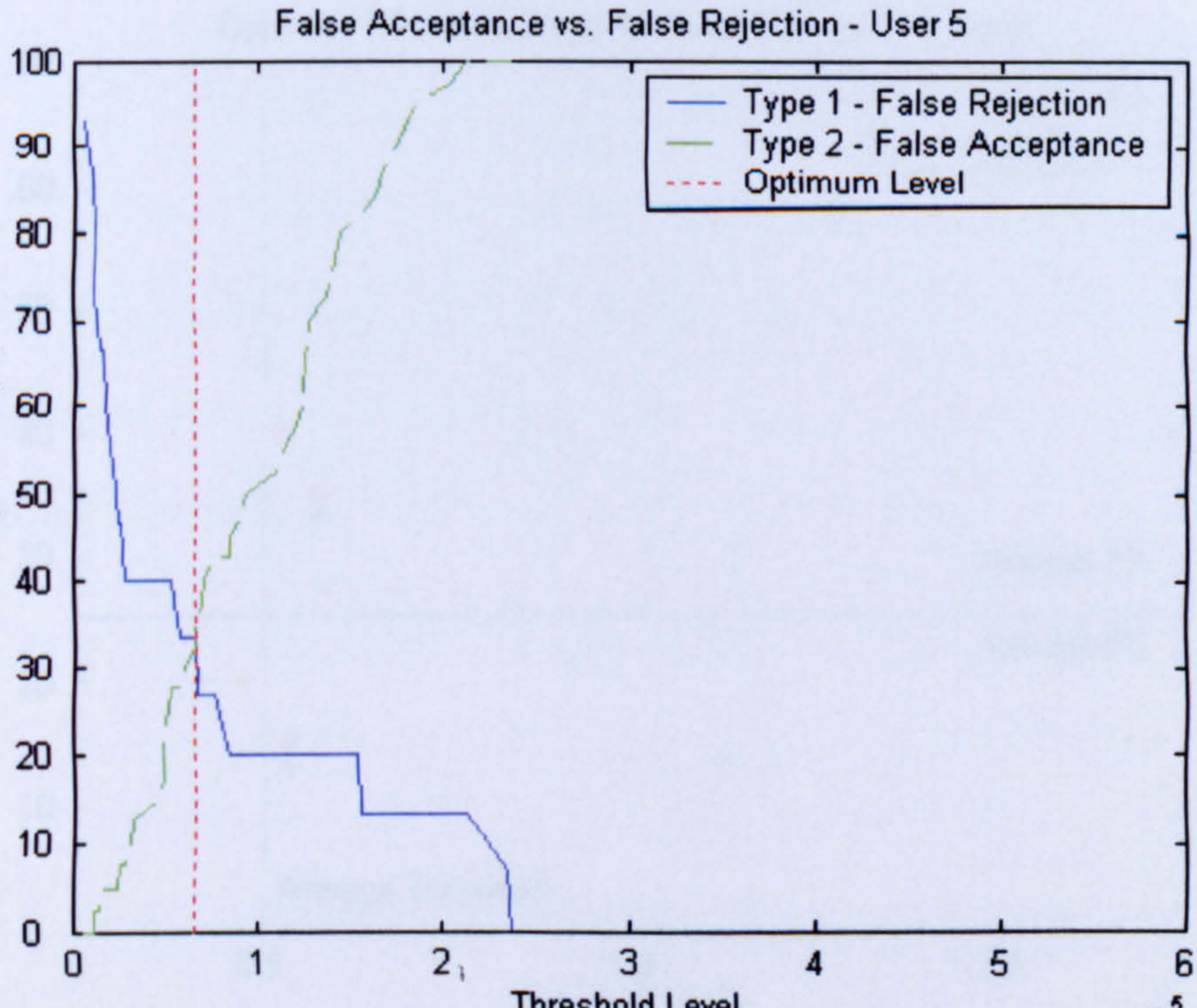


Figure 4.6 Bad User Error Curve Biased Data

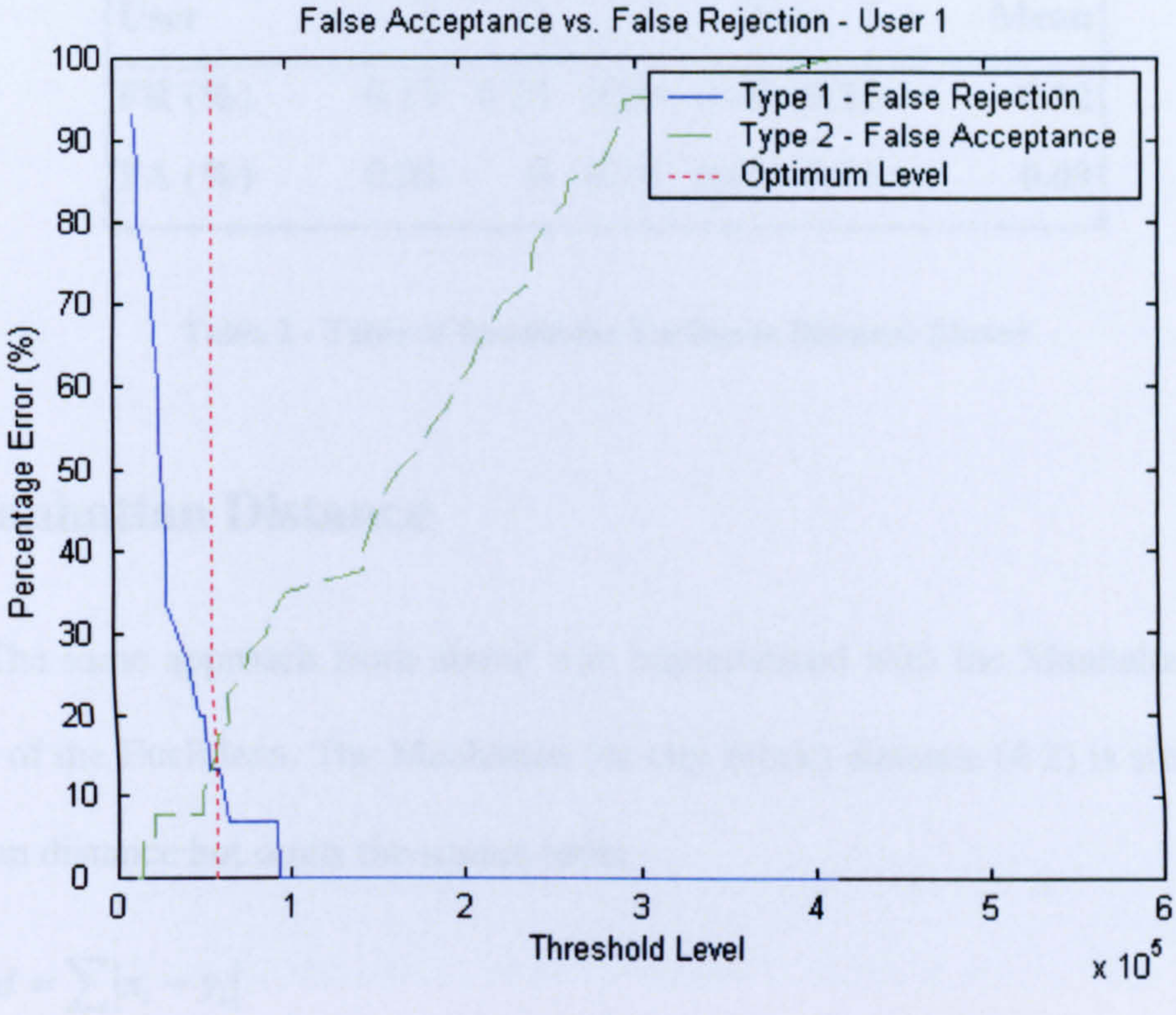


Figure 4.7 Low error User Error Curve Biased Data

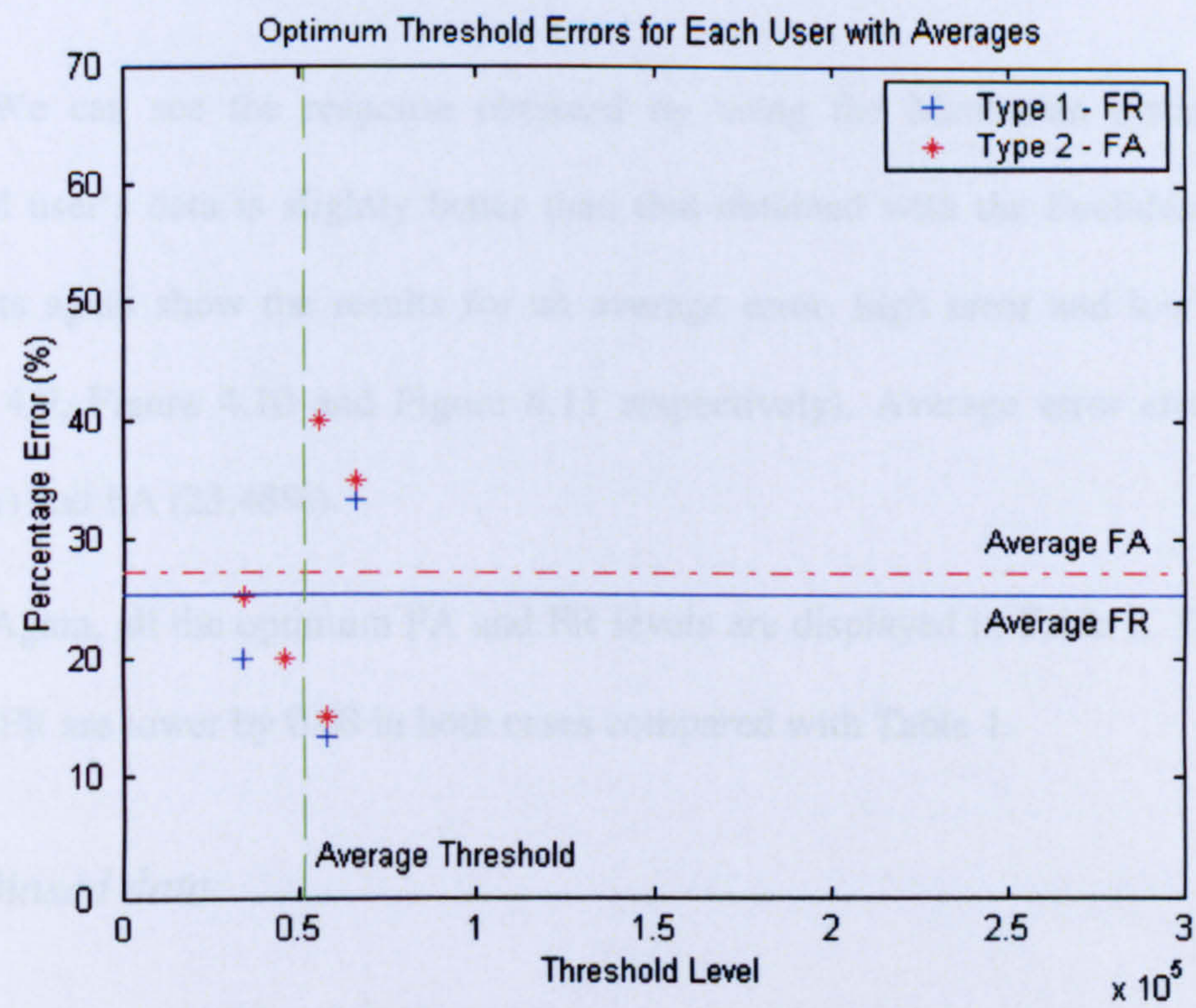


Figure 4.8 Optimum Threshold Error Levels Across Users Biased

User	1	2	3	4	5	Mean
FR (%)	0.13	0.20	0.20	0.40	0.33	**0.12**
FA (%)	0.08	0	0.18	0.05	0.15	**0.09**

Table 2 - Table of Results for Euclidean Distance Biased

4.3 Manhattan Distance

The same approach from above was implemented with the Manhattan distance in place of the Euclidean. The Manhattan (or city block) distance (4.2) is similar to the Euclidean distance but omits the square term.

$$d = \sum_i |x_i - y_i| \tag{4.2}$$

4.3.1 Unbiased data

We can see the response obtained by using the Manhattan distance in the unbiased user's data is slightly better than that obtained with the Euclidean distance. The plots again show the results for an average error, high error and low error user (Figure 4.9, Figure 4.10 and Figure 4.11 respectively). Average error errors are FR (21.34%) and FA (23.48%).

Again, all the optimum FA and FR levels are displayed in Table 3. The average FA and FR are lower by 0.08 in both cases compared with Table 1.

4.3.2 Biased data

Applying the biased data to this method results again in an improvement over the unbiased data with average error errors of FR (17.33%) and FA (19.00%). It can be seen in the figures for low error, high error and average error user that there is a general improvement over the Euclidean distance method.

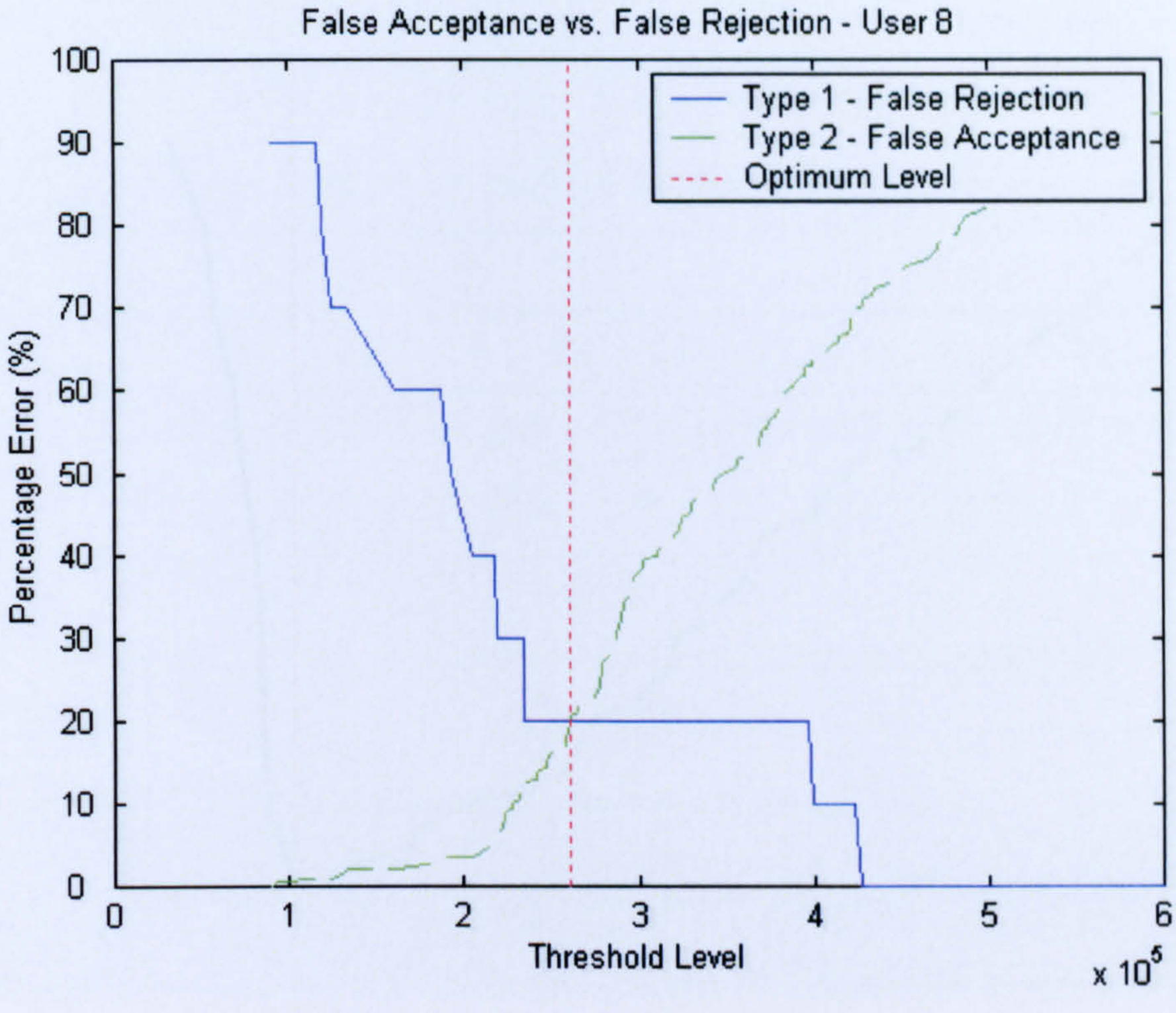


Figure 4.9 Average error User Error Curve Unbiased Data

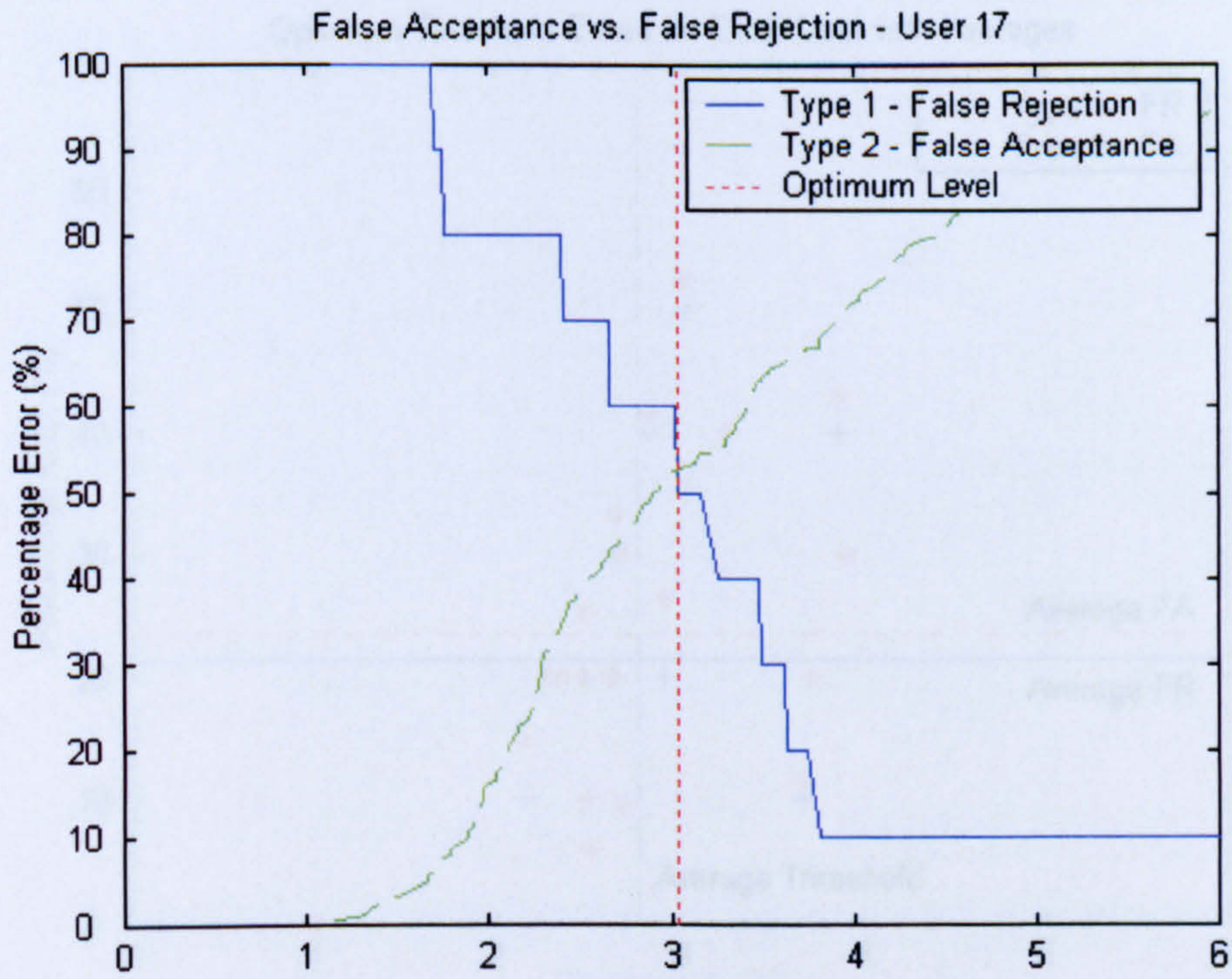


Figure 4.10 Bad User Error Curve Unbiased Data

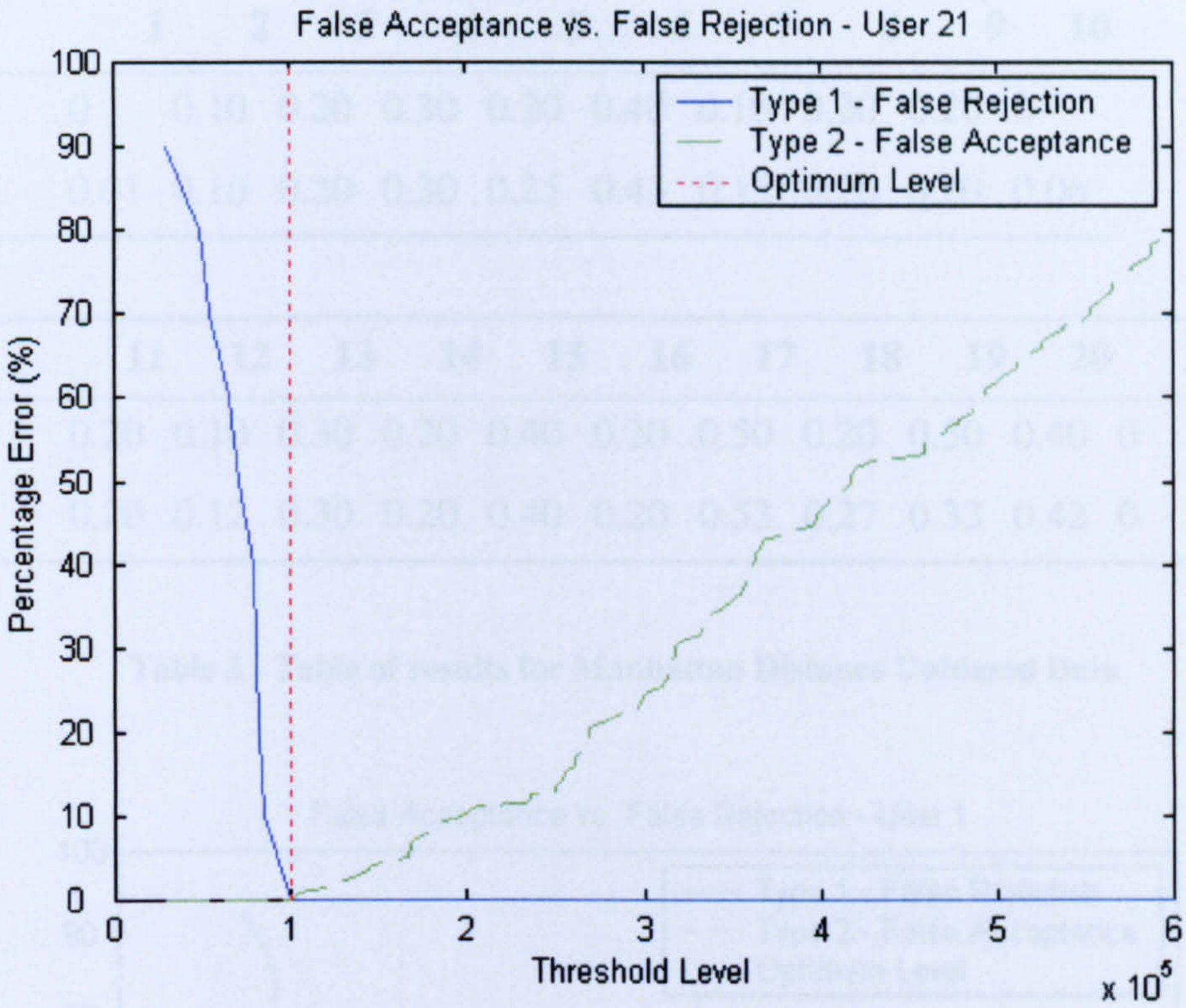


Figure 4.11 Low error User Error Curve Unbiased Data

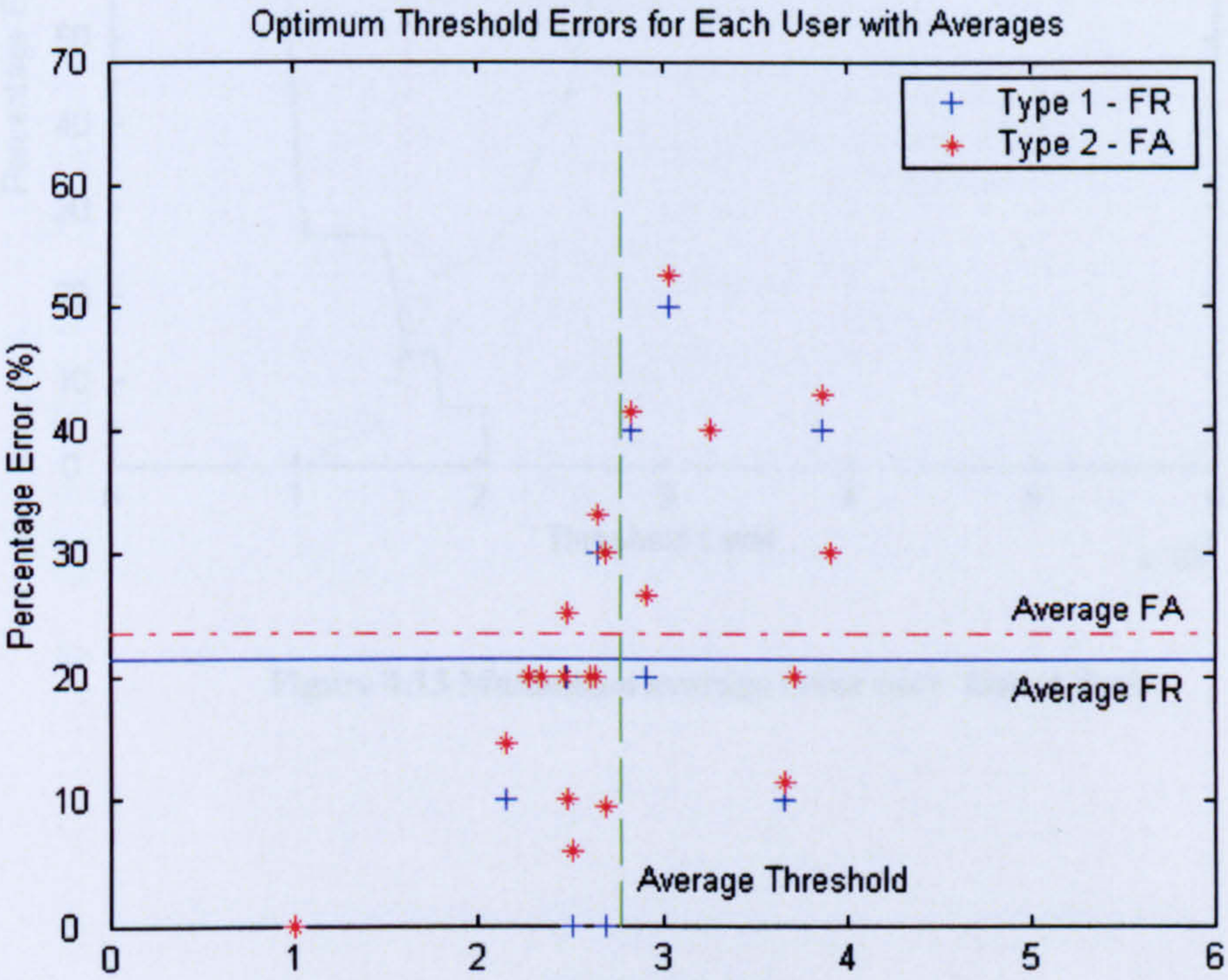


Figure 4.12 Optimum Threshold Error Levels Across Users Unbiased

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0	0.10	0.20	0.30	0.20	0.40	0.10	0.20	0.20	0
FA (%)	0.01	0.10	0.20	0.30	0.25	0.43	0.15	0.20	0.20	0.06

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0.20	0.10	0.30	0.20	0.40	0.20	0.50	0.20	0.30	0.40	0	**0.22**
FA (%)	0.20	0.12	0.30	0.20	0.40	0.20	0.53	0.27	0.33	0.42	0	**0.24**

Table 3 - Table of results for Manhattan Distance Unbiased Data

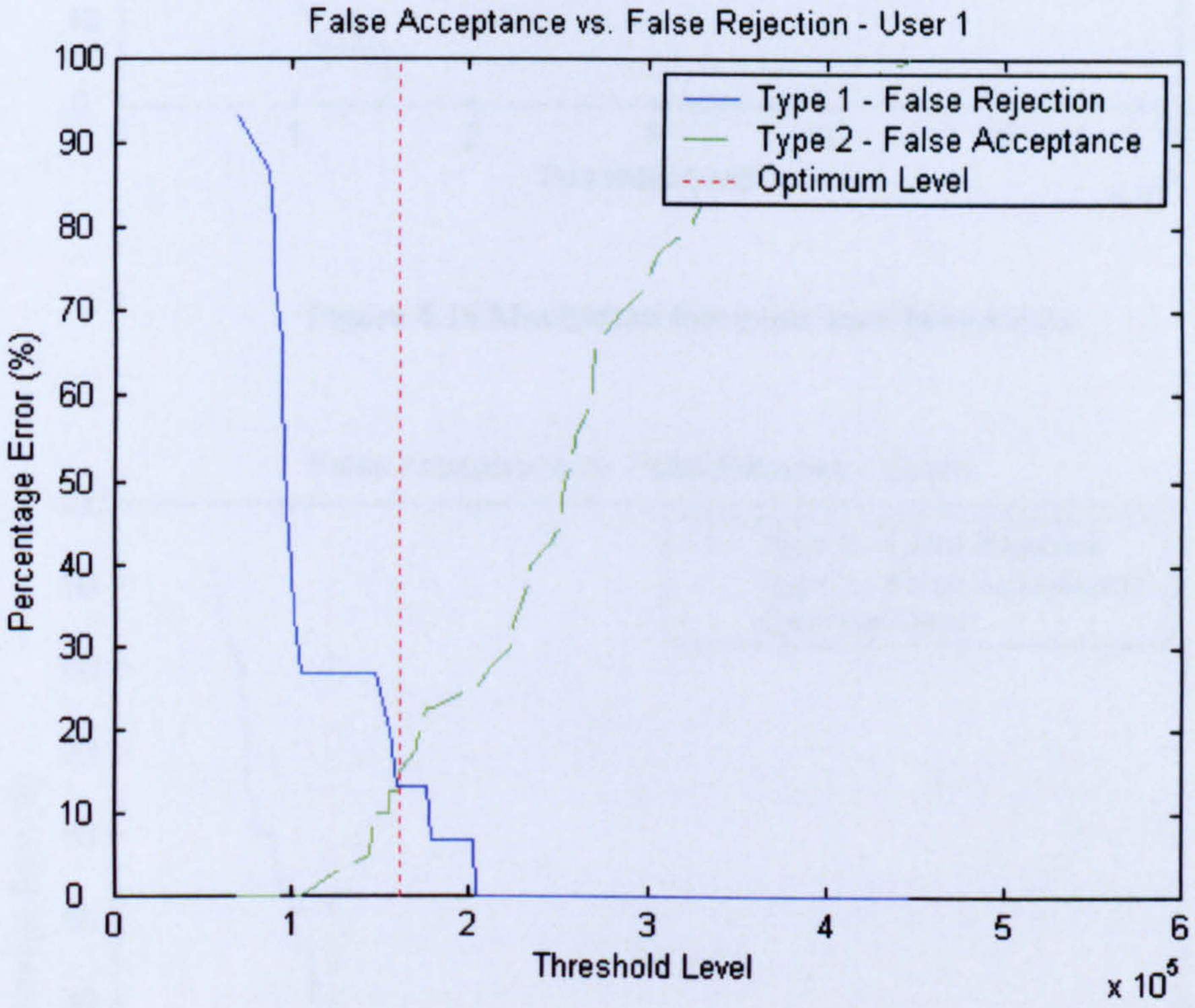


Figure 4.13 Manhattan average error user biased data

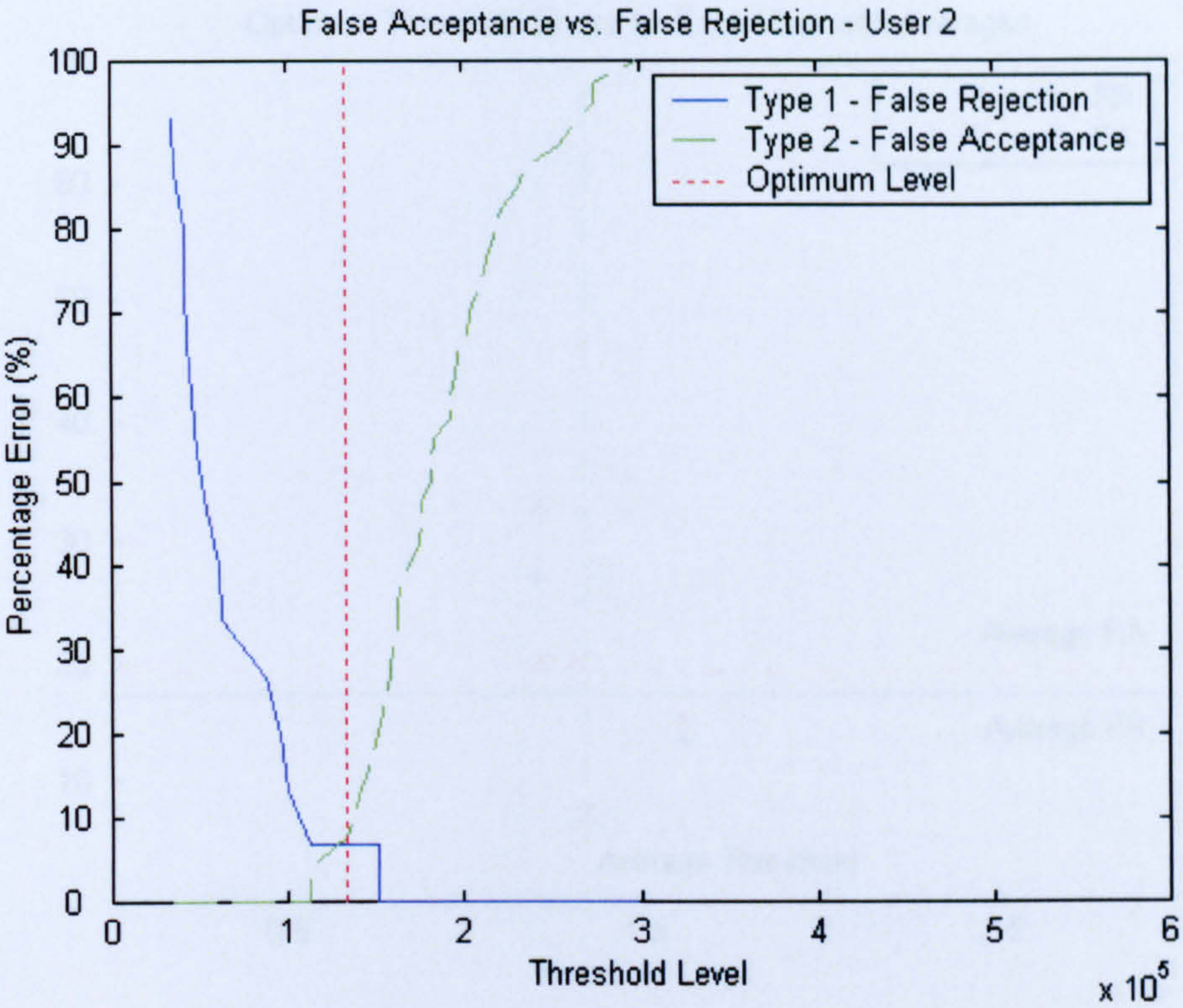


Figure 4.14 Manhattan low error user biased data

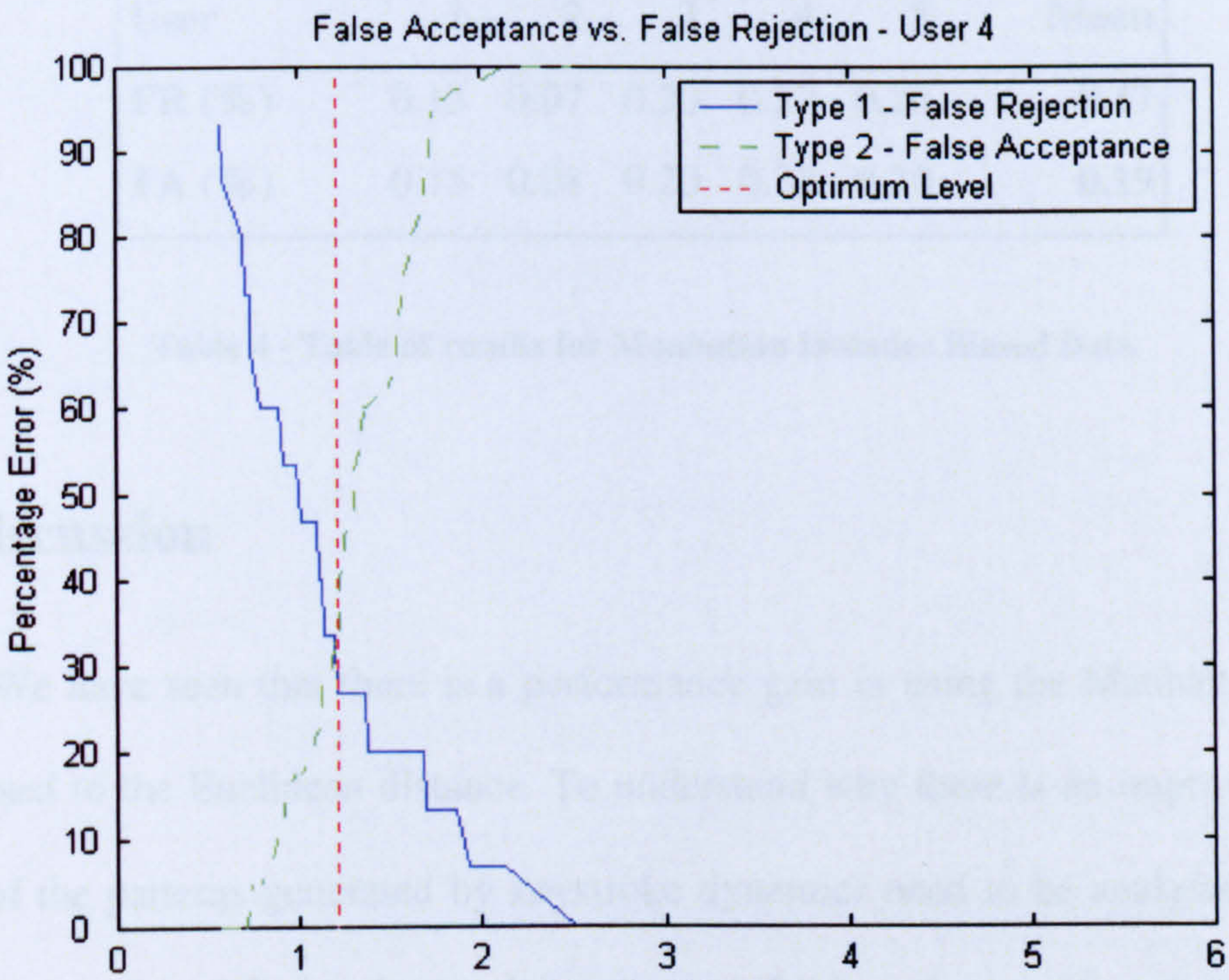


Figure 4.15 Manhattan high error user biased data

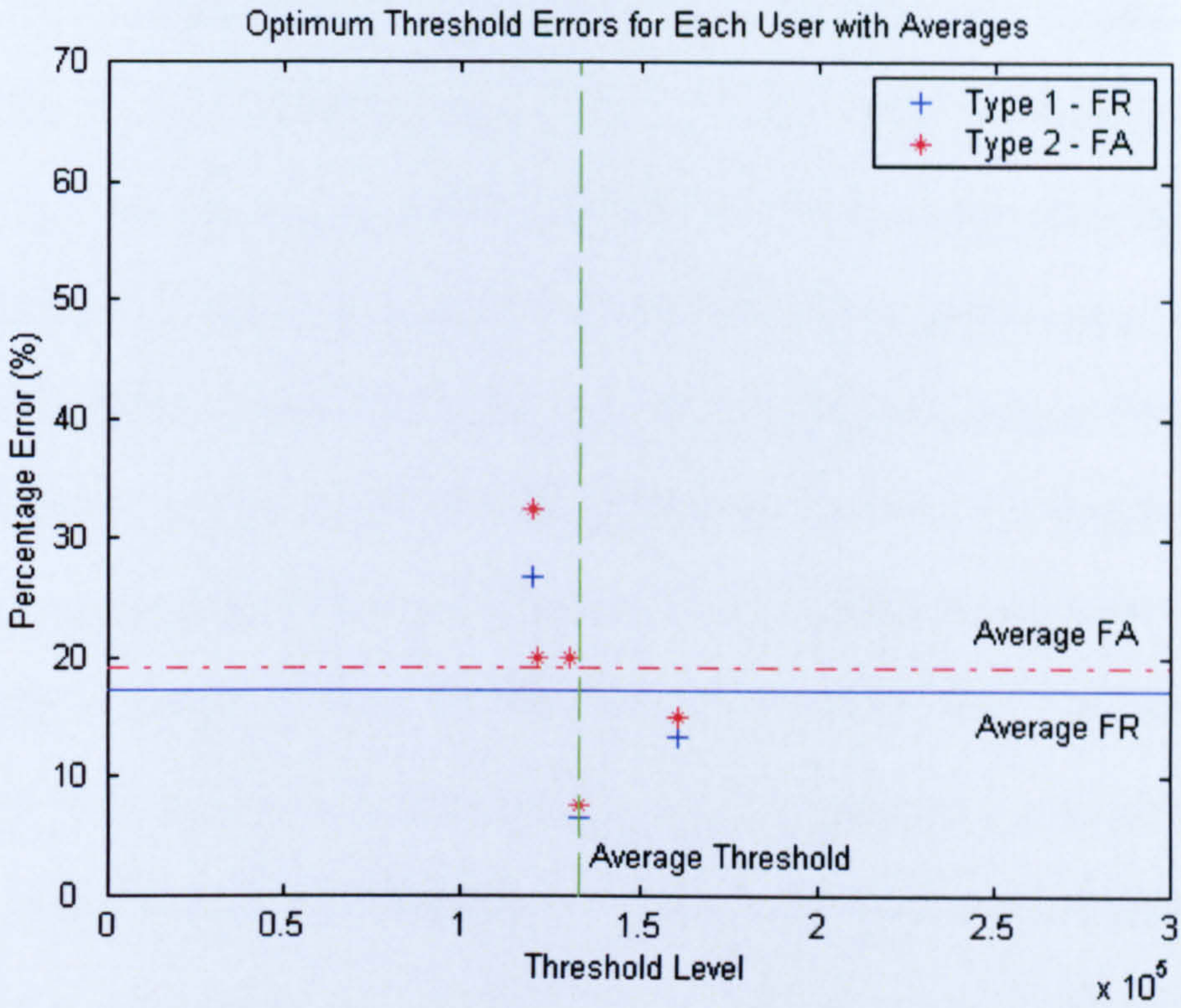


Figure 4.16 Manhattan optimum threshold error levels across users for biased data

User	1	2	3	4	5	Mean
FR (%)	0.13	0.07	0.20	0.27	0.20	**0.17**
FA (%)	0.15	0.08	0.20	0.33	0.20	**0.19**

Table 4 - Table of results for Manhattan Distance Biased Data

4.4 Discussion

We have seen that there is a performance gain in using the Manhattan distance as opposed to the Euclidean distance. To understand why there is an improvement, the nature of the patterns generated by keystroke dynamics need to be analysed. We have seen from section 3.7 that keystroke patterns exhibit a degree of variance between samples. This is generally true of many behavioural biometrics, as behaviour is a more fluid trait to capture compared to the solid, almost invariant physical biometrics such as

iris patterns. In particular, when a person is typing they often tend to pause or hesitate, interrupting the flow of their typing. This can occur whilst the brain is engaged in contemplating the next course of action to take, or is momentarily distracted. As many factors contribute to these distracting moments, hesitations tend to occur in a random fashion. The effects of these hesitations will create a keystroke interval within a sample which is uncharacteristic of the interval stored in the template. The Euclidean distance contains a square term which can exaggerate a large difference between a keystroke interval and completely eclipse all other intervals in the sample. The effect that this outlier datum point has on the Manhattan distance is reduced as it contains no square term. By using the Manhattan distance instead of the Euclidean distance, the area within which a sample could be classified as the correct user is increased, leading to the possibility that more impostors could be accepted by the classifier and hence an increase in the false acceptance error. To combat this problem, other means for dealing with outlier data are required, such as the z-score [25] pre-processing described in section 3.8.

4.5 Summary

In this chapter, a popular statistical approach to pattern recognition of keystroke dynamics has been implemented. The results show the varying degrees of performance that individual users exhibit. The difference in results for the two distance measures of Euclidean and Manhattan distance has been shown. This difference in performance has been linked to the squared term in the Euclidean distance measure inflating the effects of outlier datum points due to user hesitation whilst typing. The dependence of errors that exists in biometric systems between FA and FR has been illustrated.

Chapter 5

Backpropagation Neural Network

5.1 Introduction

The backpropagation neural network (BPNN) [26] is a well seasoned network which has been successfully applied to many pattern recognition problems [27]. It typically consists of a feed forward network with an input layer, an output layer and at least one hidden layer. The backpropagation gradient descent technique [27] is then applied to train the network's weights in a supervised mode in order to minimise the error with the output node targets.

The BPNN has the ability to solve complex problems with compact network structures [27]. However, its learning process is often unpredictable, time consuming and can become stuck in a solution that is not optimal [27]. Network design is done on a trial-and-error basis as there are many factors which can be fine-tuned to improve the network's performance. This provides additional attributes which cannot be optimally set during enrolment. The behaviour of the network is also not totally predictable which has proved unacceptable in mission critical applications and is a problem when establishing the levels of security provided [28].

5.2 Identification and BPNN

The backpropagation algorithm, while suffering from well documented

problems can, under the right conditions, perform as a highly efficient non-linear classifier [27]. It is therefore useful to implement an identification problem and observe how the BPNN performs. In this experiment, the BPNN was trained to identify a user out of the group of twenty-one using the unbiased data. In the unbiased data set, all users are typing the same phrase; therefore, any resulting differences across users (and similarities with users) will be due entirely to their typing rhythms. The degree to which the BPNN is able to classify these samples will indicate the level of individuality which exists across users. For this experiment the BPNN was set up as follows:

The training data set contained two thirds of the samples from each user. The test data set containing the remaining third of samples. Input data were normalised between -1 and 1. The BPNN architecture contained 16 input nodes, 21 hidden nodes, and 21 output nodes. The transfer function for the hidden layer was the 'tansig' squashing function [29]. This was chosen as the tansig function range is the same as that of the inputs. The number of nodes in the hidden layer were chosen to represent the twenty one different typing styles. The output transfer function was the 'logsig' function [29]. This was chosen as its range matches that of the output nodes. The target vector applied to the outputs was set to a one on the node corresponding with the user and a zero on all other nodes. Training was carried out with the resilient backpropagation method [29]. The resilient backpropagation method trains quickly and is not affected by the small changes that occur at the limit of sigmoid squashing functions as the direction of training is based on only the sign of the gradient, not the magnitude. This training method also has the advantage of fewer variables to set and is not sensitive to learning rate changes like some other methods.

The results of training for 2334 epochs give a correct classification of 92.4% of samples for the training data and 89.5% for the test data. These results were achieved after several attempts, as a good starting point provided by the random initialisation of

weights and biases is required for successful training. However, the network never failed to reach a minimum on other occasions and results for the test data were generally above 60%. This result shows that the neural network is clearly able to partition the pattern space into regions unique to each user. The response of the network with the test data is generally correct and of a high activation, this can be seen in Figure 5.1.

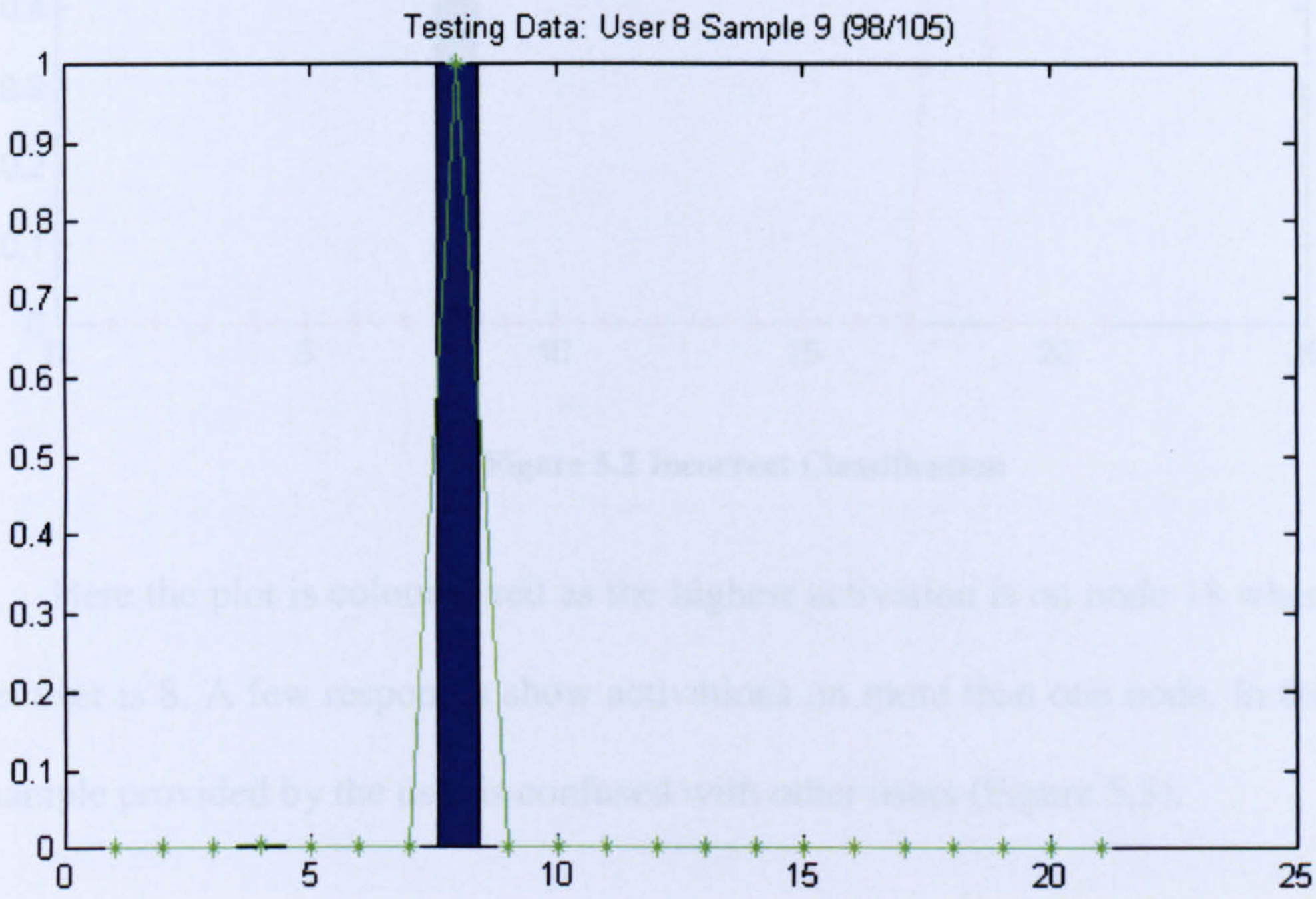


Figure 5.1 Correct classification

The line plot shows the activation on each of the output nodes. The bar shows the target user. A green plot shows the highest activation node on the output correctly corresponds to the target user. However, the network can also produce an error with a high activation as seen in Figure 5.2. This shows it is important to note that a high activation does not indicate a high degree of confidence in the decision of the network as it may in some statistical methods.

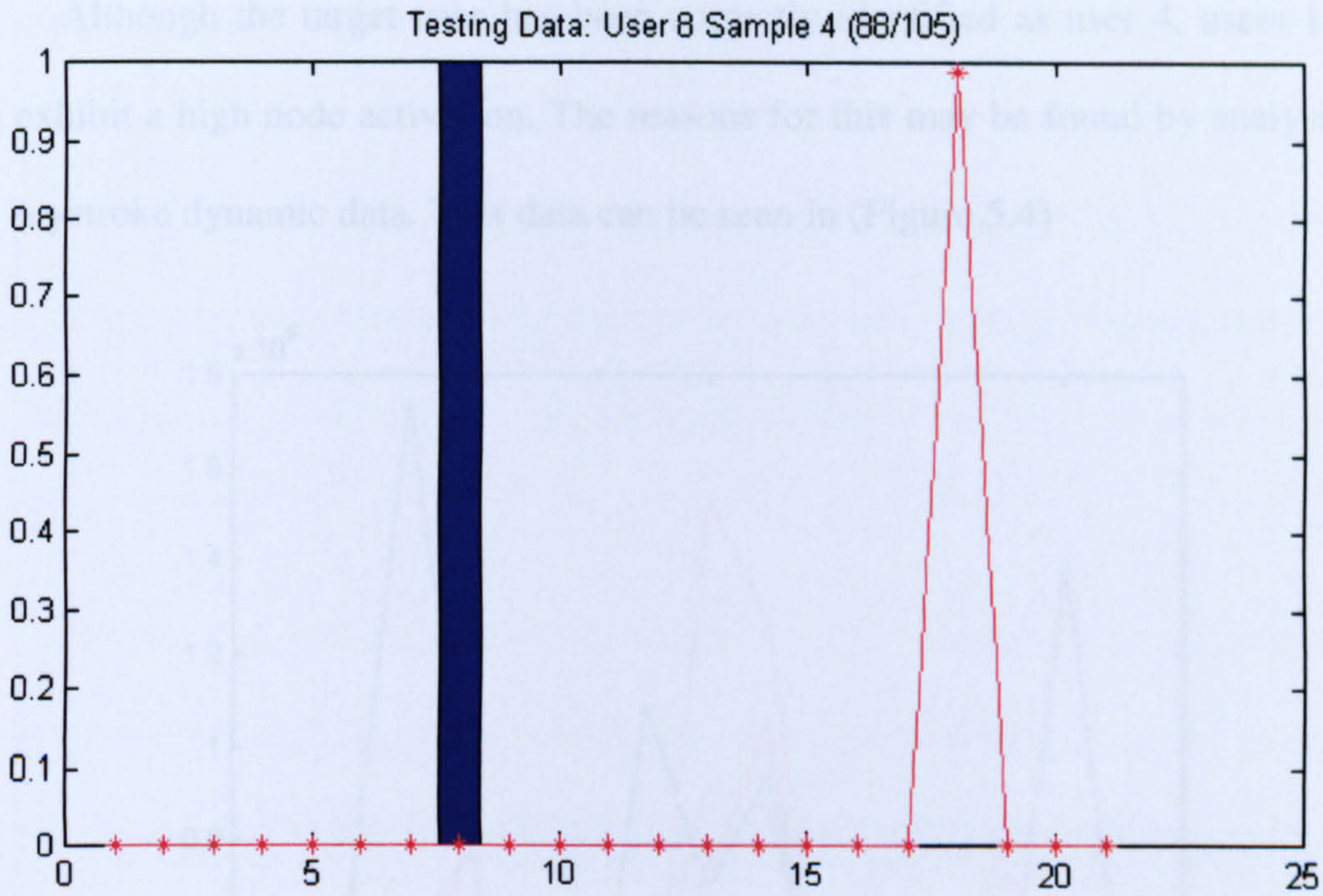


Figure 5.2 Incorrect Classification

Here the plot is coloured red as the highest activation is on node 18 whereas the target user is 8. A few responses show activations on more than one node. In this case, the sample provided by the user is confused with other users (Figure 5.3).

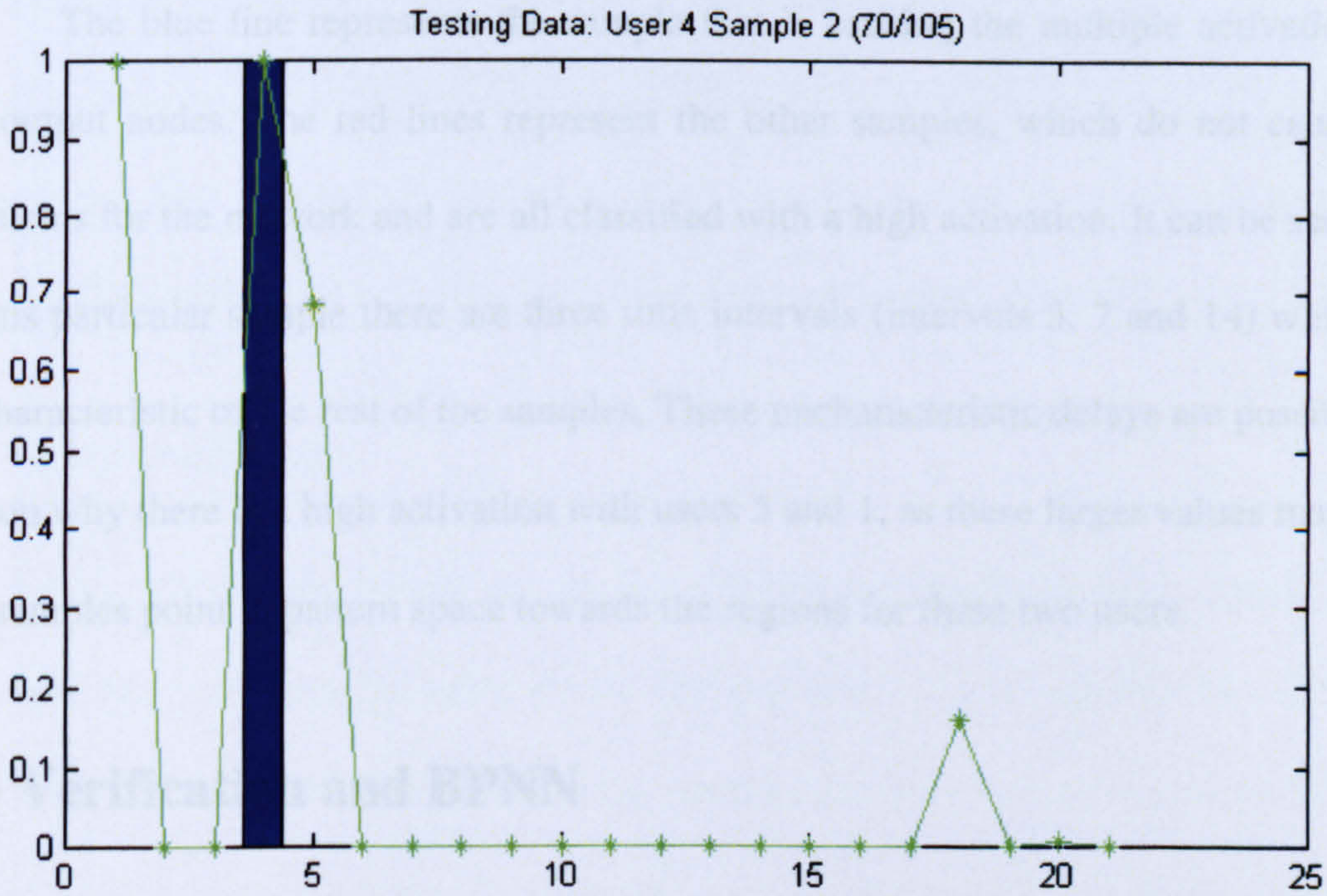


Figure 5.3 Correct classification with Other Activation

Although the target user has been correctly identified as user 4, users 1 and 5 also exhibit a high node activation. The reasons for this may be found by analysing the raw keystroke dynamic data. This data can be seen in (Figure 5.4)

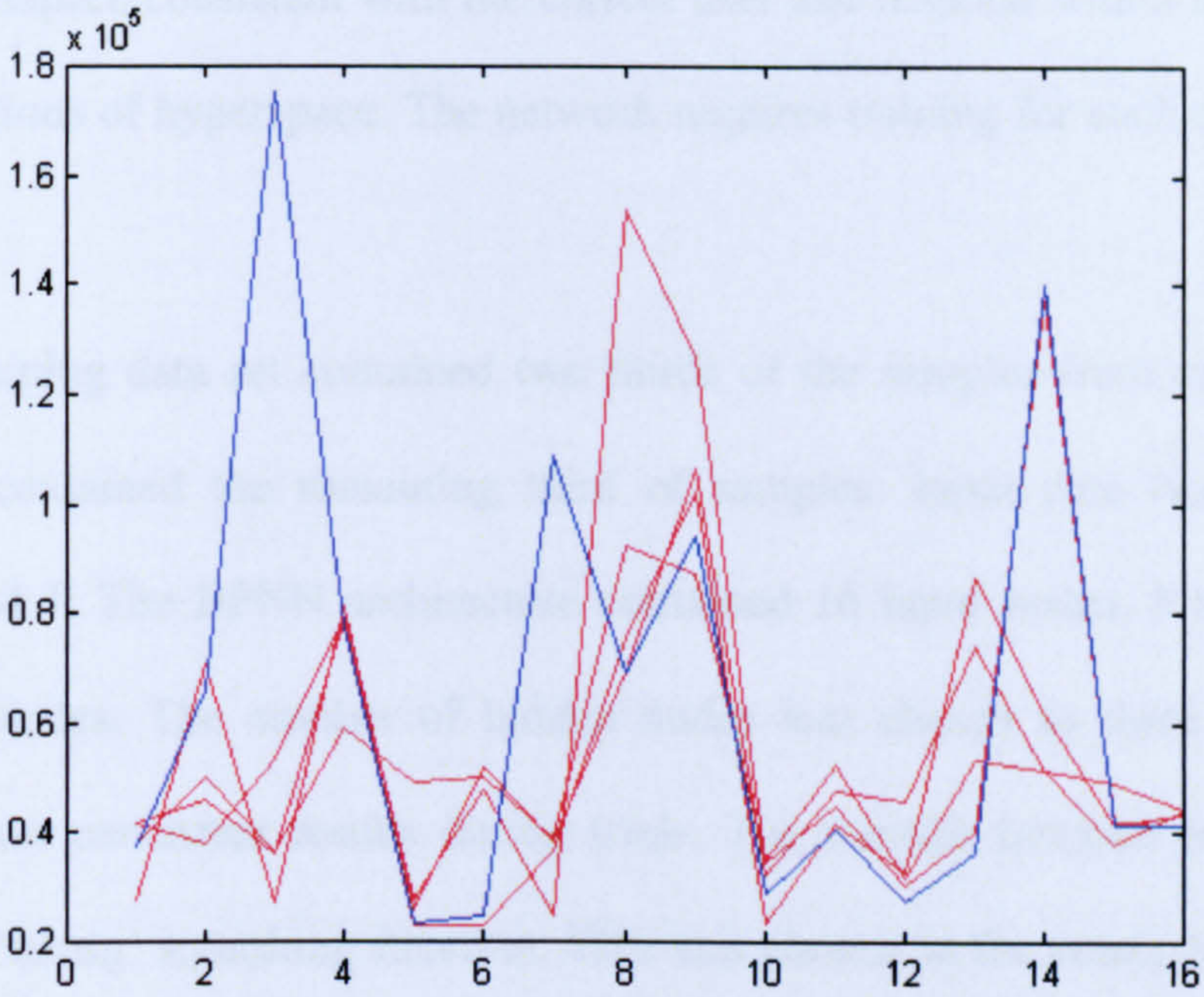


Figure 5.4 Uncharacteristic Datum

The blue line represents the sample that is causing the multiple activations on the output nodes. The red lines represent the other samples, which do not cause any problems for the network and are all classified with a high activation. It can be seen that in this particular sample there are three time intervals (intervals 3, 7 and 14) which are uncharacteristic of the rest of the samples. These uncharacteristic delays are possibly the reason why there is a high activation with users 5 and 1, as these larger values may push the samples point in pattern space towards the regions for these two users.

5.3 Verification and BPNN

The BPNN, after producing results for identification, was applied to the

verification problem. In this scenario the number of classes which the network must separate is reduced from twenty one to two. This equates to a mapping of hyperspace which is less complex. The network must generalise to an area of high activation in the region of hyperspace consistent with the correct user and respond with a low activation in all other regions of hyperspace. The network requires training for each of the users in the data set.

The training data set contained two thirds of the samples from each user. The test data set contained the remaining third of samples. Input data was normalised between -1 and 1. The BPNN architecture contained 16 input nodes, 3 hidden nodes, and 2 output nodes. The number of hidden nodes was chosen as three because this yielded the most consistent results during trials. The transfer function for the hidden layer was the 'tansig' squashing function. This was chosen as the tansig function range is the same as that of the inputs. The output transfer function was the 'logsig' function. This was chosen as its range matches that of the output nodes. The target vector applied to the outputs was set to a one on the first node and a zero on the second node if the sample corresponded with the correct user. If the sample did not correspond with the correct user then a zero was applied to the first node and a one on the second node. Training was carried out with the resilient backpropagation method [29].

5.3.1 Unbiased Data

The BPNN was trained on the unbiased data with the training regime set to minimise the false rejection error on the training data set. This enables the BPNN to generalise well for the correct user whilst keeping the false acceptance to a minimum. The behaviour of the network for each user can be more adequately compared, as the FR is minimised in all cases. The network's response to the training data shows that the resilient backpropagation algorithm is able to map the inputs to the respective targets

with minimal error. The only exception was user 20, where the network fails to train adequately.

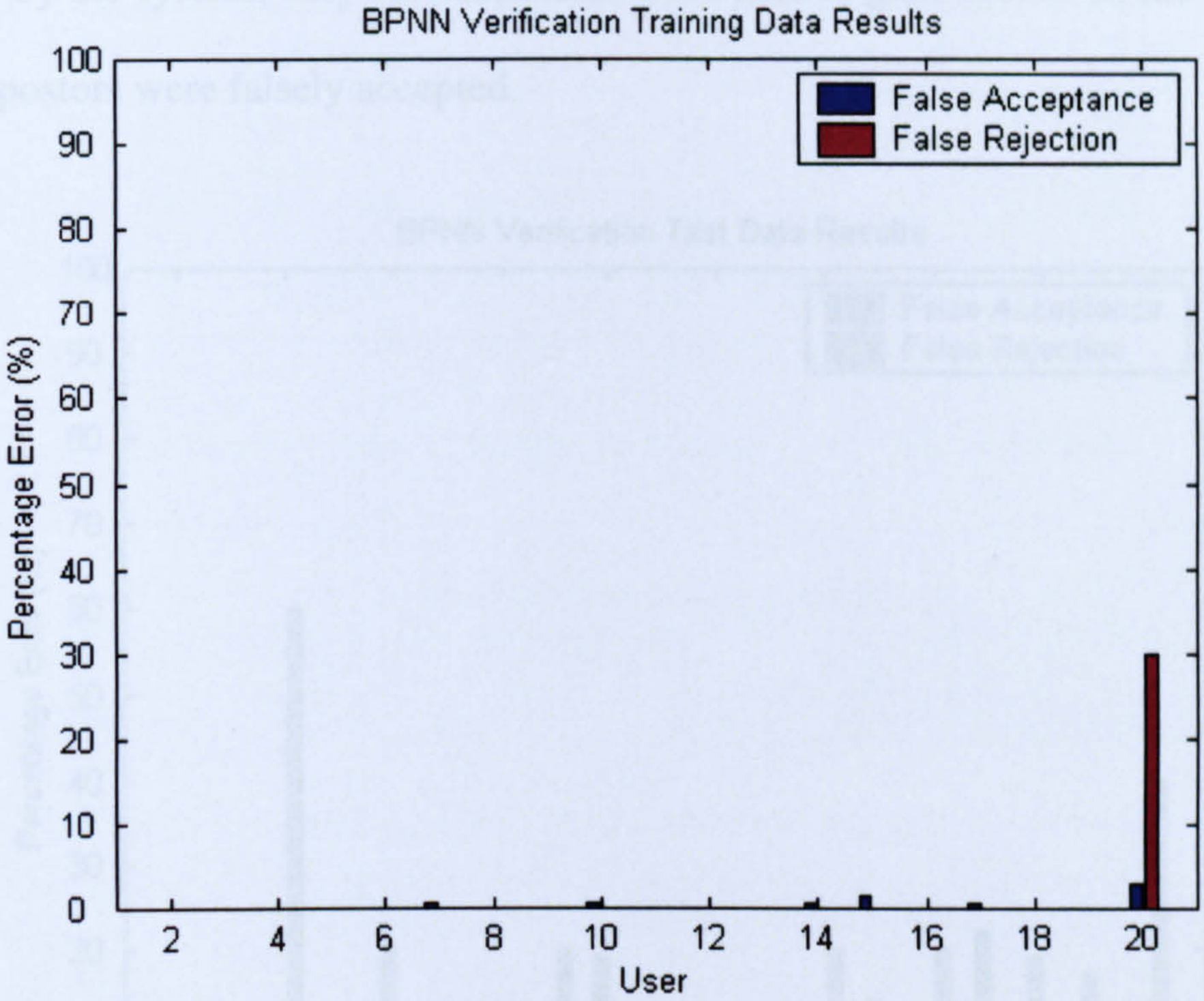


Figure 5.5 Results for Unbiased Training Data on BPNN

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0	0	0	0	0	0	0	0	0	0
FA (%)	0	0	0	0	0	0	0.01	0	0	0.01

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0	0	0	0	0	0	0	0	0	0.30	0	**0.014**
FA (%)	0	0	0	0.01	0.02	0	0.01	0	0	0.03	0	**0.001**

Table 5 - Table of results for Unbiased Training Data on BPNN

The network was then simulated on the previously unseen test data set. As expected the errors are higher on the more demanding test set. Ten users exhibit errors for FA and FR in the region of 0-10%, the remainder being around 20% apart from two

with FR errors of around 40% and 60%. These two errors are relatively high, however, the corresponding FA errors are low. This shows that while the correct user has been rejected by the system, very few impostors were able to gain access. In the case of user 4 no impostors were falsely accepted.

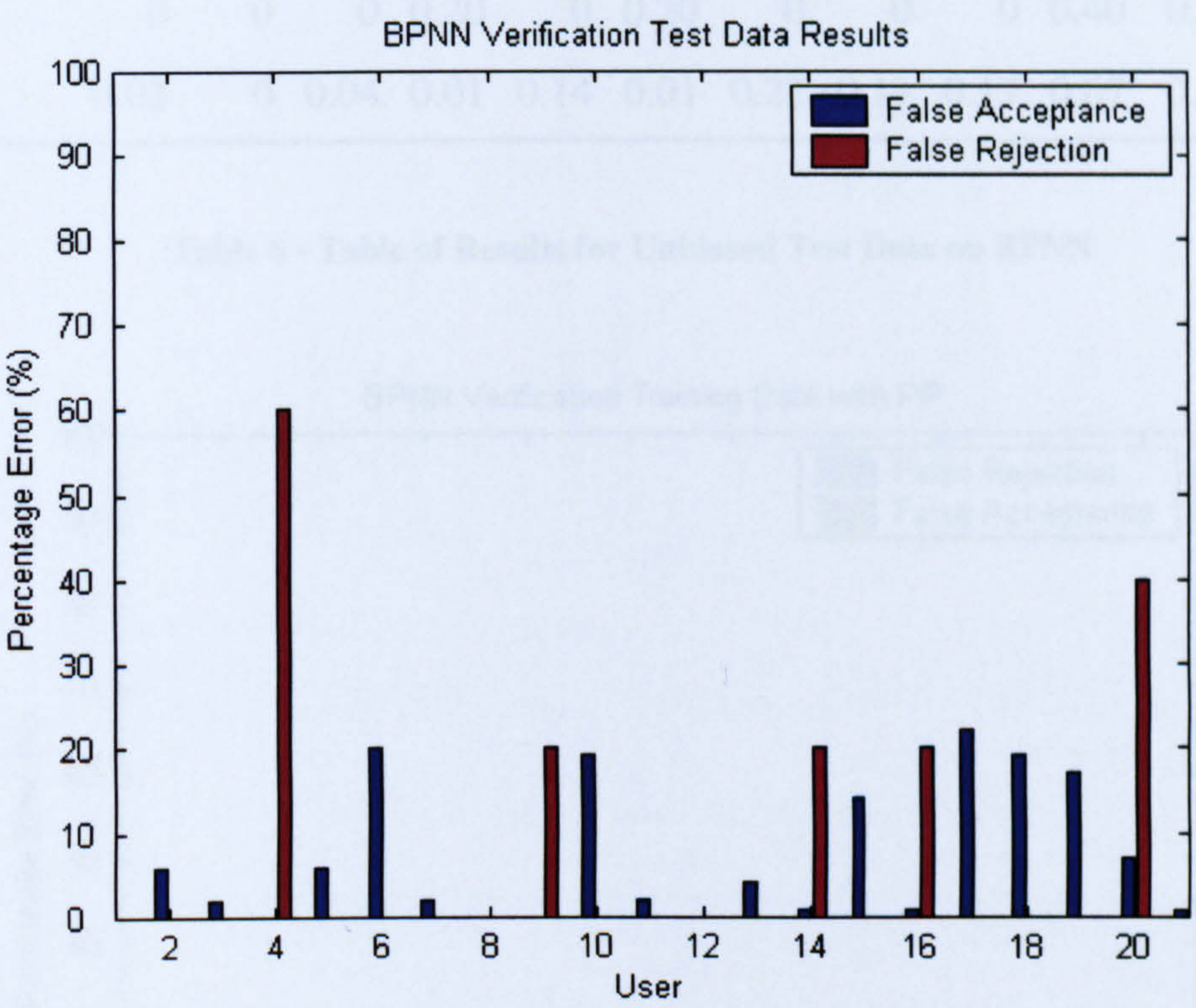


Figure 5.6 Results for Unbiased Test Data on BPNN

Applying pre-processed data to the BPNN it can be seen in (Figure 5.7) that the network trains to an acceptable level, which is slightly improved over the non pre-processed data. Users 19 and 20 still prove difficult for the network to adapt to, but there is a reduction in both FA and FR for these users under training.

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0	0	0	0.60	0	0.20	0	0	0.20	0
FA (%)	0.10	0.06	0.02	0	0.06	0	0.02	0	0	0.19

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0	0	0	0.20	0	0.20	0	0	0	0.40	0.20	**0.10**
FA (%)	0.02	0	0.04	0.01	0.14	0.01	0.22	0.19	0.17	0.07	0.01	**0.06**

Table 6 - Table of Results for Unbiased Test Data on BPNN

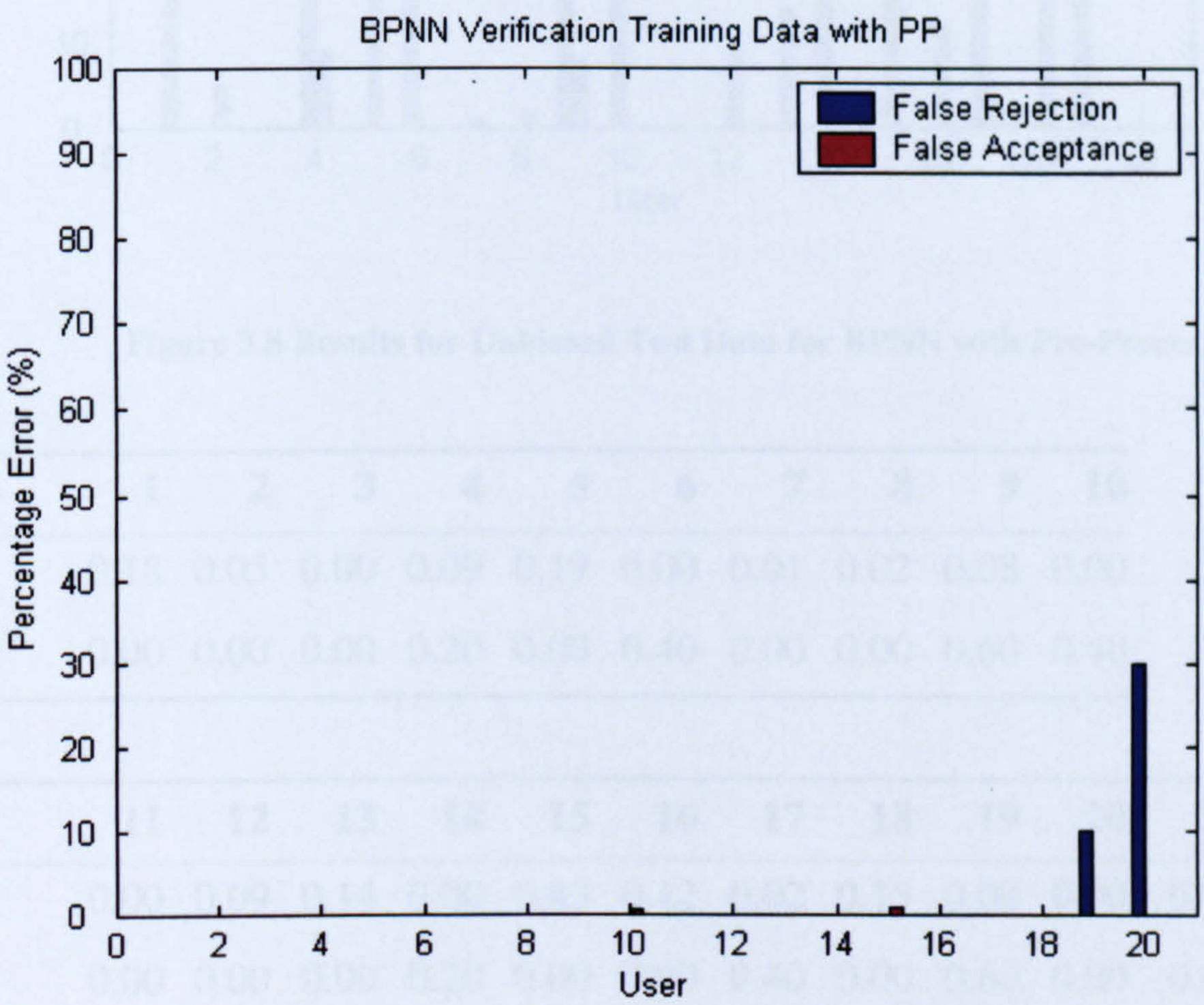


Figure 5.7 Results for Unbiased Training Data for BPNN with Pre-Processing

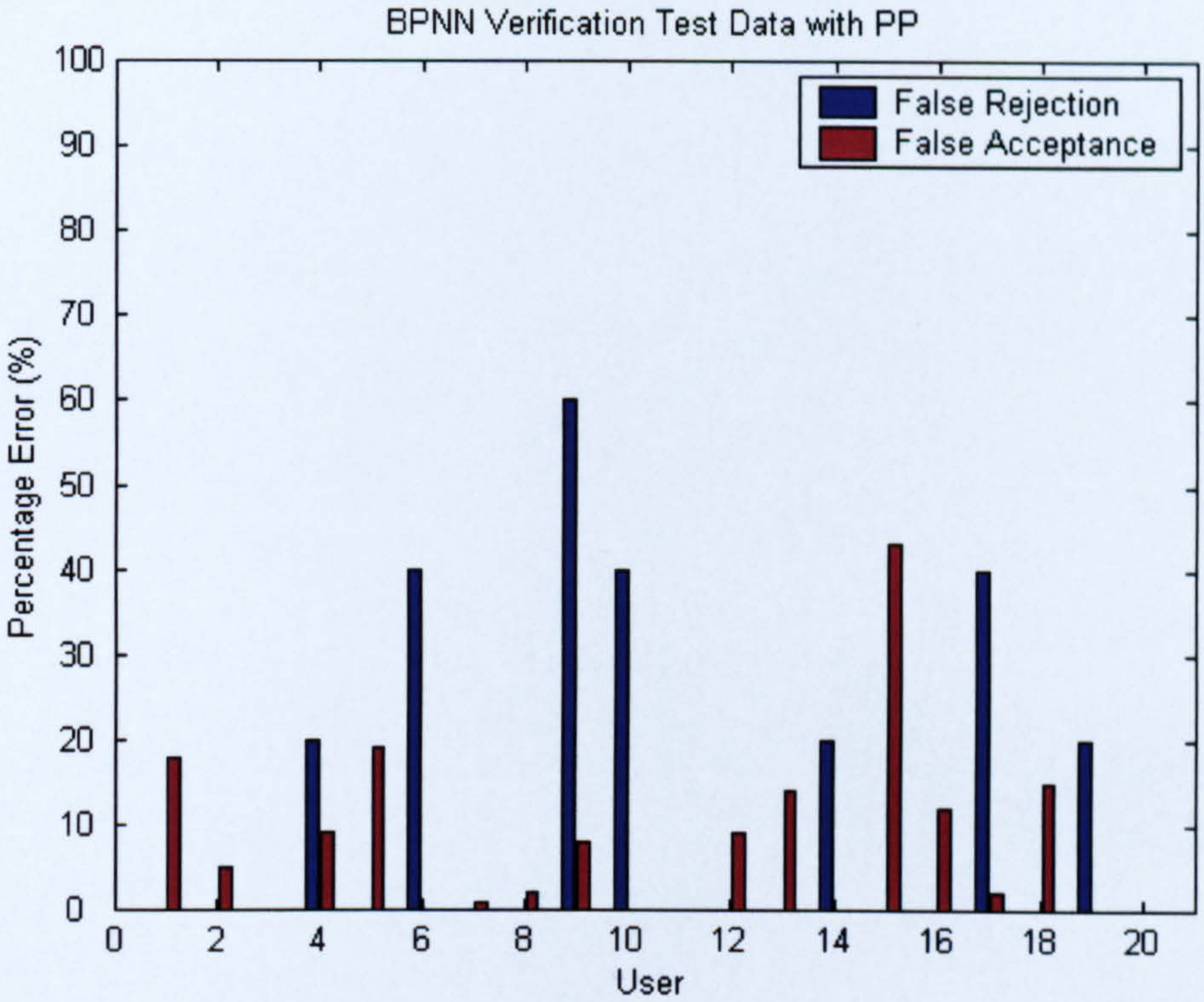


Figure 5.8 Results for Unbiased Test Data for BPNN with Pre-Processing

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0.18	0.05	0.00	0.09	0.19	0.00	0.01	0.02	0.08	0.00
FA (%)	0.00	0.00	0.00	0.20	0.00	0.40	0.00	0.00	0.60	0.40

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0.00	0.09	0.14	0.00	0.43	0.12	0.02	0.15	0.00	0.00	0.04	**0.08**
FA (%)	0.00	0.00	0.00	0.20	0.00	0.00	0.40	0.00	0.60	0.00	0.00	**0.13**

Table 7 - Table of Results for Unbiased Test Data for BPNN with Pre-Processing

The test data set with pre-processing shows a mixed picture. Some users experience a dramatic reduction in error, notably user 20, whilst others endure a rise in error for both FA and FR. Overall, the FR error is slightly reduced and the FA error slightly increased. This variation in the errors reported is possibly due to the BPNN learning inappropriately or over fitting some users.

5.3.2 Biased Data

The biased data shows results which are much improved over the unbiased data set. For training, every user attained zero error. As expected, the errors on both FA and FR are much reduced in the test set (Figure 5.9). Only user 3 is error free, but the network has generalised well for all other users. User 5 has the greatest error with 20% FA and FR.

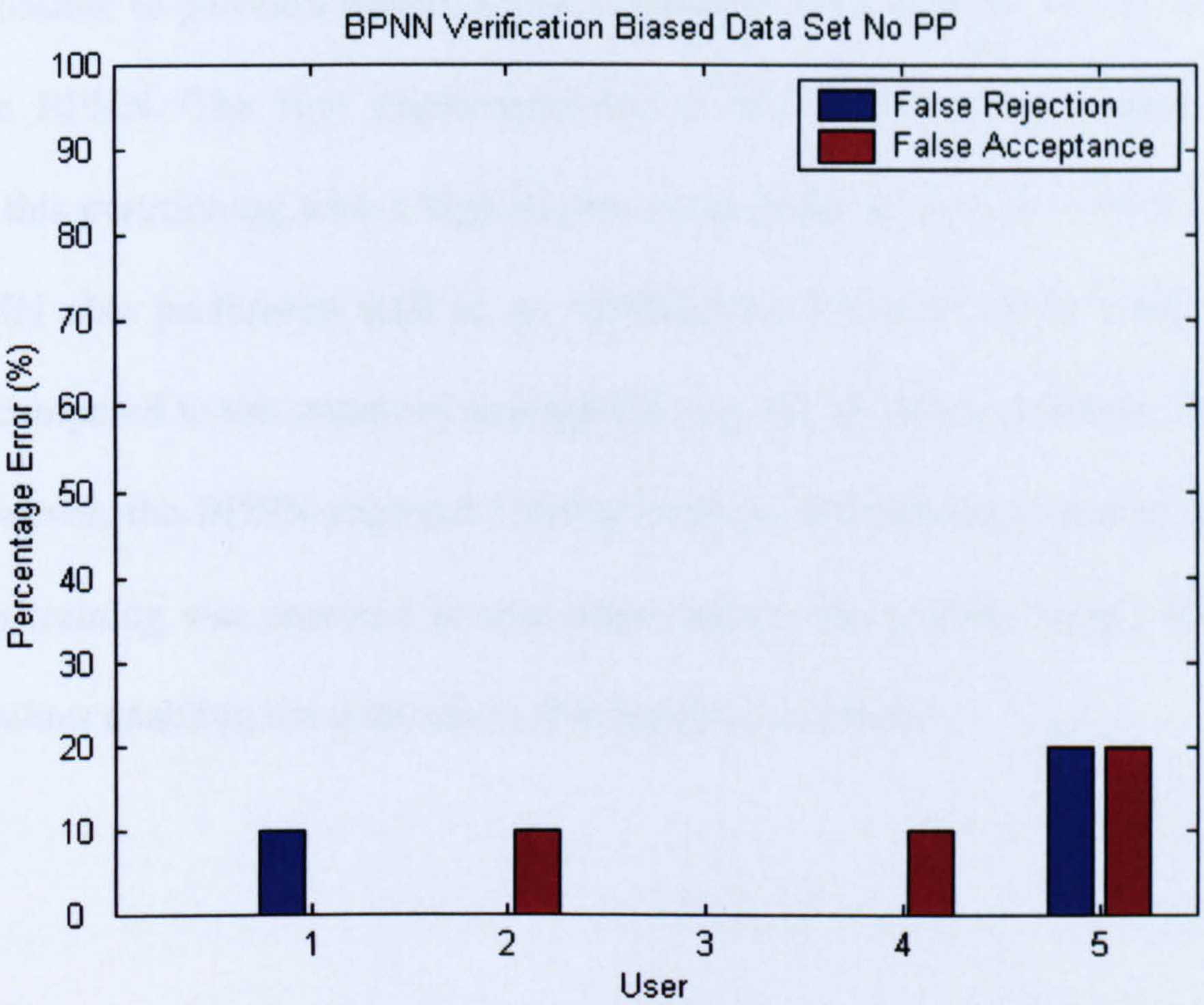


Figure 5.9 Results for Biased Test Data for BPNN

User	1	2	3	4	5	**Mean**
FR (%)	0.10	0	0	0	0.20	**0.06**
FA (%)	0	0.10	0	0.10	0.20	**0.08**

Table 8 - Table of results for BPNN biased no PP

5.4 Summary

In this chapter the BPNN, a traditional method for non-linear pattern recognition tasks, has been applied to keystroke dynamic data. The network was trained in identification mode to demonstrate the level of separation amongst users that raw keystroke dynamic data can provide. The BPNN was evaluated in a verification role with the two sets of data and pre-processing. We can see from these results that it is clearly possible to partition pattern space to classify users on their keystroke behaviour using the BPNN. The first implementation of the BPNN in an identification role achieves this partitioning with a high degree of accuracy for almost 90% of the test set. The BPNN also performed well in an verification role with errors which are much reduced compared to the statistical average FR and FA of 10% and 6% for the unbiased data. However, the BPNN required training over several thousand epochs and in most cases this training was repeated several times before the random weight initialisations were at values enabling the network to find a global minimum.

Chapter 6

Probabilistic Neural Network

6.1 Introduction

The BPNN has been shown to posses the ability to map decision boundaries around keystroke dynamic data. However, the BPNN suffers from well known problems during training such as getting stuck in local minima. Also, the many parameters that the BPNN is dependant on, and highly sensitive to, such as number of hidden layers/nodes and their associated transfer functions and learning rates cannot be set analytically. Optimising the BPNN becomes a case of trial and error. The structure of the BPNN requires that training is carried out on the whole of the data set when data is added or removed. This could be time consuming in a keystroke dynamic verification system when a user needs to enrol or leave. In order to address these problems a neural network was sought that might be more applicable to keystroke dynamics.

The PNN classifier is an effective tool in pattern recognition as it has evolved from well established Bayesian statistical techniques [23]. It is closely related to radial-basis function neural networks [23]. The following advantages when compared to backpropagation make it a more suitable candidate for keystroke dynamics:

- Rapid Training: The PNN can train much faster than the BPNN [23]. It merely needs to 'read' the data.

- The PNN is guaranteed to converge to an optimal result given sufficient data [23].

This contrasts with the unpredictable nature of the BPNN which suffers from the problem of local minima.

- Data can be added and deleted without re-training. Useful for adding new users to the system and removing old ones.

- The PNN gives a measure of confidence associated with an output, as the output node values are a combination of the similarity measures from the middle layer [23].

However, the network produced by the PNN contains many more nodes and is computationally more intensive than the BPNN, a fact that is offset somewhat by its parallel nature [23]. Due to the parallel nature of the PNN, it is a candidate for multi-processor systems. The PNN still retains many of the features associated with BPNN such as learning and generalisation, but these are performed in a more stable and predictable manner.

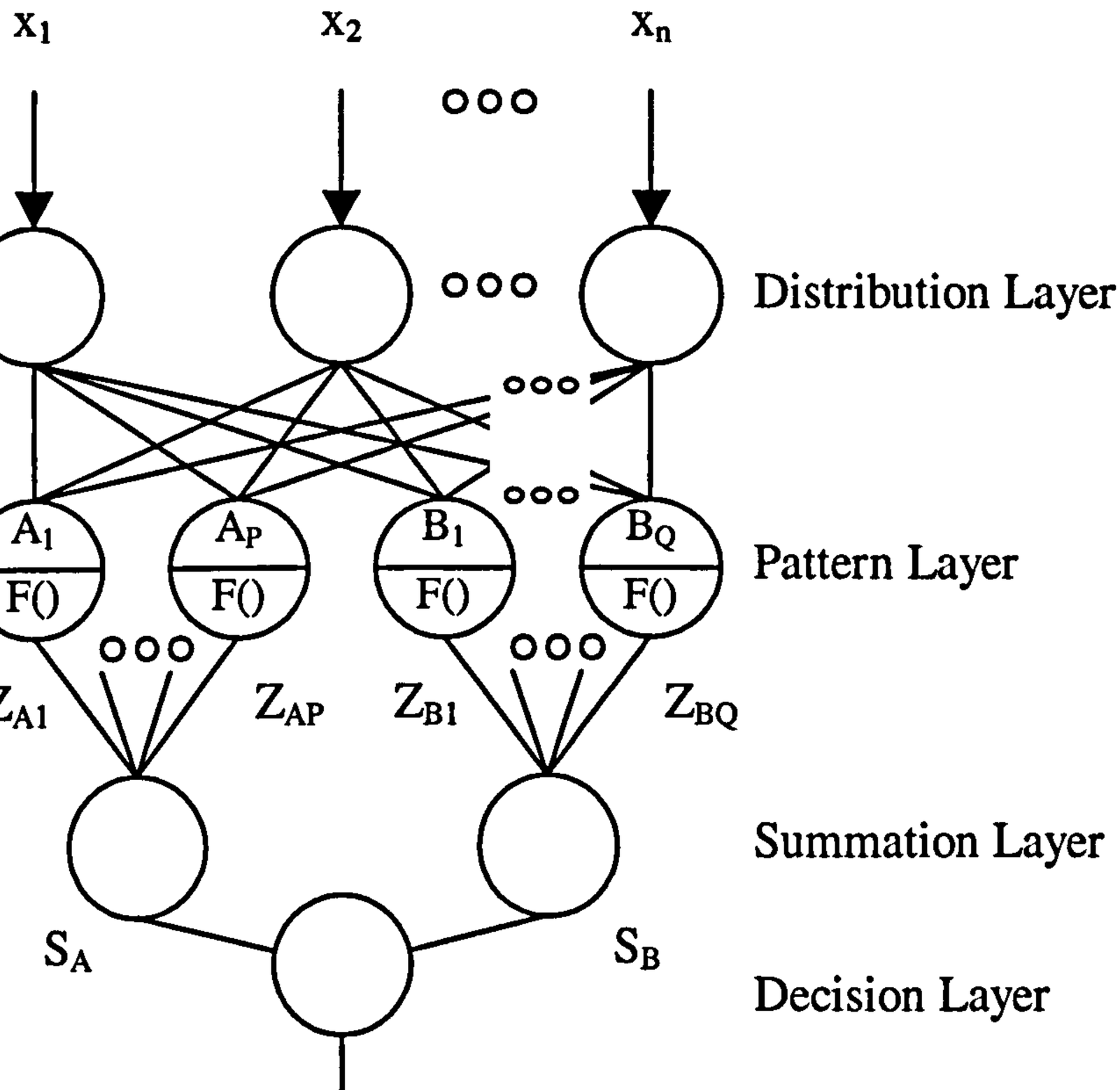


Figure 6.1 Structure of PNN

The structure of the PNN for a 2 class problem can be seen in Figure 6.1 [23]. It consists of 4 layers:

1. The Distribution Layer: This performs no computation and just connects the inputs to the next layer. The inputs to the neural network [$X_1 \rightarrow X_N$] were mapped to the time intervals for each person's sample. For example, in the unbiased data set the time interval between u & s in the phrase 'username' was mapped to X_1, the interval between s & e to X_2, etc... Hence, the number of nodes (16) in the distribution layer represents the number of time intervals in the input vector.

2. The Pattern Layer: A node is created in the pattern layer for each training sample. The inputs are passed through the nodes weights (A_F/B_F) where A and B represent weights generated by each class during training. This is then passes through a Gaussian function F(). This gaussian function is centred around zero, outputting a one if the input to it is zero. This provides the node with a higher activation for input patterns that have a small vector distance with the nodes weights.

3. The Summation Layer: The outputs of each pattern layer node are summed.

4. The Decision Layer: This picks the largest node in the summation layer, outputting a 1 for $S_a > S_B$ and a 0 otherwise.

Problems with more than 2 classes are easily dealt with by a competitive algorithm in the decision layer to pick the summation node with the highest value.

The outputs of the pattern layer nodes, are given by:

$$Z_{ci} = \exp[(X^t \bullet X_{Ri} - 1)/\sigma^2] \qquad (6.1)$$

The outputs of the summation layer nodes, are given by:

$$S_c = \sum_{i=1}^{P} \exp[(X^t \bullet X_{Ri} - 1)/\sigma^2] \qquad (6.2)$$

Where the subscript c indicates the class to which the node belongs (for example, in the two class problem in Figure 6.1, c would indicate a node belonging to either class A or class B). The subscript i identifies the training sample associated with the node in the pattern layer. The value P represents the number of nodes for each class in the pattern layer, σ^2 being the standard deviation. The transpose of the unknown vector to classify is X^t and the weights are X_{Ri} (R denoting that the vector is a 'reference' or 'training' vector, i representing the training sample associated with the node). The network operates by calculating the dot product between the unknown vector and the training vectors. However, this uses normalised training and testing vectors. Normalisation of data for this problem results in loss of information concerning the total typing duration [13]. In order to overcome the problem with normalisation, the dot product term may be replaced by a distance measure. Two distance measures which can be applied to this problem are the Euclidean Distance and the Manhattan Distance. These have already been described in chapter 4, however, their performance within the PNN is not known so an evaluation of their application to the PNN is required.

Whilst the PNN does not suffer from the trial and error approach needed for BPNNs one important variable needs to be set. The spread function determines the extent to which the network will generalise. A good description of this behaviour may be found in [23], where the effects of applying small, medium, and large spread constants on the generalisation of the network are explored. Although it is difficult to determine this spread constant objectively, its effect on results is not nearly as acute as the parameters for the BPNN. At the limit, an extremely small spread function will make the PNN behave as a 'nearest-neighbour' classifier. An extremely large spread

function will create many overlapping boundaries and result in misclassifications. The spread value was set to an empirically derived value which gave consistently good results under experimentation.

6.2 Verification and PNN

6.2.1 Unbiased Data

Results for the PNN during training produced zero FA and FR errors. Therefore, only results obtained from the training set will be examined. The first implementation of the PNN used the Euclidean distance function with no data pre-processing. It can be seen from Figure 6.2 that the FR error rate is fairly high and user 19 has completely failed to be accepted by the system. However, FA rates are low, in some cases zero. The sensitivity of the Euclidean distance metric to the outlier data within the samples is the main contributor to the high false acceptance error. Consistent users, such as user 21, perform fairly well, but inconsistent users perform badly with this set-up.

Performance for the FR error is improved when the data is pre-processed before being submitted to the PNN as can be seen in Figure 6.3. There is a trade off, however, as the FA rate has increased due to the pre-processing. User 19 is now accepted by the system on two occasions, lowering the previous FR error from 100% to 60%. This is still a considerable error, even on this difficult set of data.

Applying the PNN using the Manhattan distance function and no pre-processing Figure 6.4 it is clear to see there is a higher FR error evident than that occurring in the BPNN case Figure 5.6. The FA error with the BPNN is considerably higher for users 4 and 20, this may be due to the BPNN's tendency to over generalise to accommodate a user exhibiting an inconsistent typing style. The PNN has the ability to generalise, but

in a more controlled manner than that of the BPNN, therefore it is less likely to respond to an inconsistent user with a high FA error.

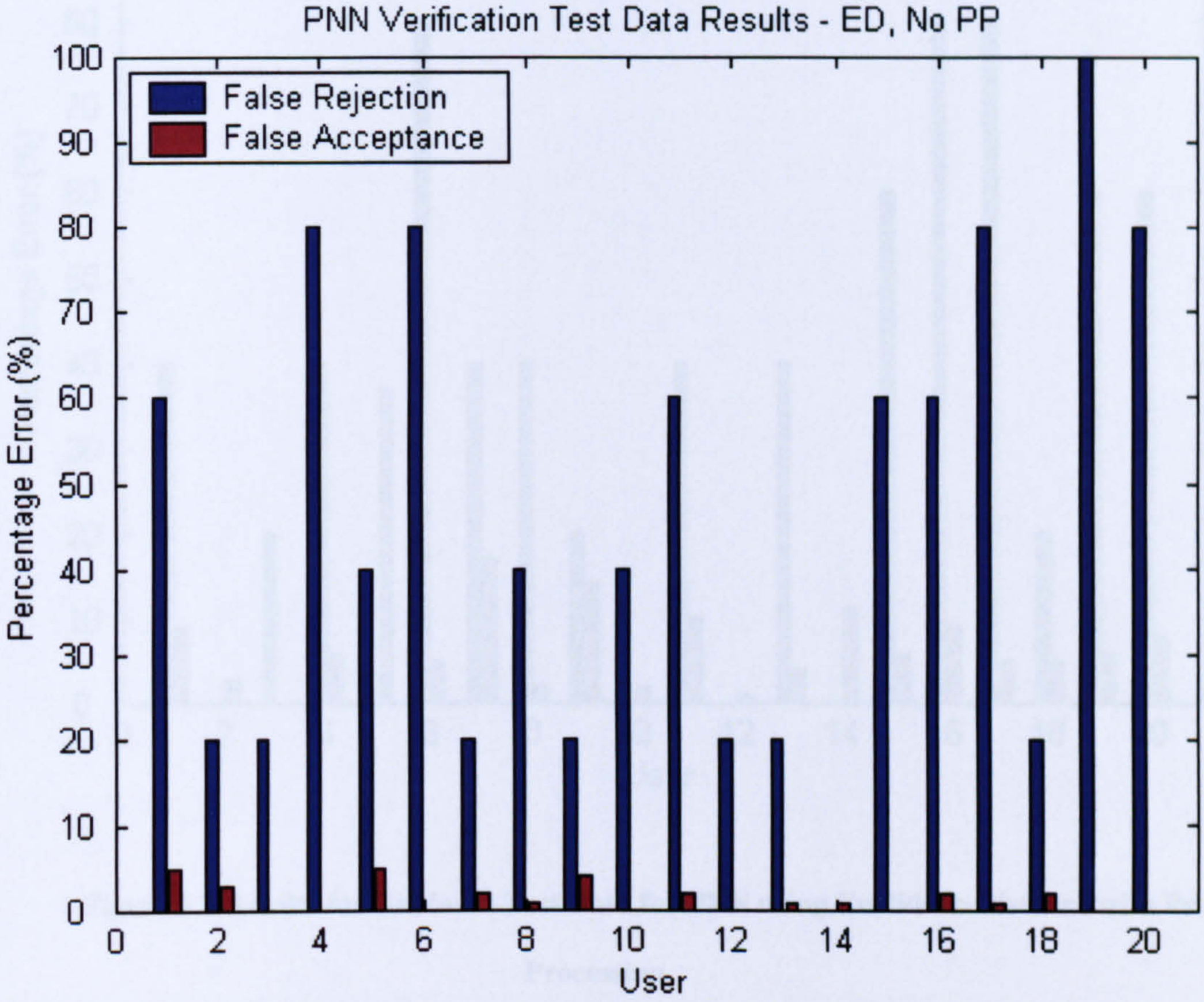


Figure 6.2 Results for Unbiased Test Data for PNN using Euclidean Distance

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0.60	0.20	0.20	0.80	0.40	0.80	0.20	0.40	0.20	0.40
FA (%)	0.05	0.03	0.00	0.00	0.05	0.00	0.02	0.01	0.04	0.00

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0.60	0.20	0.20	0.00	0.60	0.60	0.80	0.20	1.00	0.80	0.00	**0.44**
FA (%)	0.02	0.00	0.01	0.01	0.00	0.02	0.01	0.02	0.00	0.00	0.03	**0.02**

Table 9 - Table of Results for Unbiased Test Data for PNN using Euclidean Distance

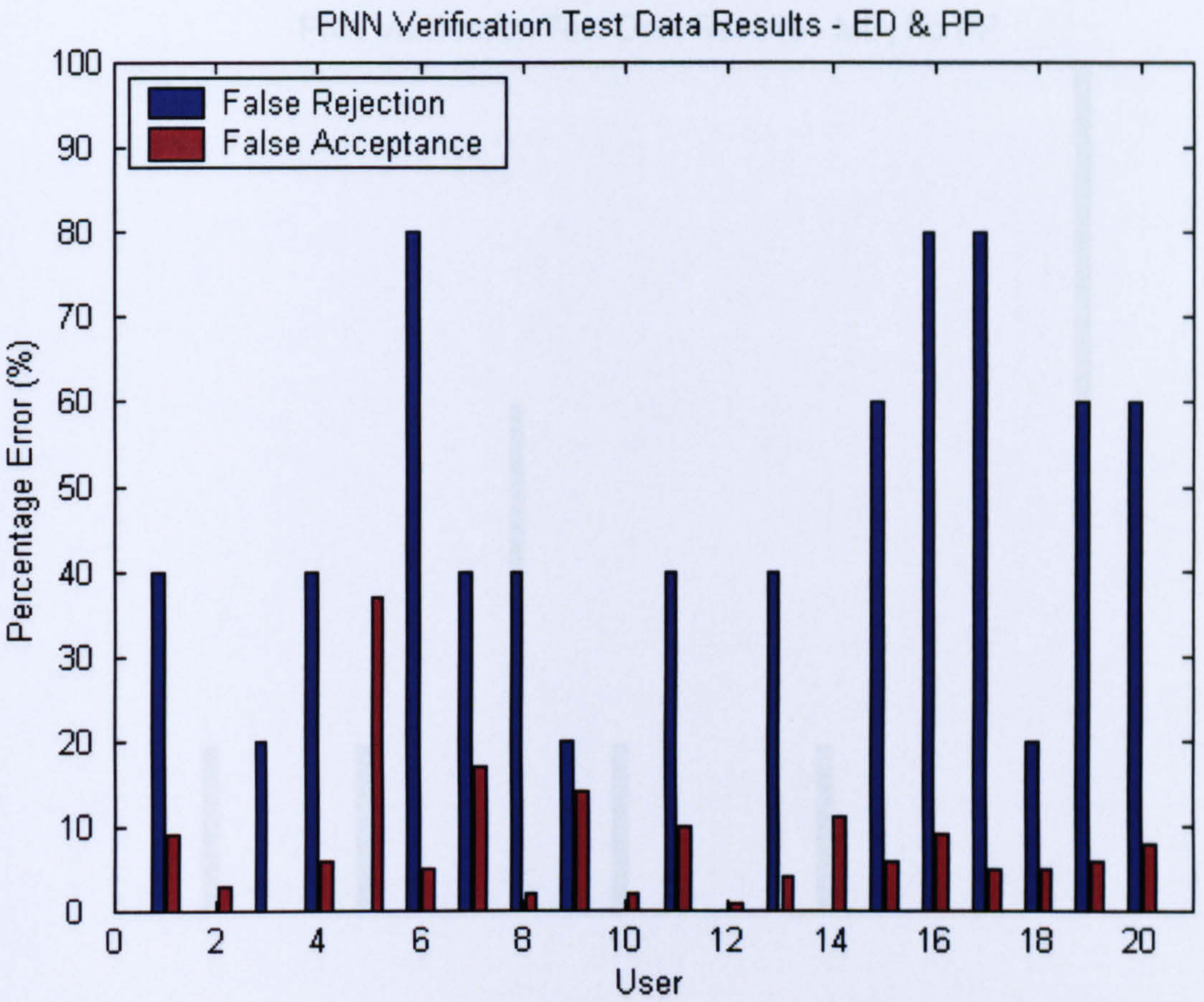


Figure 6.3 Results for Unbiased Test Data for PNN using Euclidean Distance with Pre-Processing

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0.40	0.00	0.20	0.40	0.00	0.80	0.40	0.40	0.20	0.00
FA (%)	0.09	0.03	0.00	0.06	0.37	0.05	0.17	0.02	0.14	0.02

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0.40	0.00	0.40	0.00	0.60	0.80	0.80	0.20	0.60	0.60	0.00	0.34
FA (%)	0.10	0.01	0.04	0.11	0.06	0.09	0.05	0.05	0.06	0.08	0.01	0.08

Table 10 - Table of Results for Unbiased Test Data for PNN using Euclidean Distance with Pre-Processing

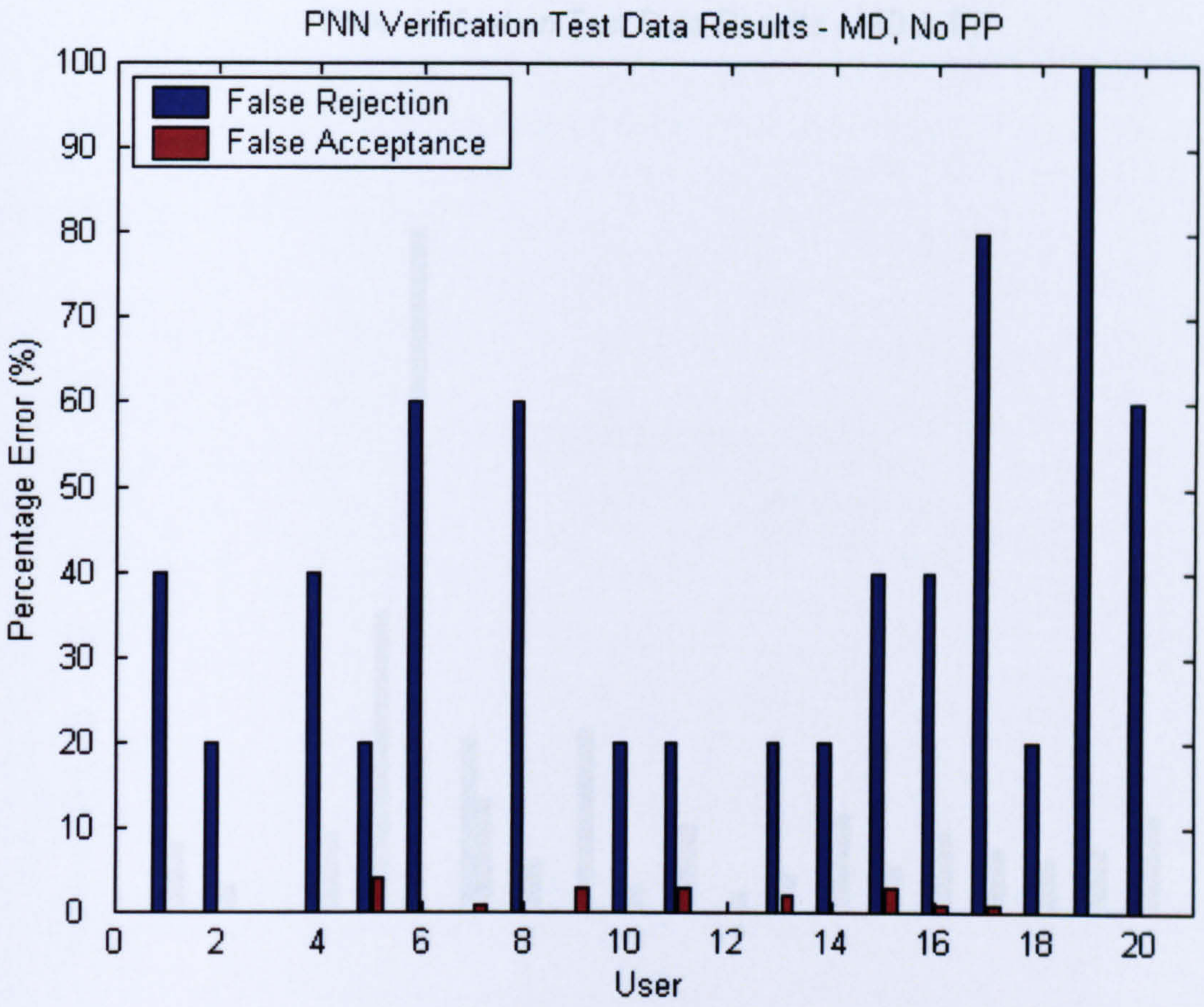


Figure 6.4 Results for Unbiased Test Data for PNN using Manhattan Distance

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0.40	0.20	0.00	0.40	0.20	0.60	0.00	0.60	0.00	0.20
FA (%)	0.00	0.00	0.00	0.00	0.04	0.00	0.01	0.00	0.03	0.00

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0.20	0.00	0.20	0.20	0.40	0.40	0.80	0.20	1.00	0.60	0.00	0.31
FA (%)	0.03	0.00	0.02	0.00	0.03	0.01	0.01	0.00	0.00	0.00	0.00	0.01

Table 11 - Table of Results for Unbiased Test Data for PNN using Manhattan Distance

Pre-processing the data before submission to the PNN with the Manhattan distance function leads to an overall reduction in FR error except for users 6 and 17. An overall increase in FA is observed as expected with the pre-processing.

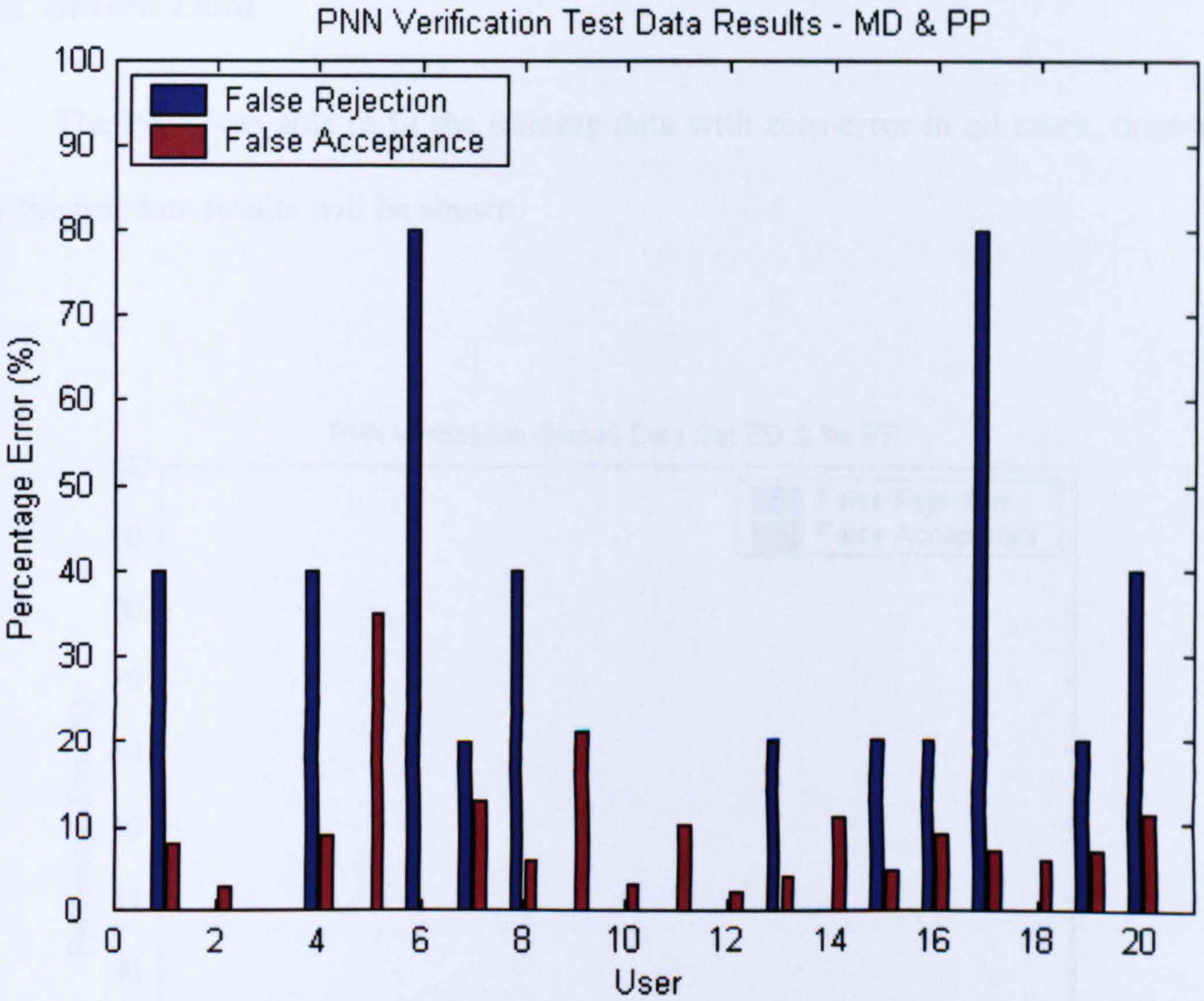


Figure 6.5 Results for Unbiased Test Data for PNN using Manhattan Distance with Pre-Processing

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0.40	0.00	0.00	0.40	0.00	0.80	0.20	0.40	0.00	0.00
FA (%)	0.08	0.03	0.00	0.09	0.35	0.00	0.13	0.06	0.21	0.03

User	11	12	13	14	15	16	17	18	19	20	21	Mean
FR (%)	0.00	0.00	0.20	0.00	0.20	0.20	0.80	0.00	0.20	0.40	0.00	0.20
FA (%)	0.10	0.02	0.04	0.11	0.05	0.09	0.07	0.06	0.07	0.11	0.02	0.08

Table 12 - Table of Results for Unbiased Test Data for PNN using Manhattan Distance with Pre-Processing

6.2.2 Biased Data

The PNN was able to fit the training data with zero error in all cases, therefore, only the test data results will be shown.

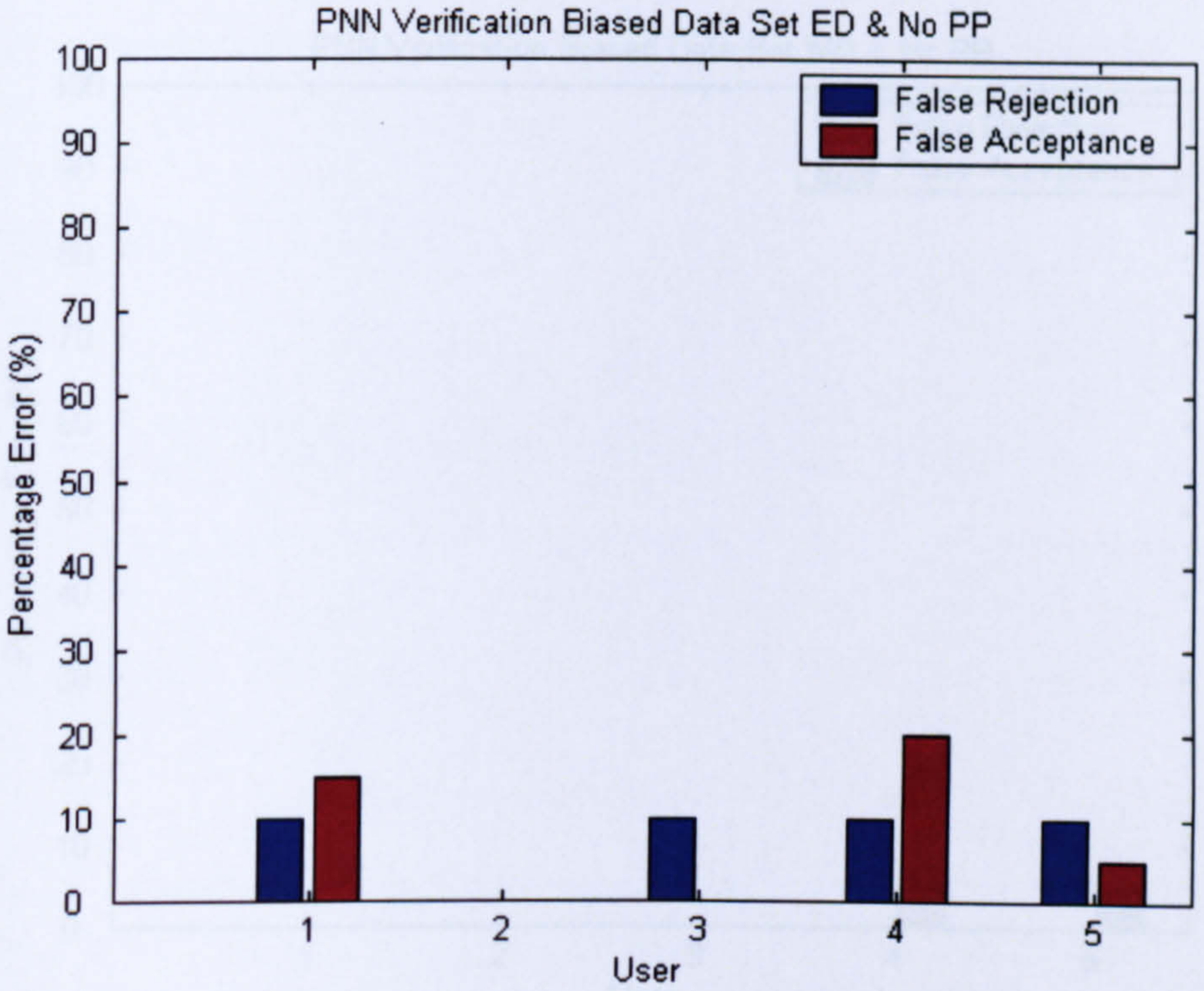


Figure 6.6 Results for Biased Test Data for PNN using Euclidean Distance

User	1	2	3	4	5	**Mean**
FR (%)	0.10	0.00	0.10	0.10	0.10	**0.08**
FA (%)	0.15	0.00	0.00	0.20	0.05	**0.08**

Table 13 - Table of Results for Biased Test Data for PNN using Euclidean Distance

Again the biased data set generates better results than the unbiased. Here the results for the PNN operating with the Euclidean distance function can be seen in Figure 6.6. User 2 generates no errors, whilst errors across the other users are consistently low.

The results for the Manhattan Distance function are an improvement over the Euclidean Distance. It can be seen in Figure 6.7 that FR errors are zero for all users. The only errors occurring are a small 5% FA rate with users 4 and 5. This clearly shows the improvement in performance which is gained by using data which is biased towards a known user and the suitability of the PNN to this task.

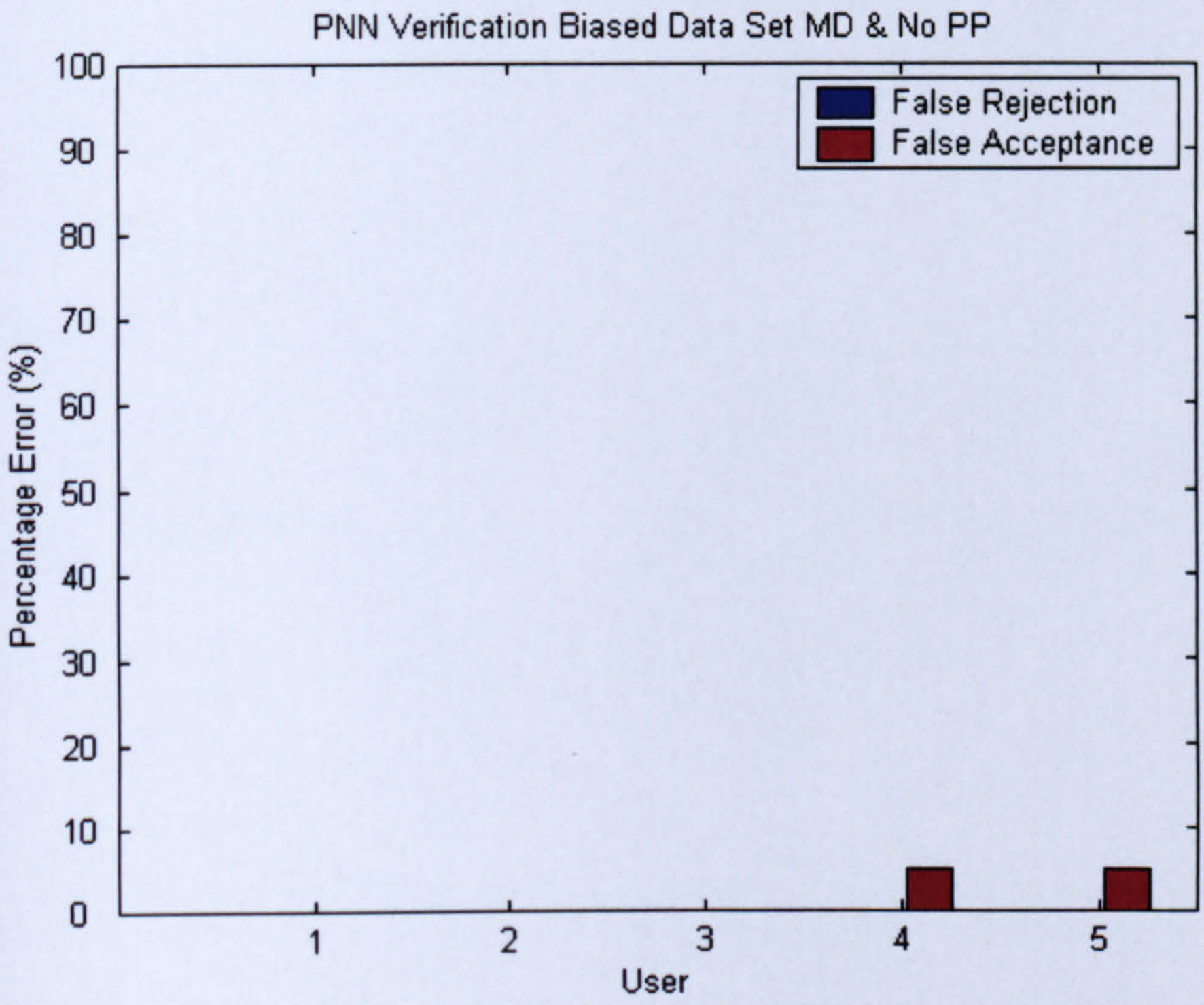


Figure 6.7 Results for Biased Test Data for PNN using Manhattan Distance

User	1	2	3	4	5	Mean
FR (%)	0.00	0.00	0.00	0.00	0.00	**0.00**
FA (%)	0.00	0.00	0.00	0.05	0.05	**0.02**

Table 14 - Table of Results for Biased Test Data for PNN using Manhattan Distance

6.3 Summary

In this chapter the PNN has been shown to be an effective tool in the verification

of users using keystroke dynamics. The performance has been demonstrated with the use of data pre-processing to reduce the FR error rate. The PNN has shown an improvement in performance with the Manhattan Distance function over the Euclidean Distance function.

Chapter 7

Cascade Forward Neural Network

7.1 Introduction

The previous pattern recognition techniques (Statistical, PNN and BPNN) work by comparing each time interval as a separate entity with its counterpart in the training data. Such that, if all the time intervals in the data set correspond to the correct typing intervals, the order in which they are presented to the classifier does not matter. A novel approach to the classification of keystroke dynamics is to consider the time intervals generated during acquisition as a sequence. As the depression of keys occurs successively, they can be viewed as sequenced events. Therefore, following measurements can be expected to be highly correlated with each other. This approach has the benefit of considering any additional information present in the *order* of the samples as well as the actual time intervals. However, this approach requires the use of dynamic neural networks, which can be considerably more difficult to analyse, model, implement and fine-tune than their static counterparts.

Dynamic networks are more complex to model due to their recurrent topologies [29]. The structure of these networks incorporates feedback. This feedback can come from the output of the system, or the output of any number of intermediate nodes, and is fed back to the input or internal nodes. In this way, the present input to the system is affected by the previous system inputs. This gives the system a form of memory. Static

systems can also be said to possess a form of memory, but there are differences between this memorizing behaviour in static and dynamic systems. These differences are distinguished as long-term and short-term memory. Static networks are said to contain long-term memory as they have a repository of past information. However, when this information is stored in the network weights, the knowledge of the time when the information was acquired is lost. Static networks are therefore unable to differentiate the relationships of time that might be present in data, as all information is compressed together in the network weights. Dynamic networks are said to contain short-term memory. This is due to the recurrent connections allowing the recording of timing information, so the network becomes sensitive to the order in which the information is presented. Dynamic networks also have a long-term memory which is stored in the connecting node weights but, unlike static networks, these weights record differences within the time window of observation applied to the data.

There are many different types of dynamic neural network [28]. This study concentrates on a network architecture that is a variant of the BPNN, the Cascade Forward Neural Network (CFNN) [28]. A Cascade Forward Neural Network is identical to a feed-forward BPNN except that each layer after the first has weights coming not just from the previous layer but also from all other previous layers. We used the Levenberg-Marquardt [29] training algorithm to accelerate training in this case. The input to the network was generated by sliding a window over the time interval data, removing the window's central element at each step. The central element that was removed from the window was then used as the training target for the networks single output. The network is only trained on the data from the correct user, with the hypothesis that any subsequent unseen data from this user will exhibit similar correlations and cause the output of the network to behave in a similar manner. If the network is presented with data from an impostor, then the data correlation with the

correct user will be uncorrelated, causing the output of the network to behave in a different manner, with large variations occurring compared with the correct user. Therefore, the role of the network in this guise is to act as a function approximater and not as a classifier.

Verification is determined by a collective measure of errors between outputs and targets for all the window positions. In this case we use the mean square error. This error criterion will give an indication of the identity of the individual. A low error will be exhibited by the correct user and a high error by an impostor. This requires that a threshold level is determined over which the user is deemed to be an impostor. This level may be set depending on the level of security required, such as the case in the statistical methods. As the main point of interest in this chapter is the training of the network we shall present all the results with the optimum threshold level selected, such that both FA and FR errors are minimised.

7.2 Verification and CFNN

7.2.1 Unbiased Data

The network was trained firstly on the unbiased data. A minor difference compared with the other methods is the number of users is 20, as user 21's data was not available at the time of this experiment. The data for the training set consisted of 10 samples from each user. The network was trained until the mean square error was minimised. Results for training are not shown as the network was able to train successfully for every user.

Test data for the network consisted of five samples from each user. The results for the test data can be seen in Figure 7.1 and Table 15. Results for the CFNN on the

unbiased data show the variation in performance that can again be attributed to the variations in the consistency of the user's typing. This is clearly shown in Figure 7.1, where user 7 (a consistent user) displays a FR error of zero and a FA error of 2% (Table 15). This can be compared with the performance of user 20 (an inconsistent user), displaying a FR error of 80% and a FA error of 28%.

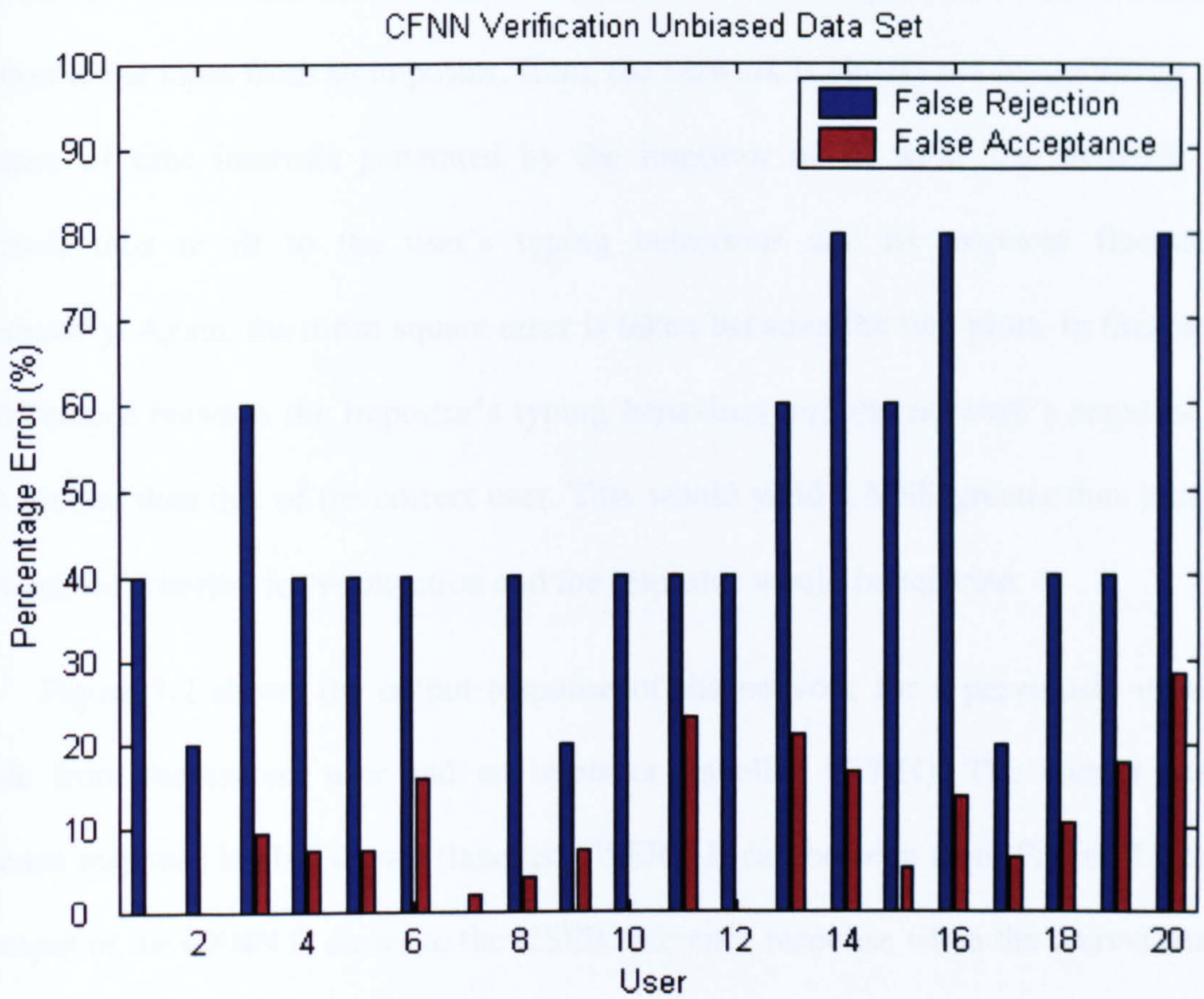


Figure 7.1 Results for CFNN verification with Unbiased Data

The general performance of the CFNN is less severe than that of user 20. The average error over the twenty users for FR is 44% and 10% for FA.

An example of the response of the network to a correct user and an impostor can be seen in Figure 7.2. The plots show the output of the network attempting to interpolate data produced by the legitimate user and the impostor. The input data is a 14 character string and a window size of 7 was chosen. This gives 13 time measurements between keystroke of which, all but the last and first 3 can be compared to the network's output

as the window's central element slides over them. Here the network's output can be seen in blue, labelled CFNN, and the user's input in red. In the first plot, the response of the network to the correct user is shown. It can be seen that the network is able to interpolate to a fair accuracy the behaviour of the user. The mean square error between these two plots would then be used as the criteron for verification, with the user being accepted for values under a certain threshold level. The second plot shows the network's response to the input from an impostor. Here, the network is clearly not interpolating the sequence of time intervals generated by the impostor at all well. The network, as expected, fails to fit to the user's typing behaviour and its response fluctuates considerably. Again, the mean square error is taken between the two plots. In this case, the difference between the impostor's typing behaviour and the network's response is much greater than that of the correct user. This would yield a MSE greater than that of the threshold criterion for verification and the impostor would be rejected.

Figure 7.2 shows the output response of the network for a previously unseen sample from the correct user and an impostor (labelled CFNN). The correct users reference response is also shown (labelled USER). It can be seen from Figure 7.2 that the output of the CFNN is closer to the USER reference response when the correct users sample is presented than with the impostors. This show that correct verification is possible, even if the accuracy of the network response to the correct user is not that high. The main factor in the decision is the difference between the correct user's network response and the impostor's network response. As long as the impostor generates a response which has a much higher MSE, the correct decision will be made. This will also make the setting of a threshold value easier, as the large separation between the correct user and the impostor will allow greater freedom in the placement of the threshold level.

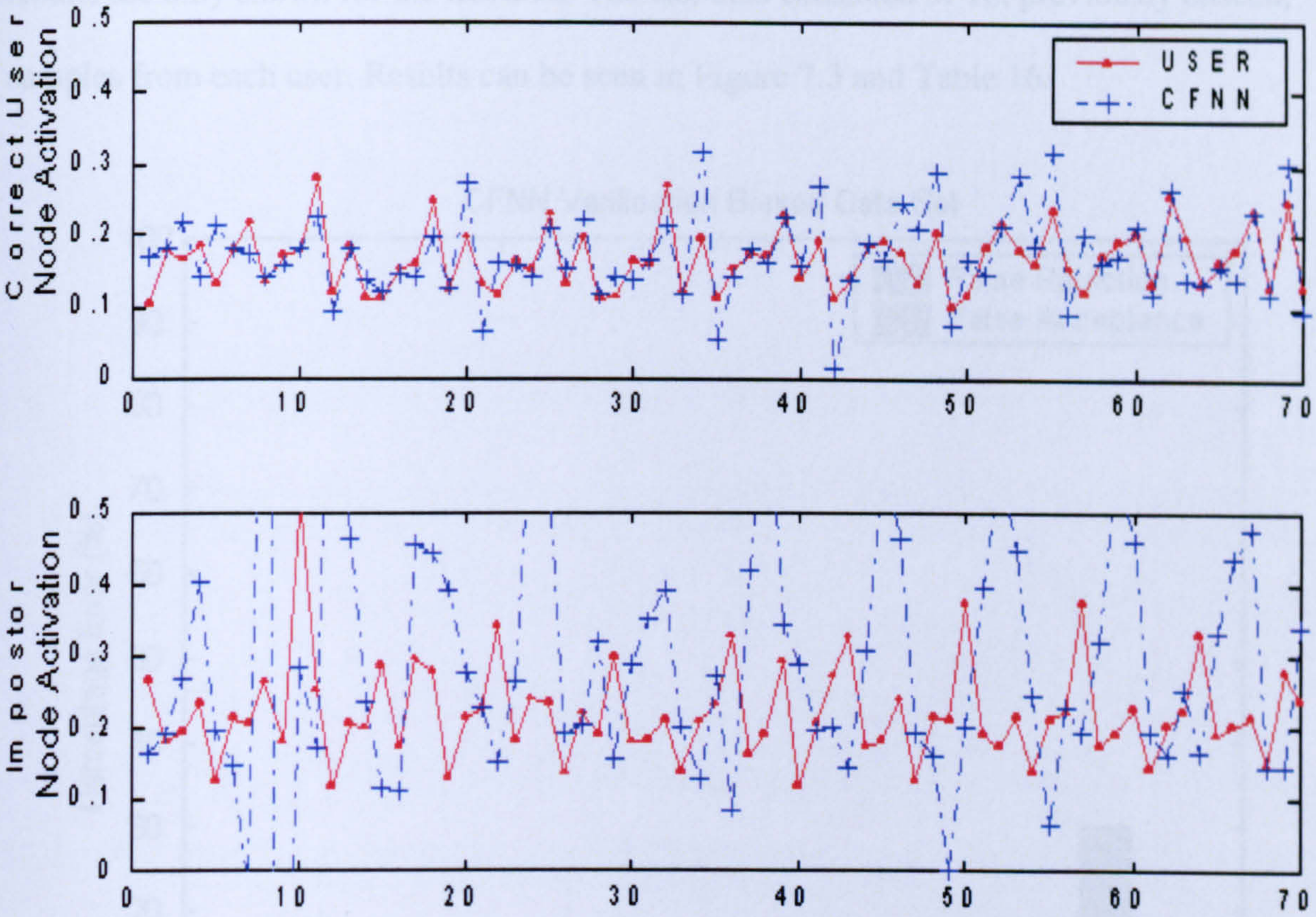


Figure 7.2 Example of CFNN output for the Correct User and an Impostor

User	1	2	3	4	5	6	7	8	9	10
FR (%)	0.40	0.20	0.60	0.40	0.40	0.40	0.00	0.40	0.20	0.40
FA (%)	0.00	0.00	0.09	0.07	0.07	0.16	0.02	0.04	0.07	0.00

User	11	12	13	14	15	16	17	18	19	20	Mean
FR (%)	0.40	0.40	0.60	0.80	0.60	0.80	0.20	0.40	0.40	0.80	**0.44**
FA (%)	0.23	0.00	0.21	0.17	0.05	0.14	0.06	0.11	0.18	0.28	**0.10**

Table 15 - Table of Results for CFNN with unbiased data

7.2.2 Biased Data

The second data set was then applied to the CFNN. Again, 10 samples were used in the training of the network. The training was successful for every user, therefore,

results are only shown for the test data. The test data consisted of 10, previously unseen, samples from each user. Results can be seen in Figure 7.3 and Table 16.

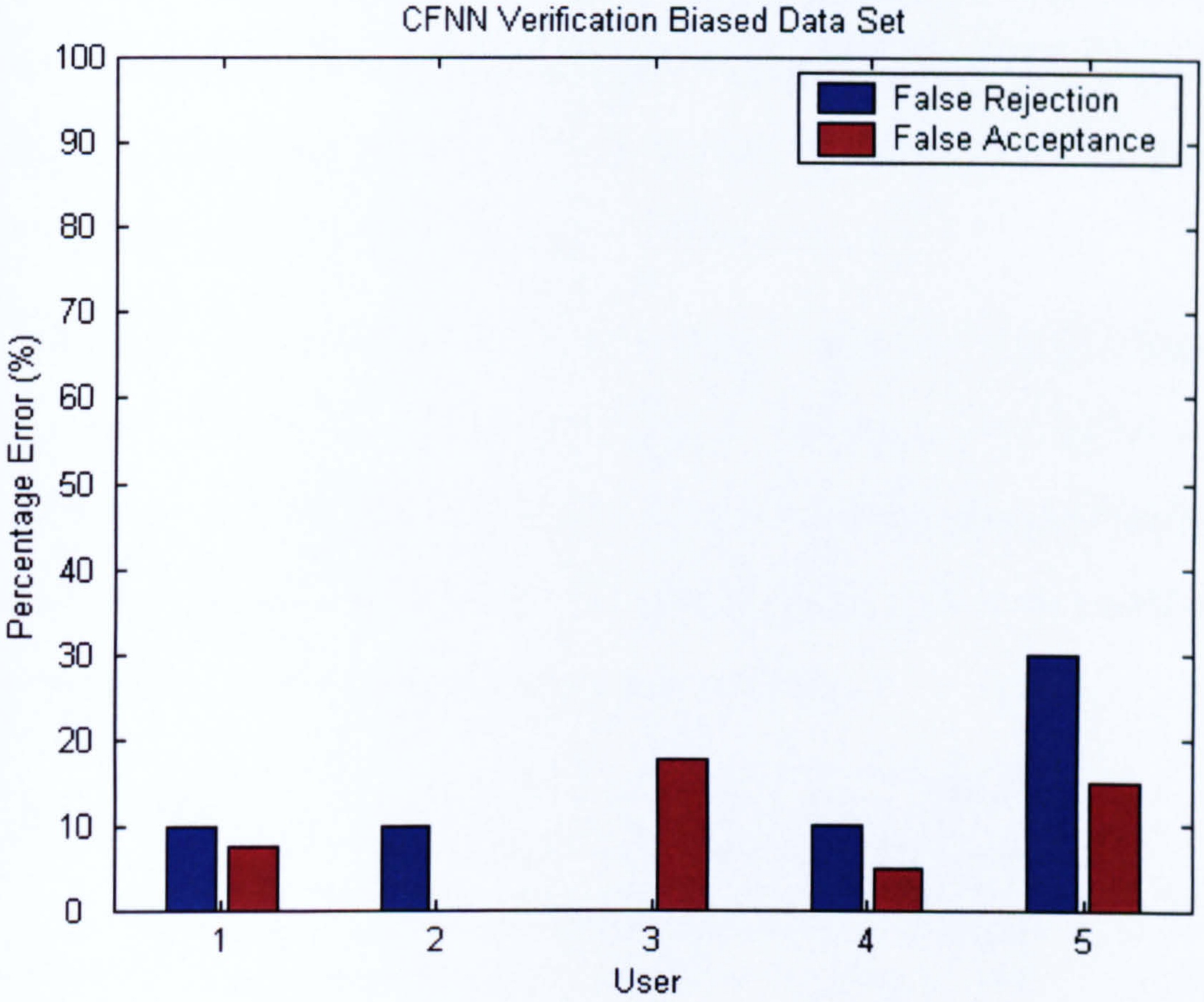


Figure 7.3 Results for CFNN Verification with Biased Data

The biased data clearly yields better results. The biggest difference can be seen in the FR error rate, this has decreased by 32% (Table 16). The reduction in the error rate for FR can be attributed to the more consistent nature of the biased data. As each user is typing their name, a common phrase to them, they exhibit less variance in their behaviour. The error rate for FA, however, has only decreased by 1% (Table 16). The reason for the smaller reduction in this case is again related to the consistency of the biased data set. The impostors are typing other people's names, which are not phrases that they are familiar with. As this is no different to the previous data set, the benefit of

a more consistent typing style amongst correct users has less of an impact on the FA error than it does on the FR error. However, the small decrease in the FA error is due to the increased consistency amongst the correct users. This increased consistency allows a much tighter boundary to be set around the correct user's data, whilst still maintaining a low FR error. The smaller the area which defines the correct users class is, the less likely it is that an impostor will be mistaken for the correct user.

The ability of the CFNN to produce a distinct separation boundary between legitimate users and impostors with only data from the legitimate user is a feature which is necessary for the application of a keystroke security system. The other methods presented have required data from impostors during training. This is an undesirable requirement as this data has to be obtained or simulated [24].

User	1	2	3	4	5	Mean
FR (%)	0.10	0.10	0	0.10	0.30	**0.12**
FA (%)	0.08	0	0.18	0.05	0.15	**0.09**

Table 16 - Table of Results for CFNN Verification with Biased Data

The table in Table 17 draws together the mean results over the two data sets for the methods implemented in the previous chapters. The poorest performing methods are the statistically based ones. In the unbiased data set the rate of FR for both statistical methods is comparable to the neural network techniques, but the FA error is much worse. To reduce the FA error to a level comparable with the neural network based techniques would increase the FR error to unacceptable levels. The statistical methods fare better with the biased data, but are still the poorest performers. The simplistic approach to classifying keystroke dynamics with a vector distance measure and threshold level is clearly not effective enough to be of practical use.

7.3 Comparison of Methods

	Unbiased Data		Biased Data	
Method	**Mean FR**	**Mean FA**	**Mean FR**	**Mean FA**
Statistical: Euclidean Distance	0.30	0.32	0.12	0.09
Statistical: Manhattan Distance	0.22	0.24	0.17	0.19
BPNN: Verification no PP	0.10	0.06	0.06	0.08
BPNN: Verification with PP	0.08	0.13	-	-
PNN: Euclidean Distance no PP	0.44	0.02	0.08	0.08
PNN: Euclidean Distance with PP	0.34	0.08	-	-
PNN: Manhattan Distance no PP	0.31	0.01	0.00	0.02
PNN: Manhattan Distance with PP	0.20	0.08	-	-
CFNN	0.44	0.10	0.12	0.09

Table 17 - Comparison Chart

The BPNN produces good results. Its training algorithm lets it adapt to the complex pattern space classification boundaries. It is interesting to note that the performance gain in the use of data pre-processing is not as great with the BPNN as it is with the PNN. The improvement in the PNN for pre-processed data is about 10%, whilst in the BPNN it is 2%. This can be explained by the BPNN's adaptive behaviour. During training the weights on the network may be altered such that the inputs with a high inconsistency are given lower weighting. In this way it is possible for the BPNN to learn so that it incorporates some data pre-processing within the network itself. The more ridgid form of the PNN prevents this 'automatic' data-filtering effect from occurring, hence the pre-processed data shows a larger improvement in performance.

However, the BPNN has significant drawbacks. The training of the BPNN was effectively a trial-and-error procedure. Choosing the number of hidden nodes, the learning rates and initial network weights was a non-trivial task. The network had to be trained several times with varying parameters in order for a satisfactory result to be achieved. The training itself was a drawn out process with the network taking several thousand epochs to come to a solution. Although the BPNN ultimately gave good results, this difficulty in its unpredictable training must be taken into account when comparing with the other methods. Another problem with the BPNN, which makes it unsuitable for this application, is its unpredictable behaviour once trained. Analysing the structure of a neural network is extremely difficult. Often the only method for testing the reliability of decisions made by the network is to thoroughly test it with thousands of samples. Even then, the reliability of the system cannot be 100% guaranteed, this makes such neural networks unsuitable for mission critical tasks. The extent to which this becomes a problem will depend largely on the final application. But for applications where security is important, the BPNN will not be acceptable as a measure of confidence in its decisions is not available. What it does hint at, especially in the identification role, is the extent of the uniqueness which keystroke data contains.

The PNN displays a large difference in performance between the Euclidean and Manhattan distance measures. The gain in performance for the Manhattan distance is around 10% in both cases. Its performance is clearly the best when applied to the biased data set. Here is outshines the other methods as it is the only one which has an FR error of zero coupled with the lowest FA rate of just 2%. This beats the BPNN by 6% in both FA and FR, showing its potential as a candidate for a final application. Apart from its obvious performance benefits, it also has some functional benefits over the BPNN. Its structured topology is much easier to analyse than the BPNN. The network is based on robust Bayesian theory and its output is consistent. This makes the problem of

measuring the confidence of decisions an easier task. Training is another performance gain with the PNN, it merely needs to 'read' the training data. As this procedure doesn't rely on randomised initial weights, the performance of the network is consistent every time. The network also has a parallel structure. This would make it possible to add or remove users from the system without the need to retrain. The BPNN would require the intensive retraining process to be carried out on every change of user. The disadvantage of the PNN's parallel structure is, as is the case with most radial basis networks, a larger number of nodes than the BPNN. Therefore, it is more computationally intensive in operation. Its parallel structure does enable the use of parallel processing if increased speed were required. So this problem may be overcome by the addition of more hardware. The PNN is also easier to set parameters for than the BPNN. Once the distance measure function has been chosen there is only one more parameter that needs to be set. This is the spread value and controls the spread of the gaussian function in the middle layer. This parameter is easily adjusted as variability of its value has only a small effect on the operation of the network.

The CFNN performance is considerably worse than the other methods on the unbiased data and tolerable on the biased data. However, its performance is achieved without the prior knowledge of the impostors typing behaviour. This is a desirable attribute for a final application. The network is very time sensitive to the keystroke data. It can fit to the correct users typing pattern, but only with a considerable amount of training. However, this difficulty in fitting to a user's data can make it very strong against impostors. This time sensitivity can make the network unstable when inconsistent data is applied. This is the reason for the large 34% difference in the FR error rate between the unbiased and biased data set. The more inconsistent nature of the unbiased data is a difficult task for the CFNN. The increased consistency of the biased

data allows the network to fit with greater ease. The CFNN also suffers from the trial-and-error based problems in setting parameters that plague the BPNN. However, the novel method in which the CFNN approaches the task does give it an advantage in suitability over the BPNN. The CFNN has shown a great deal of promise in this application and it is worthy of further study.

7.4 Summary

The CFNN has been presented as a novel method for the verification of keystroke dynamics. The network has been shown to distinguish legitimate users from impostors by training on data from the legitimate user only. The networks performance has been shown to be good in comparison with the other methods presented, given the lower data requirements for training. It can be seen that the CFNN is a promising method that gives good results for consistent users. The biased data gives results which are on a par with the other methods without the need for the network to be trained on the impostor samples.

Chapter 8

Conclusions

The probabilistic neural network has been successfully applied to the task of keystroke recognition. It has been shown to perform better than the BPNN and statistical techniques with 6% improvement in FA and FR over the traditional BPNN approach. A method of pre-processing the keystroke time intervals has been implemented and its effects shown to help reduce the FR error for a small increase in FA error. The features that make up a keystroke sample have been discussed. It has been shown that there is a distinct pattern in the way people type and that it may be fully exploited to reinforce password security. Our results highlight the PNN and CFNN as strong candidates for this application.

The PNN, by nature, has several advantages inherent in its structure when compared to other networks. Advantages such as data removal/addition without re-training and parallel processing are particularly suited to this application, allowing new users to be added and old ones removed with minimal disruption (a BPNN would require time-consuming re-training). The PNN implementation does produce a significantly larger network than the BPNN, with each training sample allocated to a node in the pattern layer. However, the parallel nature of the PNN lends itself easily to implementation with parallel processing. The PNN is also shown to be more predictable than its BPNN counterpart and hence more suited to a mission critical application such as security.

Using the Manhattan distance measure in place of the Euclidean distance measure has yielded significant improvements of over 10% in performance for the PNN. The choice of distance measure has been demonstrated to have greater effect when outlier data is present in the samples.

Problems encountered with outlier data are a major factor affecting the performance of all the neural networks presented. The pre-processing of data using z-scores has been implemented. It offers a method of reducing FR errors, but has been shown to have a small but undesirable effect on the FA errors.

We have demonstrated a novel approach using a CFNN to interpolate typing characteristics. The difference between the CFNN output and the actual sequence can be evaluated, with a large difference indicating an impostor. We have used a basic interpolation scheme to perform pattern classification with high success rates with the added benefit that the network is very easy to implement, as the windowing is a pre-processing function.

This work shows that keystroke dynamics provides a method of securing passwords without the need for any extra hardware. The uniqueness of keystroke timing data amongst users is clearly distinguishable with pattern recognition techniques. The performance of the PNN on the biased data is close to meeting acceptable error levels that would be required for a truly secure system. It is evident, however, that there is still scope for further research in this area.

8.1 Further Work

Keystroke dynamics is an area of biometric identification that has received the attention of only a few research studies. As such, there is still a great deal of work that may be pursued on the subject. For instance, the consistency of the typing samples is a

major problem in keystroke dynamics. Users that are more consistent allow tighter decision boundaries and therefore better distinction between classes, resulting in lower errors. A method of determining the consistency and suitability of a user's typing style would be useful. As these inconsistent features are applied to the network, some weighting may be applied so that less reliable ones do not confuse the classifier. It has been shown in this thesis that the BPNN may automatically compensate for this in its weights. A method of adapting the BPNN as a feature selector on the front of a PNN or CFNN network may increase performance and is therefore an area for further research.

A method for determining the optimal spread function for the PNN would bring performance gains. This decision could be based on the output of a technique that determines the consistency of the user.

Feature extraction of the keystroke dynamic data requires more work. Whilst the z-score test provides a method for pre processing the data which gives improved results, no doubt there is a more elegant solution to this problem.

The use of keystroke dynamics with other keyboard-based behaviour classification could be explored. For example, there are often many different ways of accomplishing the same task on a PC, such as copy and pasting. It is possible to copy and paste using the keyboard shortcuts, the toolbar buttons or the menu bar items. A system may learn the specific traits of a user with regard to these tasks and suspect an impostor if the pattern of behaviour deviates from that of the regular user.

The use of keystroke dynamics for the typing of numbers such as credit card codes or telephone numbers is an area for further exploration.

References

[1] Umphress D. and Williams G. (1985). "Identity verification through keyboard characteristics." *International Journal of Man Machine Studies*, **23**, 263-273.

[2] Garcia, J. D. (1986). "Personal Identification Apparatus" **4,621,334.**

[3] Bleha, S. A. (1988). "Recognition systems based on dynamics time durations between terminal keystrokes" *Ph.D Dissertation, University of Missouri-Columbia*, **AAG8826572.**

[4] Leggett J. and Williams G. (1988). "Verifying identity via keystroke characteristics." *Int.J.of Man-Machine studies*, **28**, 67-76.

[5] Young, J. R. and Hammon, R. W. (1989). "Method and apparatus for verifying an individual's identity" *United States Patent* **4,805,222.**

[6] Bleha S., Slivinsky C. and Hussien B. (1990). "Computer-access security systems using keystroke dynamics." *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **12**, 1217-1222.

[7] Bleha S.A. and Obaidat M.S. (1991). "Dimensionality reduction and feature extraction applications in identifying computer users." *IEEE Transactions on Systems, Man and Cybernetics*, **21**, 452-456.

[8] Bleha S.A. (1993). "Computer users verification using the perceptron algorithm." *IEEE Transactions on Systems, Man and Cybernetics*, **23**, 900-902.

[9] Legget J. and Williams G. (1991). "Dynamic identity verification via keystroke dynamics." *International Journal of Man Machine Studies*, **35**, 859-870.

[10] Brown, M. E. and Rogers, S. J. (1996). "Method and apparatus for verification of a computer user's identification, based on keystroke characteristics" *United States Patent* **5,557,686.**

[11] Brown M. (1993). "User identification via keystroke characteristics of typed names using neural networks." *International Journal of Man Machine Studies*, **39**, 999-1014.

[12] Obaidat M.S. (1993). "An online neural network system for computer access security." *IEEE Transactions on Industrial Electronics*, **40**, 235-242.

[13] Obaidat M.S. and Macchiarolo D.T. (1994). "A multilayer neural network system for computer access security." *IEEE Transactions on Systems, Man and Cybernetics*, **24**, 806-813.

[14] Mahar D. and Napier R. (1995). "Optimizing digraph-latency based biometric typist verification systems - inter and intra typist differences in digraph latency distributions." *International Journal of Human Computer Studies*, **43**, 579-592.

[15] Obaidat M.S. (1995). "Verification methodology for computer systems users." *Proceedings of the ACM Symposium on Applied Computing*, 258-262.

[16] De Ru W.G. (1996). "Reinforcing password authentication with typing biometrics." *South African Computer Journal*, **17**, 26-35.

[17] Monrose F. and Rubin A. (1997). "Authentication via keystroke dynamics." *Proceedings of the ACM Conference on Computer and Communications Security*, 48-56.

[18] Obaidat M.S. (1997). "A simulation evaluation study of neural network techniques to computer user identification." *Information Sciences*, **102**, 239-258.

[19] Robinson J.A., Vicky M., Michael J.A. and Christine L. (1998). "Computer user verification using login string keystroke dynamics." *IEEE Transactions on Systems, Man and Cybernetics*, **28**, 236-241.

[20] Monrose F. and Rubin A. (2000). "Keystroke Dynamics as a biometric for authentication." *Future Generation Computer Systems*, **16**, 351-359.

[21] Sungazoon C., Chigeun H. and Dae Hee H. (2000). "Web-based keystroke dynamics identity verification using neural network." *Journal of Organizational Computing & Electronic Commerce*, **10**, 295-307.

[22] Daugman, J. (1994). "Biometric Personal Identification System Based on Iris Analysis" **5,291,560.**

[23] Wasserman P.D. (1993). *Advanced Methods in Neural Computing.* Van Nostrand Reinhold, New York.

[24] Theodoridis S. (1999). *Pattern Recognition.* Academic Press , San Diego.

[25] Daugman, J. (1999). "Biometric Decision Landscapes" *The Computer Laboratory*, University of Cambridge.

[26] Werbos P.J. (1990). "Backpropagation through time: What it does and how and how to do it." *Proceedings of the IEEE*, **78**, 1550-1560.

[27] Bishop C.M. (1995). *Neural Networks for Pattern Recogniton.* Clarendon Press, Oxford.

[28] Haykin S. (1999). *Neural Networks: A comprehensive foundation.* Prentice-Hall, NJ.

[29] "Neural Network Toolbox", *Matlab Users Guide*, The Mathworks Inc.

[30] Riganati J.P. (1975). "An overview of electronic ID systems." *WESCON Technical Papers*,

[31] Clarke G.M. and Cooke D. (1998). *A basic course in statistics.* Arnold, London, New York.

[32] Hemmi K. and Inoue K. (1989). "A characteristic of pointing behavior with a linear cursor key." *Transactions of the Institute of Systems, Control & Information Engineers*, **2**, 301-305.

[33] Obaidat M.S. (1994). "A comparative performance study of neural network paradigms for identiying computer users." *1994 IEEE 13th annual int.phoenix conf.on computers and communications (cat.no.94ch3399-3)*, 161-167.

[34] Obaidat M.S. (1993). "A methodoloy for improving computer access security." *Computers & Security*, **12**, 657-662.

[35] Haider S., Abbas A. and Zaidi A.K. (2000). "A multi-technique approach for user identification throughkeystroke dynamics." *2000 IEEE International Conference on Systems, Man and Cybernetics.'Cybernetics Evolving to Systems, Humans,Organizations, and their Complex Interactions' (Cat.No.00CH37166)*, **2**, 1336-1341.

[36] Anagun A.S. and Cin I. (1998). "A Neural Network Based computer access security for multiple users." *Computers & Industrial Engineering*, **35**, 351-354.

[37] Yau-Hwang, Jang-Pong Hsu and Chen-Wen Wang (1998). "A parallel fuzzy inference model with distributed prediction scheme for reinforcement learning." *IEEE Transactions on Systems, Man and Cybernetics*, **28**, 160-172.

[38] Jackson J.E. (1991). *A users guide to priciple components.* Wiley, New York.

[39] Chin-Teng L. and Ming-Chih K. (1998). "Adaptive fuzzy command acquisition with reinforcement learning." *IEEE Transactions on Fuzzy Systems*, **6**, 102-121.

[40] Hand D.J. (1994). *AI and Computer Power.* Chapman & Hall, London, New York.

[41] Kasukawa M., Mori Y. and Komatsu K. (1992). "An evaluation and improvement of user authentication system based on keystroke timing data." *Transactions of the Information Processing Society of Japan*, **33**, 728-735.

[42] Kasukawa M., Kakuda H. and Mori Y. (1993). "An identity authentication method based on arpeggiokeystrokes." *Transactions of the Information Processing Society of Japan*, **34**, 1198-1205.

[43] Mefford M.J. (1990). "An in-depth exploration of the PC keyboard and its interrupt serviceroutines." *Microsoft Systems Journal*, **5**, 22-46.

[44] Chia-Feng J. and Chin-Teng L. (1998). "An online self-constructing neural fuzzy inference network and its applications." *IEEE Transactions on Fuzzy Systems*, **6**, 12-32.

[45] Katsikas S. and Gritzalis D. (1996). "Applications of keystroke analysis for improved login security and continous user authentication." *International Security Conference*, 283-294.

[46] Zhang C. and Sun Y. (2000). "AR model for keystroker verification." *2000 IEEE International Conference on Systems, Man and Cybernetics.'Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions' (Cat.No.00CH37166)*, **4**, 2887-2890.

[47] Gaines R., Lisowsli W., Press S. and Shapiro N. (1980). "Authentication by Keystoke Timing: Some preliminary results." *Rand Report*, **R-2526-NSF**, 5/80.

[48] Phillips K. (2000). "Biometric identification looms on the landscape on network log-ins -- High end technology is becoming more affordable." *PC WEEK*, **14**, 99-

[49] Coltell O. and Badia J. (1999). "Biometric identification system based in keyboard filtering." *IEEE Annual International Carnahan Conference on security technology, proceedings*, 203-209.

[50] Alexandre T.J. (1997). "Biometrics on Smart Cards: an approach to keyboard behavioural signature." *Future Generation Computer Systems*, **13**, 19-26.

[51] Obaidat M.S. (1994). "Comparative performance study of neural network paradigms for identifying computer users." *International Phoenix Conference on Computers and Communications*, 161-167.

[52] Emad W.S., Prokhorov D.V. and Wunsch D.C. (1998). "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks." *IEEE Transactions on Neural Networks*, **9**, 1456-1469.

[53] Daw-Tung L. (1997). "Computer-access authentication with neural network based keystroke identity verification." *1997 IEEE International Conference on Neural Networks.Proceedings (Cat.No.97CH36109)*, **1**, 174-178.

[54] Lin D. (1997). "Computer Access Authentication with Neural Network based keystroke identity vefication." *IEEE International conference on neural networks*, **1**, 174-178.

[55] Adair K.L. (1994). "Computer imposter generation for user identification via keystroke characteristics of typed names using neural networks." *Proceedings of SEC '94.32nd Annual ACM Southeast Conference*, 121-124.

[56] Reynolds T.H. and Bonk C.J. (1996). "Computerized prompting partners and keystroke recording devices: two macro driven writing tools." *Educational Technology Research & Development*, **44**, 83-97.

[57] Hand D.J. (1997). *Construction and Assessment of Classification Rules.* Wiley, Chichester, England.

[58] Sheperd S. (1995). "Continuous authentication by analysis of keyboard typing characteristics." *IEE conference on security and detection*, 111-114.

[59] Therrien C. (1989). *Decision Estimation and Classification.* Wiley, New York.

[60] Lachenbruch P.A. (1975). *Discriminant Analysis.* Wiley, New York.

[61] Henderson R., Mahar D., Saliba A., Deane F. et al (1998). "Electronic monitoring systems: an examination of physiologycal activity and task performance within a simulated keystroke security and electronic performance monitoring system." *International Journal of Human-Computer Studies*, **48**, 143-157.

[62] De Ru W.G. and Eloff J.H.P. (1997). "Enhanced password authentication through fuzzy logic." *IEEE Expert*, **12**, 38-45.

[63] Shen W., Surette M. and Khanna R. (1997). "Evaluation of automated biometrics based identification and verification systems." *Proc of IEEE*, **85**, 1464-1478.

[64] Pittner S. and Kamarthi S.V. (1999). "Feature extraction from wavelet coefficients for pattern recognition tasks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**, 83-88.

[65] Soechting J.F. (1997). "Flexibility and repeatibility of finger movements during typing: Analysis of multiple degrees of freedom." *Journal of computational neuroscience*, **4**,

[66] Omlin C.W. (1998). "Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks." *IEEE Transactions on Fuzzy Systems*, **6**, 76-89.

[67] Zhang J. and Morris A.J. (1995). "Fuzzy neural networks for nonlinear systems modelling." *IEE Proceedings - Control Theory and Applications*, **142**, 551-561.

[68] Yager R.R. (1994). *Fuzzy Sets Neural Networks and Soft Computing.* Van Nostrand Reinhold, New York.

[69] Scheunders P. (1999). "High dimensional clustering using frequency sensitive competitive learning." *Pattern Recognition*, **32**, 193-202.

[70] Jung-Hsien C. and Gader P.D. (1997). "Hybrid fuzzy-neural systems in handwritten word recognition." *IEEE Transactions on Fuzzy Systems*, **5**, 497-510.

[71] Joyce R. and Gupta G. (1990). "Identity authentication based on keystroke latencies." *Communications of the ACM*, **33**, 168-176.

[72] Bascou J. and Redon L. (1995). "Improving security by analysing users behaviour (computer access)." *Pacific Telecommunications Council Seventeenth Annual*

Conference Proceedings Honolulu, HI, USA., 615-619.

[73] Morikawa O. (1987). "Interpretation of keystroke data with timing information." *Transactions of the Institute of Electronics & Communication Engineers of Japan*, **J70D**, 2198-2203.

[74] Anon (1984). "Keystroke dynamics authentication of computer terminal users." *Bioaccess systems 2000, 2000A, 2001, 2011, and OEM* ,

[75] Napier R. (1995). "Keyboard user verification: toward an accurate, efficient and ecologically valid algorithm." *Australian Computer Science Communications*, **17**, 407-412.

[76] Morikawa O., Kimura I. and Kasukawa M. (1990). "Keystroke data collection system for a personal computer." *Transactions of the Information Processing Society of Japan*, **31**, 1822-1831.

[77] Rostar R. and Olejar J. (1995). "Keystroke dynamics based user autentication using neural networks." *Artificial Neural Nets and Genetic Algoritms.Proceedings of the International Conference*, 194-197.

[78] Floyd R.E., Ovies H., Sweet D.A. and Wirick P.G. (1981). "Keystroke intercept circuit." *IBM Technical Disclosure Bulletin*, **23**, 4420-4422.

[79] Newham E. (1996). "Knowing me knowing you: How biometric security systems use human characteristics." *Communications International*, **23**, 55-58.

[80] Mitra S., De R.K. and Pal S.K. (1997). "Knowledge based fuzzy MLP for classification and rule generation." *IEEE Transactions on Neural Networks*, **8**, 1338-1350.

[81] Hanningham T.J. (1997). *Mastering statistics.* Macmillan, Basingstoke, England.

[82] Haider S., Abbas A., Zaidi A.K. and Abbas K. (2000). "Multi-technique approach for user identification through keystroke dynamics." *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, **2**, 1336-1341.

[83] Hsin-Chia F. and Yeong Y., X (1998). "Multiliuguistic Handwritten Character Recognition by Bayesian Decision-Based Neural Networks." *IEEE Transactions on Signal Processing*, **46**, 2781-2789.

[84] Hair J.F. (1998). *Multivariate Data analysis.* Prentice Hall, Upper Saddle River, N.J.

[85] Anagun A.S. (1999). "Neural Network application for a computer access security system: Keystroke dynamics versus voice patterns." *Intelligent systems through artificial neural networks*, **9**, 905-910.

[86] Bart K. (1992). *Neural Networks for signal processing.* Prentice Hall, Englewood Cliffs, N.J.

[87] Obaidat M.S. (1992). "On-line neurocomputing system to identify computer users." *Proceedings of the 1992 summer computer simulation conference*, **xxii+1273**, 403-407.

[88] Suzuki K. (1995). "On a simple method to measure the intensity of keystrokes." *Symbiosis of Human and Artifact.Proceedings of the Sixth International Conference on Human-Computer Interaction (HCI International '95).Amsterdam, Netherlands*, **B**, 797-801.

[89] Brown M. and Rogers S.J. (1994). "Practical approach to user authentication." *Annual Computer Security Applications Conference*, 108-116.

[90] Monrose F., Reiter M.K. and Wetzel S. (1999). "Password hardening based on keystroke dynamics." *Proceedings of the ACM Conference on Computer and Communications Security*, 73-82.

[91] de Bruyne P., Diehl R., Losa E. and Saladini M. (1990). "Password Plus, authentication using dynamic features." *Scientia Electrica*, **36**, 11-19.

[92] Duda R.O. and Hart P.E. (1973). *Pattern Classification & Scene Analysis.* Wiley, New York.

[93] Sing-Tze B. (1984). *Pattern Recognition - Applications to large data-set problems.* Marcel Dekker, New York.

[94] Schalkoff R. (1992). *Pattern Recognition: Statisical, Structural, and Neural Approaches.* Wiley, New York.

[95] Ullmann J. (1973). *Pattern Recognition Techniques.* Butterworths, London.

[96] Bleha S.A., Knopp J. and Obaidat M.S. (1992). "Performance of the perception algorithm for the classification of computer users." *Applied Computing: Technological Challenges of the 1990's*, 863-866.

[97] Fabian W. (1995). "Physical access to computers: can your computer be trusted?" *Proceedings.The Institute of Electrical and Electronics Engineers 29th Annual 1995 International Carnahan Conference on Security Technology (Cat.No.95CH3578-8)*, 244-251.

[98] Tee E.R. and Selvanathan N. (1996). "PIN signature verification using wavelet transform." *Malaysian Journal of Computer Science* , **9**, 71-78.

[99] Crepeau C. and Salvail L. (1995). "Quantum oblivious mutual identification." *Advances in Cryptology - EUROCRYPT '95.Int.Conf.on the theory and application of cryptoqraphic techniques.proceedings*, **xiv+416**, 133-146.

[100] De Ru W.G. and Eloff J.H.P. (1995). "Reinforcing password authentication with typing biometrics." *Information Security - the Next Decade.Proceedings of the IFIP TC11 Eleventh International Conference on Information Security, IFIP/Sec 1995*, 562-572.

[101] Richards D.R. (1995). "Rules of thumb for biometric systems." *Security Management*, **39**, 67-69.

[102] Chen S.B., Wu L. and Wang Q. (1997). "Self-learning fuzzy neural networks for control of uncertain systems with time delays." *IEEE Transactions on Systems, Man and Cybernetics*, **27**, 142-148.

[103] Cliff A.D. (1973). *Spatial Autocorrelation.* Pion, London.

[104] Candocia F.M. (1999). "Super resolution images based on local correlations." *IEEE Transactions on Neural Networks*, **10**, 372-380.

[105] Mahar D. and Napier R. (1998). "The effects of password length and reference profile size on the performance of a multivariate text-dependent typist verification system." *Interacting with computers*, **10**, 375-383.

[106] Anderson T.W. (1996). *The new statistical analysis of data.* Springer, New York.

[107] Kohonen T. (1990). "The self-organising map." *Proceedings of the IEEE*, **78**, 1464-1488.

[108] Hand D.J. (1987). *The statistical consultant in action.* Cambridge University Press,

Cambridge, New York.

[109] Deane F., Henderson R., Mahar D. and Saliba A. (1995). "Theoretical examination of the effects of anxiety and electronic performance monitoring on behavioural biometric security systems." *Interacting with computers*, **7**, 395-411.

[110] Rostar R. and Durkovic M. (1994). "Two methods for computer users identification." *Proceedings of the Twelfth IASTED International Conference Applied Informatics.IASTED '94*, 180-183.

[111] Omote K. and Okamoto E. (1999). "User identification system based on biometrics for keystroke." *Information and Communication Security.Second International Conference, ICICS'99.Proceedings*, **1726**, 216-229.

[112] Barrelle K., Laverty W., Henderson R., Gough J. et al (1995). "User verification through pointing characteristics: an exploration examination." *International Journal of Human Computer Studies*, **45**, 47-57.

[113] Yinghua L., Cunningham G.A. and Coggershalc S. (1997). "Using fuzzy partitions to create fuzzy systems from input-output data and set the initial weights in a fuzzy neural network." *IEEE Transactions on Fuzzy Systems*, **5**, 614-621.

[114] Obiadat M.S. (1997). "Verification of computer users using keystroke dynamics." *IEEE Transactions on Systems, Man and Cybernetics*, **27**, 261-269.

[115] Kechriotis G., Zervas E. and Manolakos E.S. (1994). "Using recurrent neural networks for adaptive communication channel equalisation." *IEEE Transactions on Neural Networks*, **5**, 267-278.

[116] Nerrand O., Roussel-Ragot P., Urbani D., Personnaz L. et al (1994). "Training recurrent networks: Why and how? An illustration in dynamical process modeling." *IEEE Transactions on Neural Networks*, **5**, 178-184.

[117] Puskorius G.V. and Feldkamp L.A. (1994). "Neurocontrol of nonlinear dynamical systems with Kalman filter-trained recurrent networks." *IEEE Transactions on Neural Networks*, **5**, 279-297.

[118] Bose N.K. and Liang P. (1995). *Neural network fundamentals with graphs, algorithms, and applications*. McGraw-Hill, New York.

[119] Elman J.L. (1990). "Finding structure in time." *Cognitive Science*, **14**, 179-211.

[120] Song, D., Venable, P., and Perrig, A. (1997). "User recognition by keystroke latency pattern analysis" *Dissertation, Carnegie Mellon University*, **Internet.**

Appendices

8.2 Appendix A - Keystroke Capture Program

```
*****************************************************************
*       Keystroke time capture program                          *
*       By Steven Shorrock              1/11/98                 *
*       Motorola MC68000 Assembly Code                          *
*****************************************************************

*****************************************************************
outport EQU     $FE00A1 * address of output port
status  EQU     $FE00A3 * address of status register
prog    EQU     $1000
data    EQU     $2000
timesu  EQU     $3000
timesp  EQU     $4000
outrdy  EQU     0       * location of output ready bit
inprdy  EQU     1       * location of input ready bit
stkptr  EQU     $97E
lf      EQU     $0A
cr      EQU     $0D
*****************************************************************

start   ORG     prog
        LEA     stkptr,A7

* Main program loop
choice  LEA     mes6,A5     * loads first menu
        BSR     outstr      * prints to screen
        BSR     inpchr      * receives user input
        CMP.B   #'c',D0     * chooses capture/save/quit
        BEQ     capture
        CMP.B   #'s',D0
        BEQ     save
```

```
        CMP     #'q',D0
        BEQ     finish
        LEA     err1,A5
        BSR     outstr
        BRA     choice
* End main program loop

*****************************************************************
*       Capture: Saves the persons name and then gets their     *
*       username and password so that they can be compared with *
*       that of the future samples.                             *
*****************************************************************
capture MOVE.L  #0,samnum  * resets sample number to 0
        LEA     mes4,A5    * asking to type persons name
        BSR     outstr     * print to screen

        LEA     name,A5
        BSR     inputst    * stores name

        MOVE.L  #$3000,lastu
        MOVE.L  #$4000,lastp     * sets values for lastu lastp

        LEA     mes5,A5    * please enter username & password
        BSR     outstr
        LEA     mes1,A5    * username prompt
        BSR     outstr
        LEA     compu,A5   * stores comparison username at compu
        BSR     inputst
        LEA     mes2,A5    * password prompt
        BSR     outstr
        LEA     compp,A5   * stores comparison password at compp
        BSR     inputst
        LEA     mes7,A5    * ******data capture******* starts
        BSR     outstr

*****************************************************************
*       Cap: this is the menu loop for obtaining the            *
*       keystroke timing data.                                  *
*****************************************************************
cap     LEA     mes1,A5    * Please enter username
        BSR     outstr
```

```
        LEA     user,A5     * sets storage area for username
        LEA     tuser,A6
        CLR.L   D0          * clears D0 so character echo is on
        BSR     inpstr      * obtains timing data

        LEA     mes2,A5     * Password prompt
        BSR     outstr

        LEA     pass,A5     * sets storage area for password
        LEA     tpass,A6
        MOVE.L  #1,D0       * sets echo off and asterisks on
        BSR     inpstr      * obtains timing data

        LEA     endl,A5     *line feed & carriage return
        BSR     outstr

        LEA     mes8,A5     * accept reject finish
        BSR     outstr
        BSR     inpchr
        CMP.B   #'f',D0
        BEQ     fini        * IF f THEN goto finish
        CMP.B   #'r',D0
        BEQ     cap         * IF r THEN goto cap (rejects sample)
        BSR     compare     * ELSE goto compare (check typed correctly)
        LEA     mes12,A5    * display number of samples
        BSR     outstr
        MOVE.L  samnum,D0
        BSR     outnum
        BRA     cap         * loop

*****************************************************************
*       Fini: this finishes the capturing                       *
*****************************************************************
fini    BSR     compare     * compares the last sample
        LEA     mes12,A5
        BSR     outstr      * display number of samples
        MOVE.L  samnum,D0
        BSR     outnum
        BRA     choice      * back to main program loop
*****************************************************************
```

```
*       Save: this prints the timing data to screen, this can   *
*       be saved to disk through the 68000 softlog command      *
*****************************************************************
save    LEA     mes11,A5
        BSR     outstr      * prints the persons name
        LEA     name,A5
        BSR     outstr

        LEA     mes9,A5
        BSR     outstr
        LEA     $3000,A5
        LEA     lastu,A6
        BSR     display     * prints the username times

        LEA     mes10,A5
        BSR     outstr

        LEA     $4000,A5
        LEA     lastp,A6
        BSR     display     * prints the password times

        LEA     endl,A5
        BSR     outstr

*****************************************************************
finish  LEA     endl,A5     * line feed & carriage return
        BSR     outstr
        TRAP    #15         * Interrupt processor
        DC.W    $10         * send Halt command to processor
*****************************************************************
*****************************************************************
*       Compare: checks to make sure the username and password  *
*       that the user typed corresponds with the origionals and *
*       that the counter has not overflowed                     *
*****************************************************************
compare LEA     user,A5     * point to current username
        LEA     compu,A6    * point to stored comparison origional
loop2   CMPM.B  (A5)+,(A6)+ * compare
        BNE     error2      * if not equal error
        CMP.B   #0,(A6)     * IF not at end of username
        BNE     loop2       * THEN loop back
```

```
        LEA     pass,A5    * same as above for password
        LEA     compp,A6
loop3   CMPM.B  (A5)+,(A6)+
        BNE     error2
        CMP.B   #0,(A6)
        BNE     loop3

        LEA     tuser,A5   * point to timings for username
testu   CMP.L   #$0009FFFF,(A5)
        BGT     error      * IF counter has overflowed error
        CMP.L   #0,(A5)+   * Keep going through all samples
        BNE     testu
        LEA     tpass,A5   * same for password
testp   CMP.L   #$0009FFFF,(A5)
        BGT     error
        CMP.L   #0,(A5)+
        BNE     testp

store   MOVEA.L lastu,A5   * store correct timing data
        LEA     tuser,A6
loop4   MOVE.L  (A6)+,(A5)+
        BNE     loop4
        MOVE.L  A5,lastu
storep  MOVEA.L lastp,A5
        LEA     tpass,A6
loop5   MOVE.L  (A6)+,(A5)+
        BNE     loop5
        MOVE.L  A5,lastp

        MOVE.L  samnum,D0
        ADD     #1,D0      * add one to correct samples number
        MOVE.L  D0,samnum
        RTS

error2  LEA     err3,A5    * displays wrong username
        BSR     outstr
        RTS

error   LEA     err,A5     * displays too long a pause (overflow)
        BSR     outstr
        RTS
```

```
*****************************************************************
*       Display: this prints the timing sample to screen        *
*****************************************************************
display MOVE.L  (A5)+,D0   * get the first time interval
        MOVEA.L A5,A4      * save A5 as outnum uses it
        BSR     outnum     * print it to screen
        MOVEA.L A4,A5      * reload A5
        CMP.L   #0,(A5)    * check for end
        BNE     display    * ELSE repeat
        LEA     endl,A5    * line feed
        BSR     outstr
        MOVEA.L A4,A5      * reload A5
        ADD     #4,A5      * increment A5 by a longword (1 sample)
        CMP.L   (A6),A5    * IF at end of sample THEN RTS
        BLT     display    * ELSE repeat
        RTS

*****************************************************************
*       Subroutine: inpstr                                      *
*       Receives characters and the delays between key          *
*       presses. Outputs a '*' after each character if          *
*       D0 is set to '1'                                        *
*****************************************************************
inpstr2 CMP     #1,D0      * check for echo flag
        BNE     inpstr3    * IF echo on goto inpstr3
        MOVE.B  #'*',outport * ELSE echo asterisk to screen
        BRA     inpstr     * skip next line
inpstr3 MOVE.B  -1(A5),outport *echo character to screen
inpstr  CLR.L   D3         * clear counter, start polling loop
inpstr1 ADDQ.L  #1,D3      * add one to counter
        BTST    #inprdy,status   * IF keypress THEN stop
        BEQ     inpstr1    * ELSE repeat adding to counter
        MOVE.L  D3,(A6)+   * store timing data
        MOVE.B  outport,(A5) * store key that was pressed
        CMP.B   #$0D,(A5)+ * IF carriage return pressed THEN RTS
        BNE     inpstr2    * ELSE repeat
        MOVE.L  #0,(A6)
        RTS

*****************************************************************
*       Subroutine: outstr                                      *
*       Takes the start of message in A5                        *
```

```
*       and outputs to screen                                          *
*******************************************************************
outstr  BTST    #outrdy,status * check output is ready
        BEQ     outstr     * poll outrdy
        MOVE.B  (A5),outport * display character
        CMP.B   #0,(A5)+   * IF end THEN RTS
        BNE     outstr     * ELSE repeat
        RTS
*******************************************************************
*       Subroutine: outnum                                         *
*       The number in D0 is output to the screen                   *
*******************************************************************
outnum  MOVE.L  #6,D2
        LEA     num+8,A1
        MOVE.L  D0,D1
        SWAP    D0
        CMP.W   #9,D0
        BGT     toobig
        SWAP    D0
rep     DIVU    #10,D1
        SWAP    D1
        MOVE.B  D1,D0
        ADD.B   #$30,D0
        MOVE.B  D0,-(A1)
        LSR.L   #8,D1
        LSR.L   #8,D1
        TST     D1
        BEQ     exit
        SUB     #1,D2
        BNE     rep
exit    MOVE.B  (A1)+,D0
        BSR     outchr
        CMP     #num+8,A1
        BNE     exit
        MOVE.B  #' ',D0
        BSR     outchr
        RTS
toobig  LEA     mes3,A5
        BSR     outstr
        RTS
*******************************************************************
```

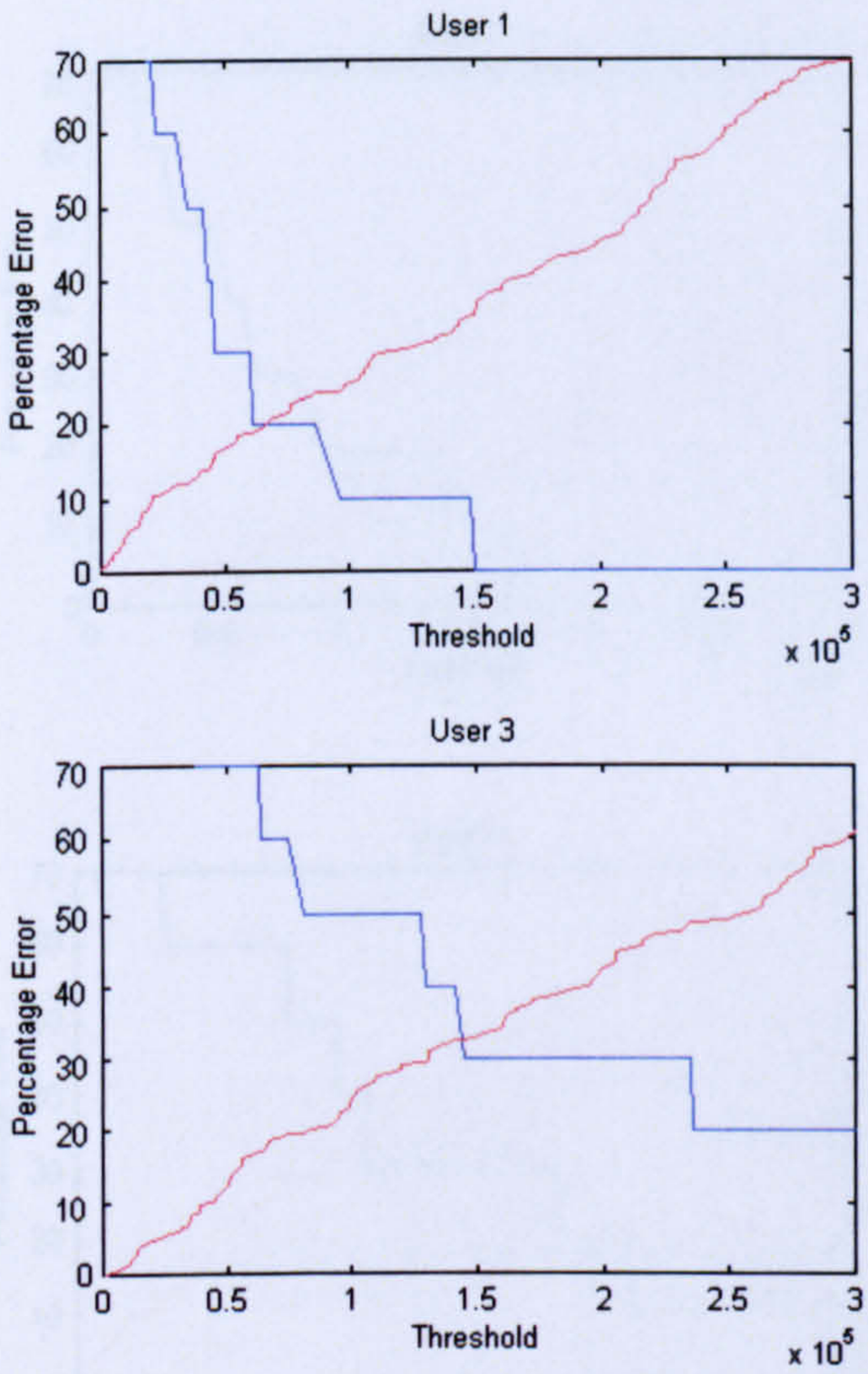
```
* outchr: display a single character
      outchr  BTST    #outrdy,status * poll outrdy bit
              BEQ     outchr
              MOVE.B  D0,outport * print to screen
              RTS

      inpchr  TRAP    #15
              DC.W    $11         * inputs a character, D0 contains value
              RTS
* inputst: inputs a string, making sure it is less than 20
* characters (for name)
      inputst CLR.L   D1          * clear character counter
              MOVEA.L A5,A6       * set initial conditions for inpchr
      inputl  BSR     inpchr
              MOVE.B  D0,(A6)+    * save character
              MOVE.B  D0,outport * echo character
              ADD     #1,D1       * add one to counter
              CMP     #19,D1      * check character counter is less than 20
              BGT     overflo     * IF >20 error
              CMP     #$0D,D0     * check for carriage return
              BNE     inputl      * ELSE repeat
              MOVE.B  #0,(A6)     * mark end of string with a '0'
              RTS
overflo MOVEA.L A5,A4       * save A5
              LEA     err2,A5     * output error msg
              BSR     outstr
              MOVEA.L A4,A5
              CLR.L   D1          * start again
              BRA     inputl

*****************************************************************
              ORG     data
*****************************************************************

user    DS.B    20
pass    DS.B    20
name    DS.B    20
compu   DS.B    20
compp   DS.B    20
lastu   DS.L    1
lastp   DS.L    1
```

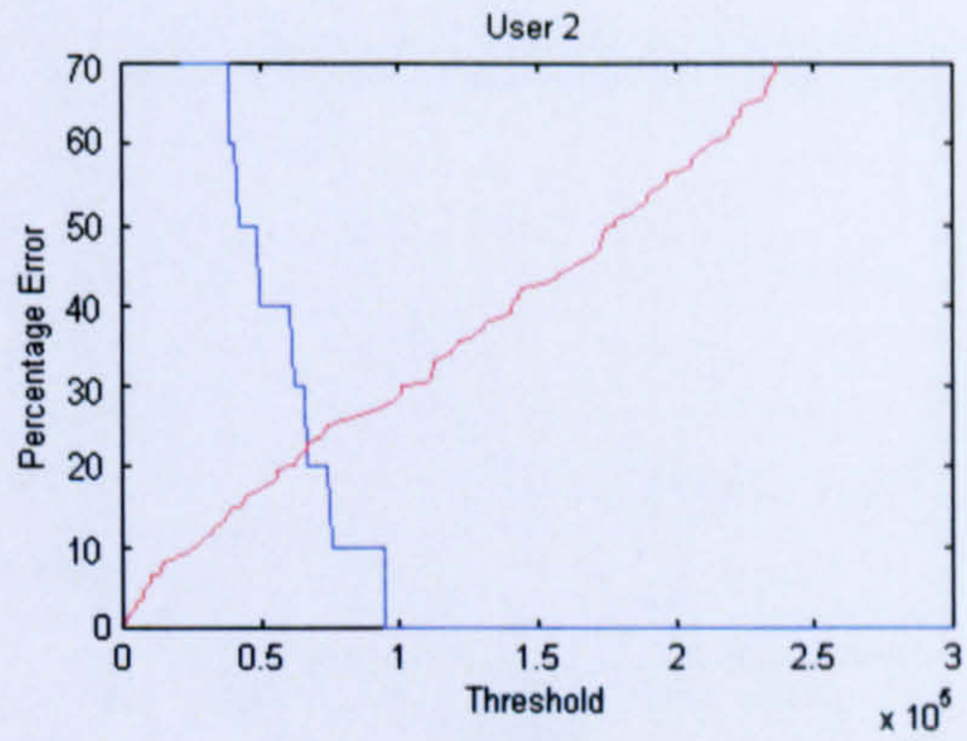
```
mes1    DC.B    cr,lf,cr,lf,'Please type (username): ',0
mes2    DC.B    cr,lf,'Please type (password): ',0
mes3    DC.B    'toobig ',0
mes4    DC.B    cr,lf,'***Set-up***',cr,lf,'Please enter your name: ',0
mes5    DC.B    cr,lf,'Please enter your username and password,',0
mes6    DC.B    cr,lf,'Please choose capture/save or quit (c/s/q): ',0
mes7    DC.B    cr,lf,'***Data capture***',cr,lf,0
mes8    DC.B    cr,lf,'Accept? Reject? Finish? (a/r/f)',0
mes9    DC.B    cr,lf,'Username times:',cr,lf,0
mes10   DC.B    cr,lf,'Password times:',cr,lf,0
mes11   DC.B    cr,lf,'Name: ',0
mes12   DC.B    cr,lf,'Number of samples = ',0
err     DC.B    lf,cr,'error! too long a pause',lf,cr,0
err1    DC.B    cr,lf,'error, try again',cr,lf,0
err2    DC.B    cr,lf,'error! max characters 20, try again',cr,lf,0
err3    DC.B    cr,lf,'wrong username or password, try again',cr,lf,0
endl    DC.B    cr,lf,0
num     DS.B    10
tuser   DS.L    200
tpass   DS.L    200
samnum  DS.L    1
************************//////////***\\\\\\\\\*************************
```

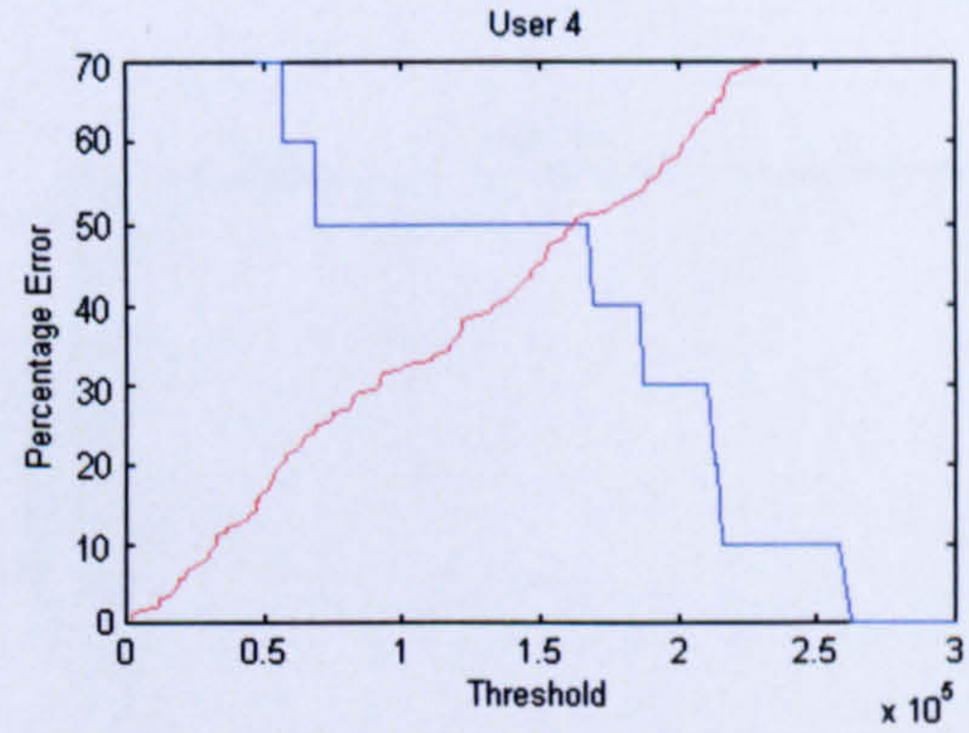
8.3 Appendix B - Graphs of Statistical Results

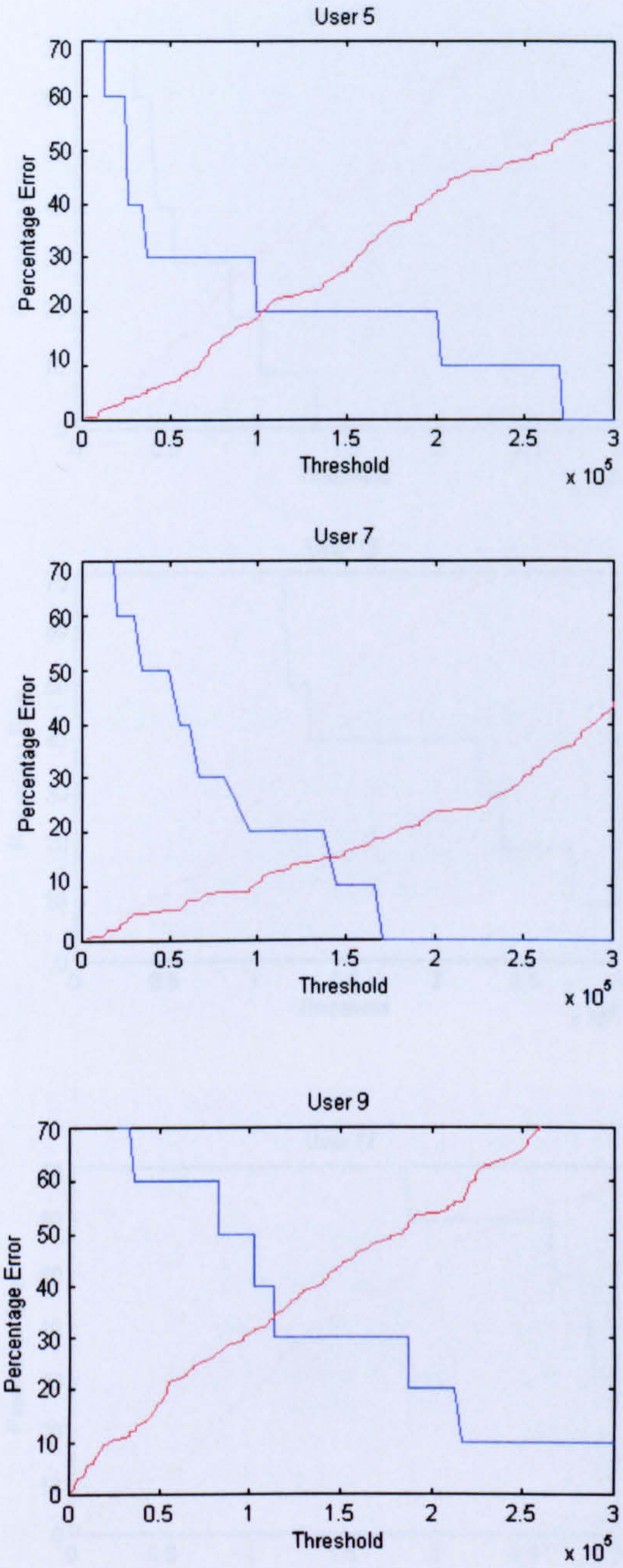
These graphs show the FA/FR plots of all the users for the statistical methods. The red line represents FA error the blue line FR error.

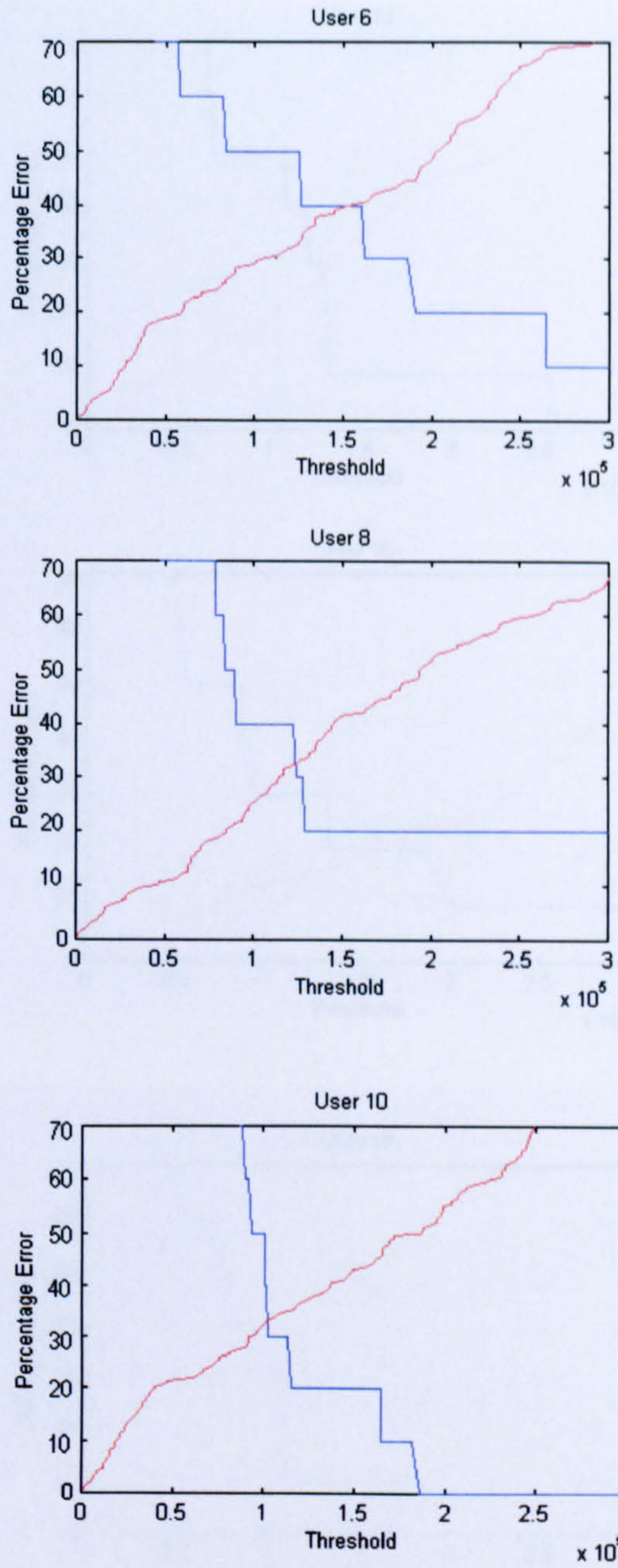
8.3.1 Euclidean Unbiased Data








User 5
Percentage Error
Threshold
x 10^5

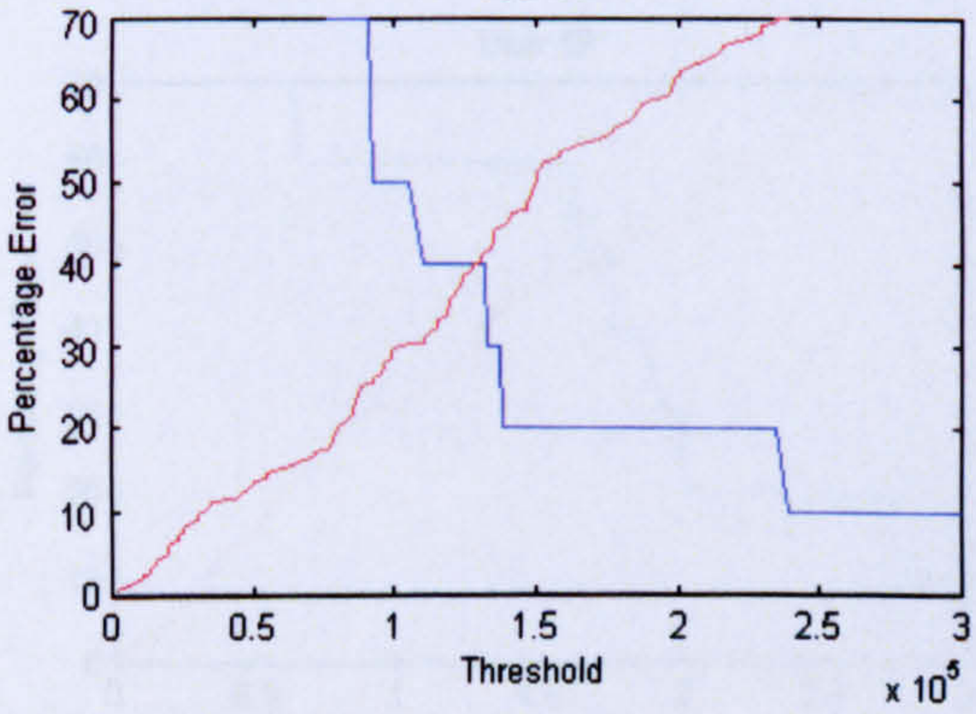

User 6
Percentage Error
Threshold
x 10^5

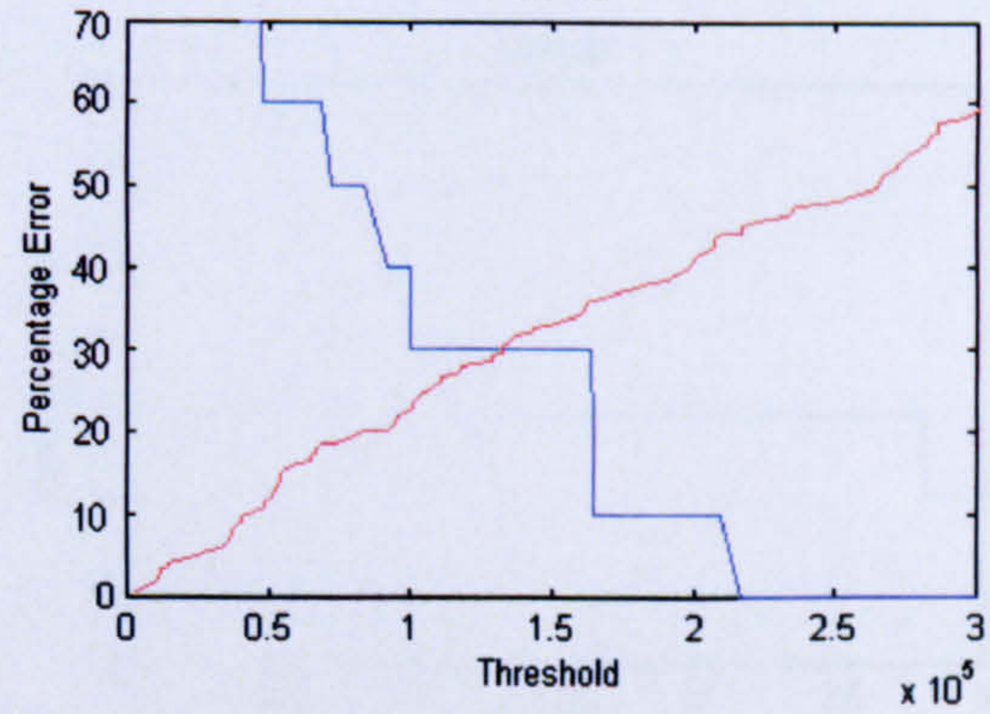
User 7
Percentage Error
Threshold
x 10^5

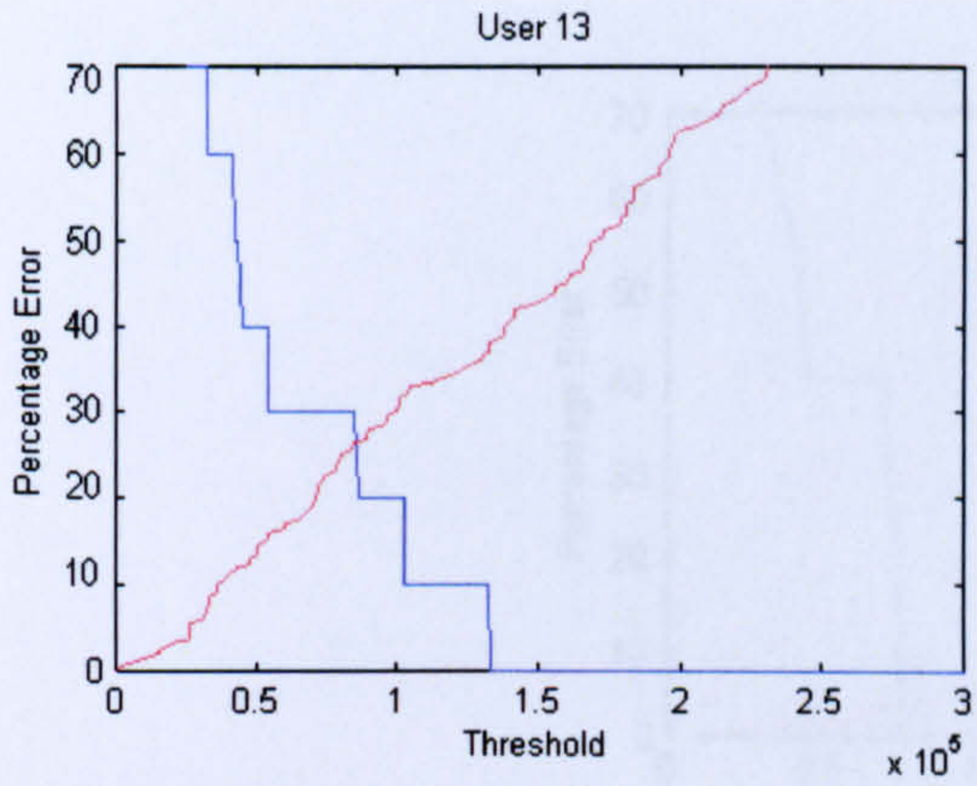
User 8
Percentage Error
Threshold
x 10^5

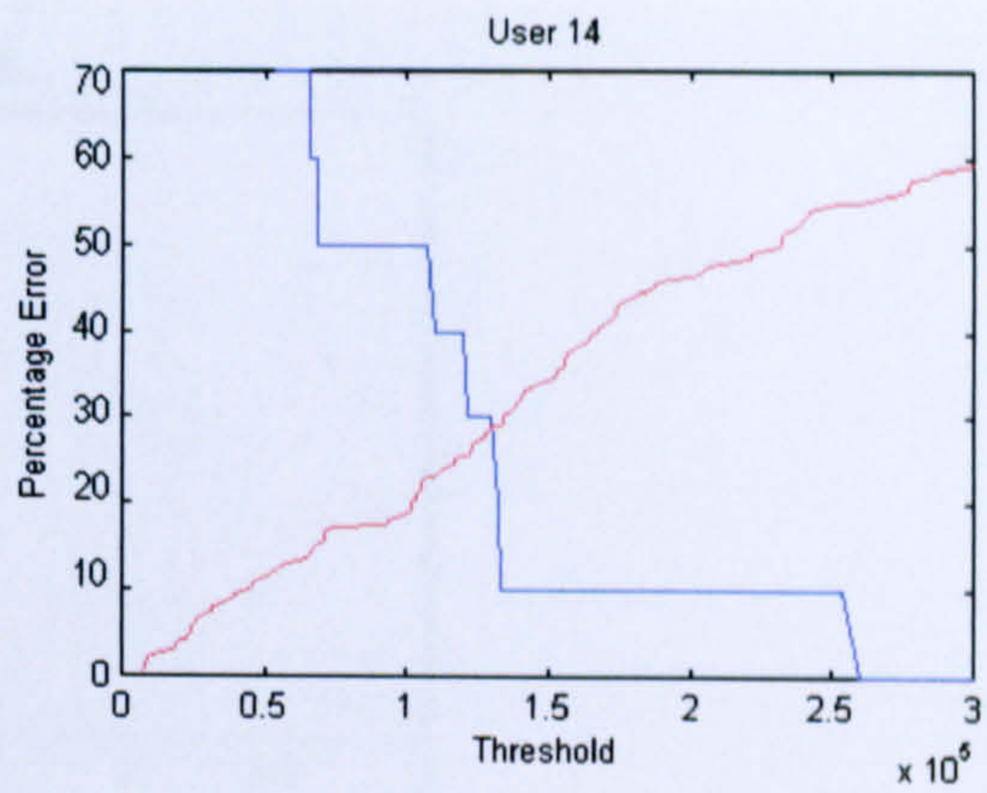
User 9
Percentage Error
Threshold
x 10^5

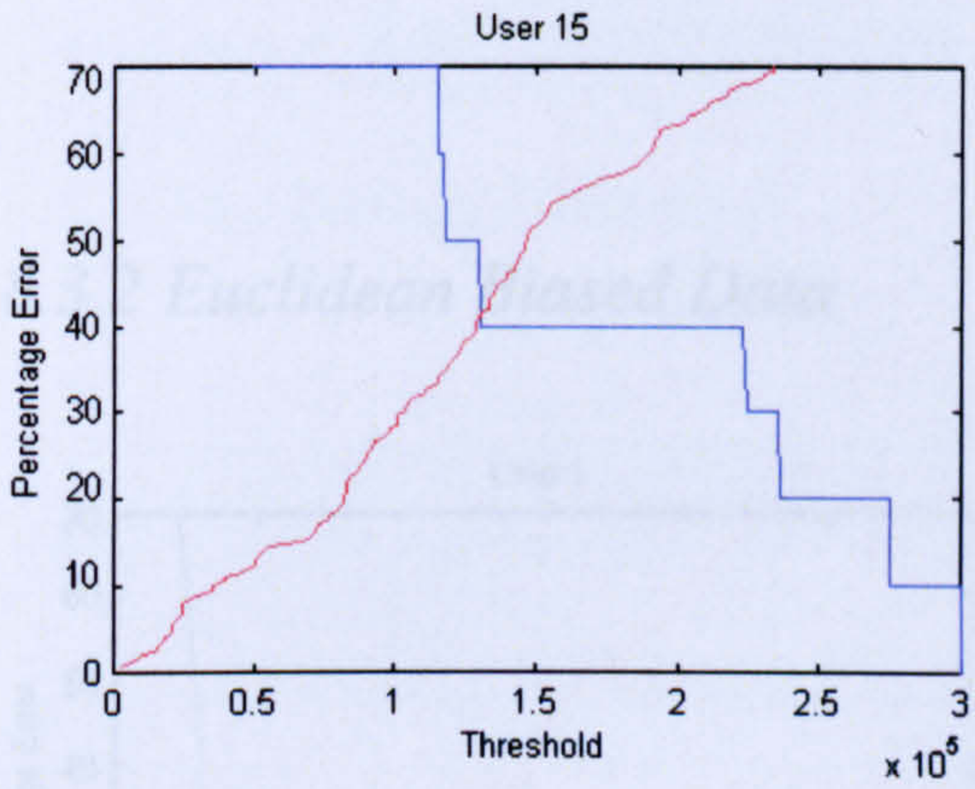
User 10
Percentage Error
Threshold
x 10^5

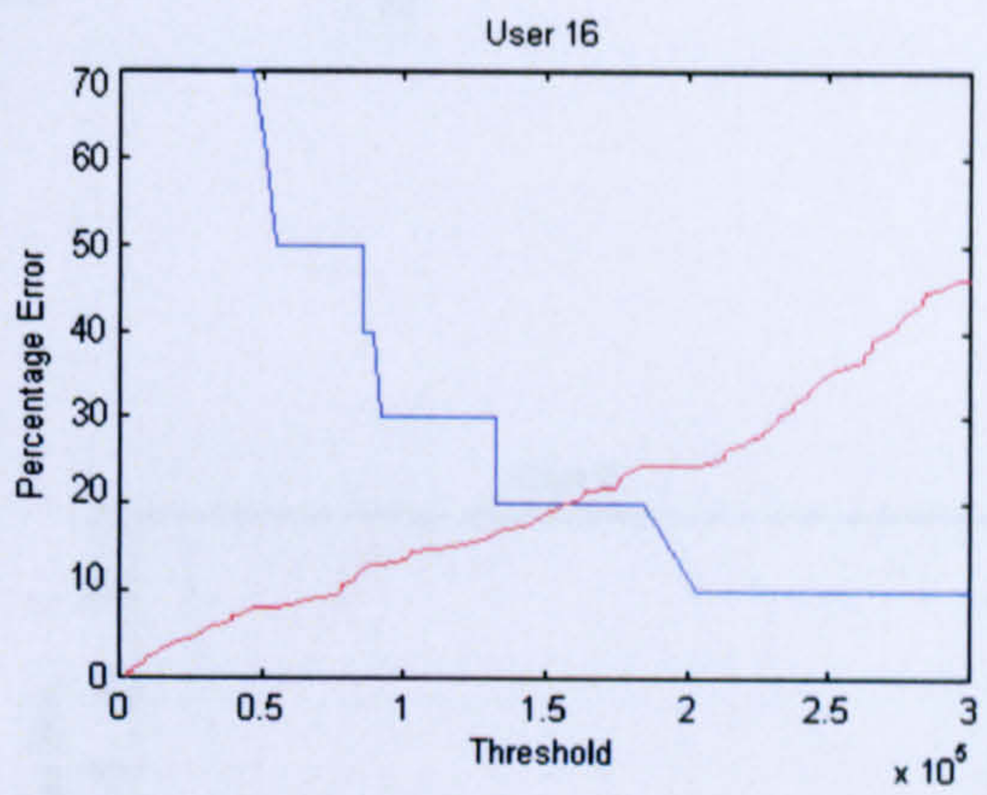

User 11
Percentage Error
Threshold
x 10^5

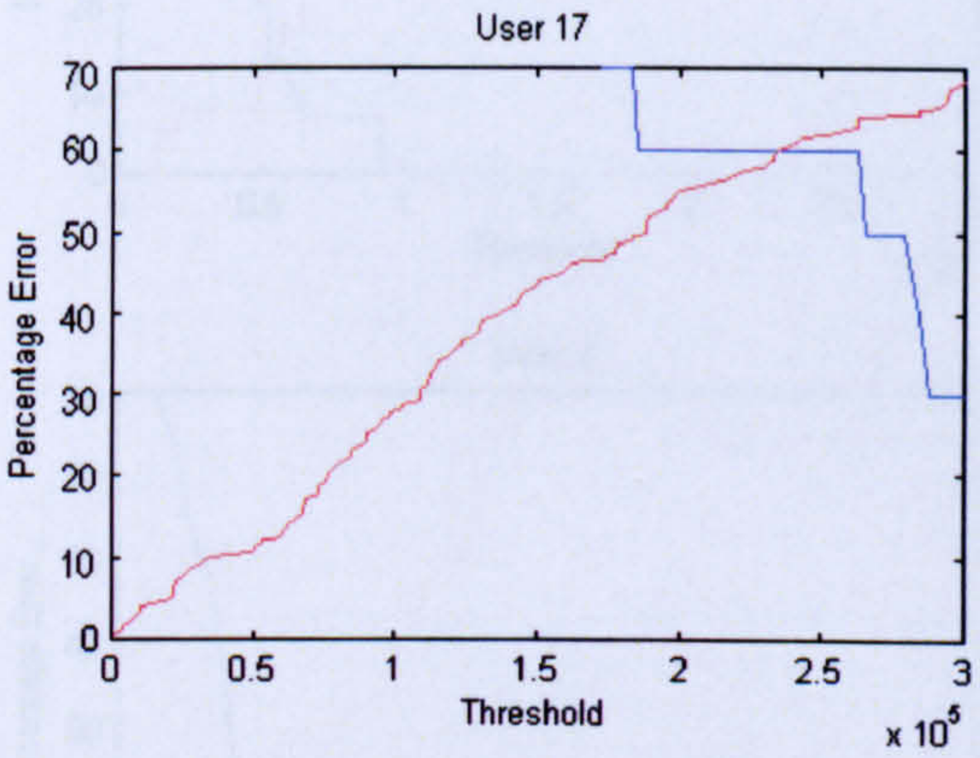

User 12
Percentage Error
Threshold
x 10^5

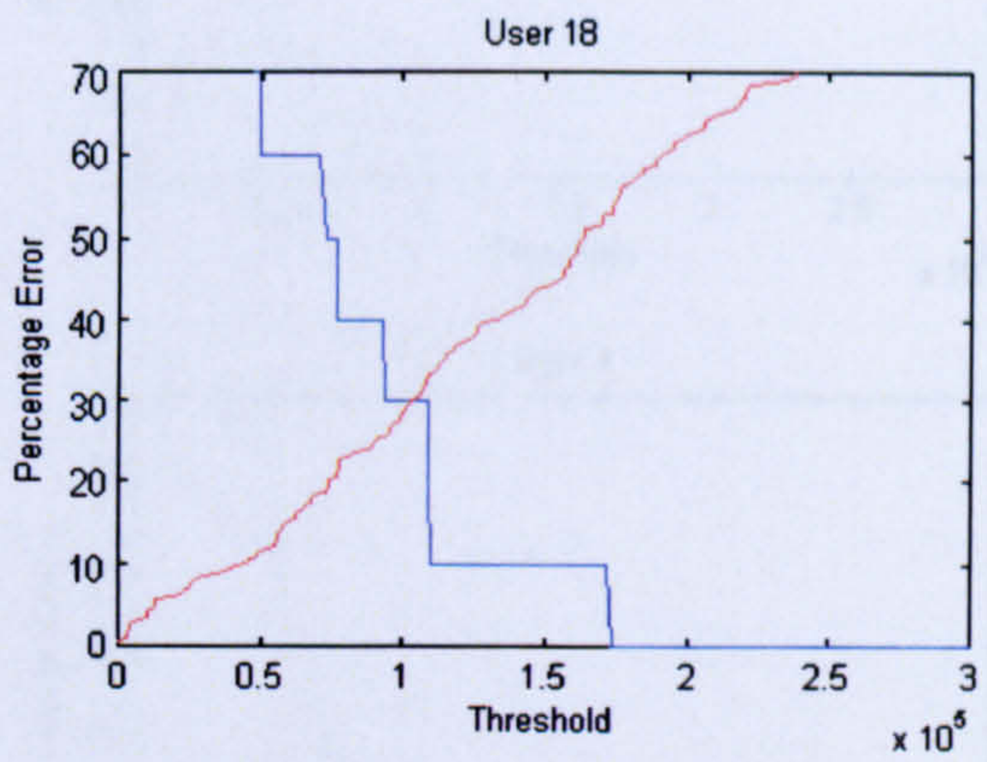

User 13
Percentage Error
Threshold
x 10^5

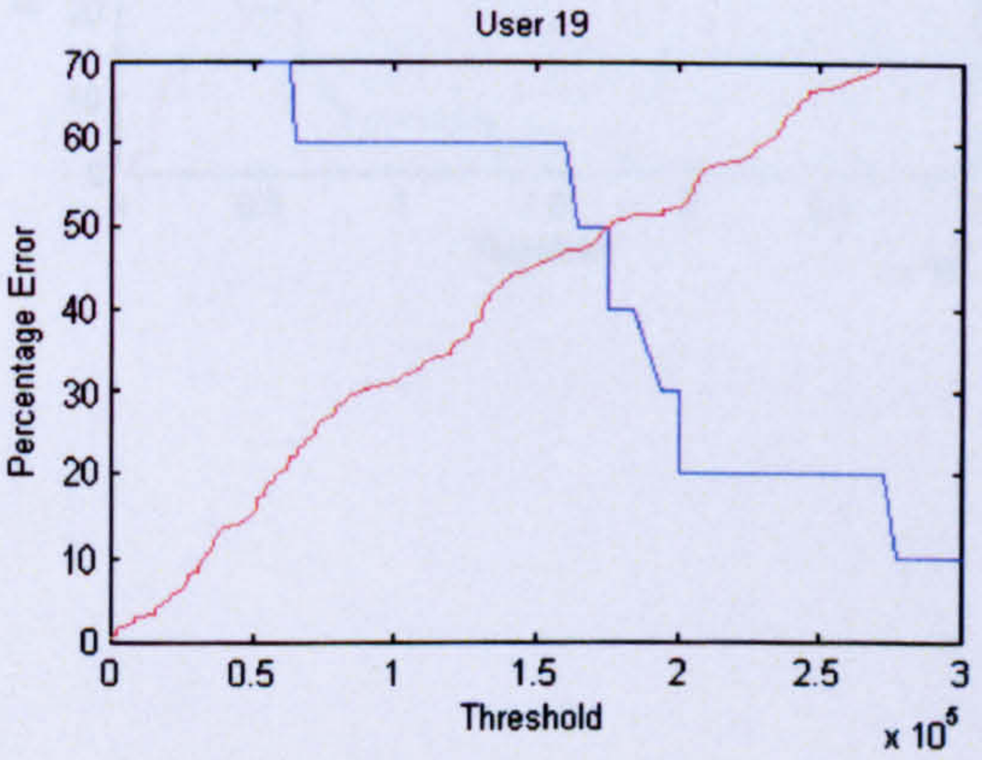

User 14
Percentage Error
Threshold
x 10^5

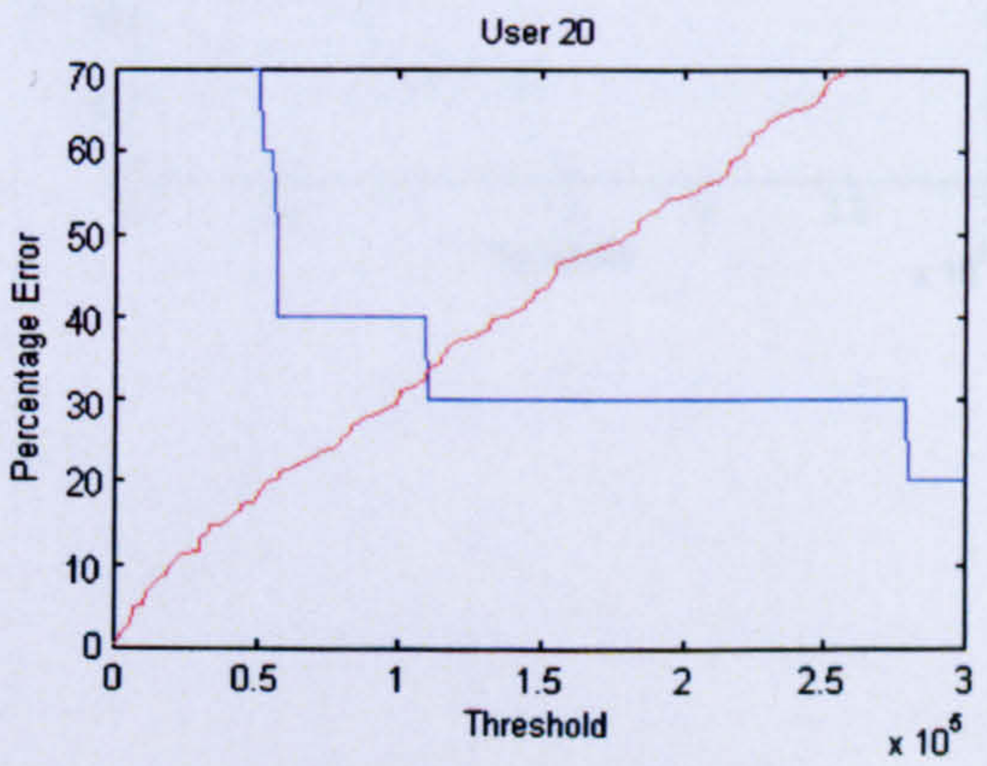

User 15
Percentage Error
Threshold
x 10^5

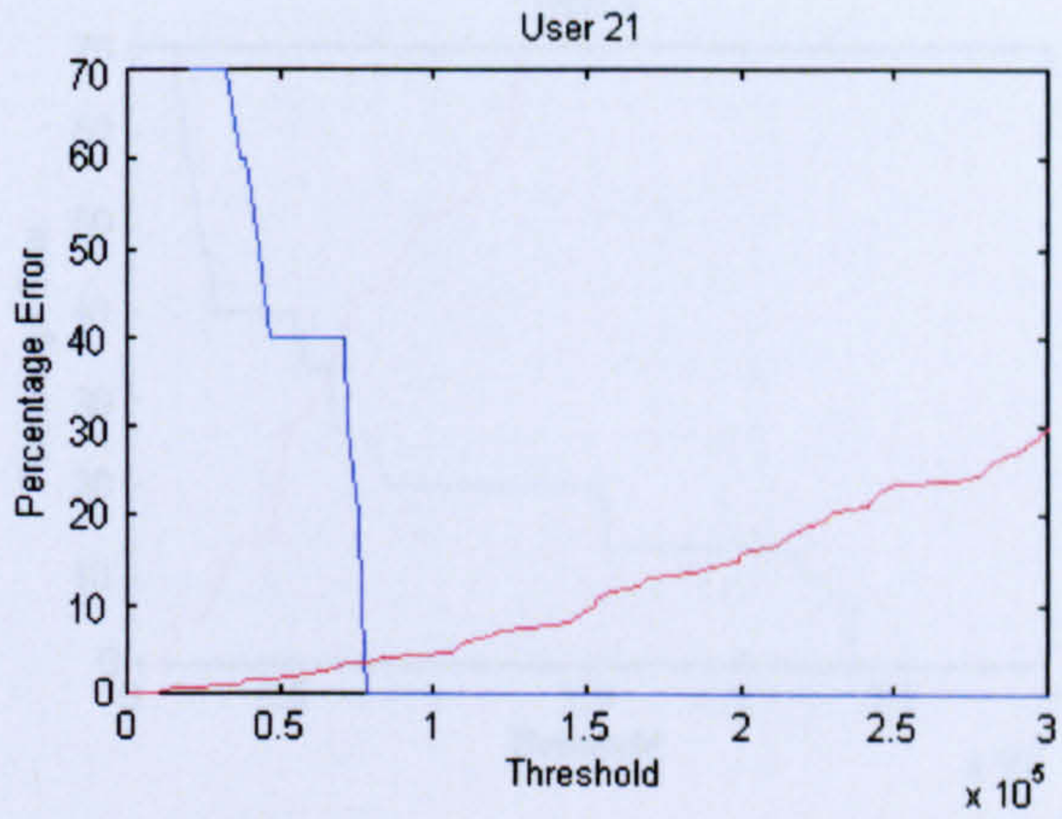

User 16
Percentage Error
Threshold
x 10^5


User 17
Percentage Error
Threshold
x 10^5

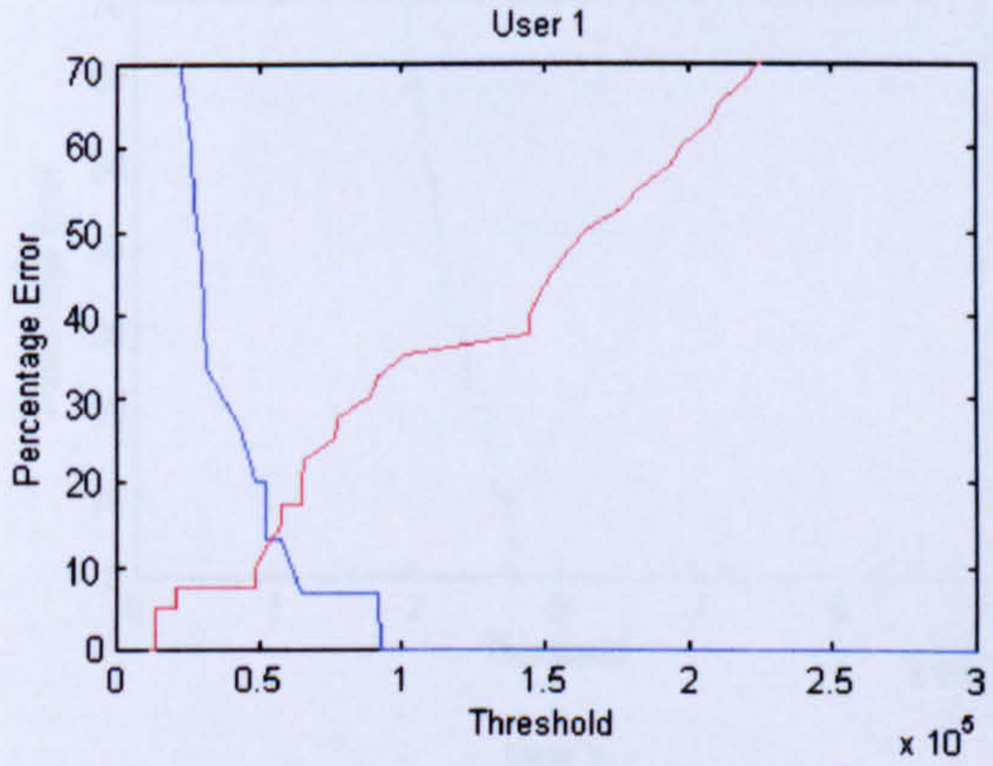

User 18
Percentage Error
Threshold
x 10^5

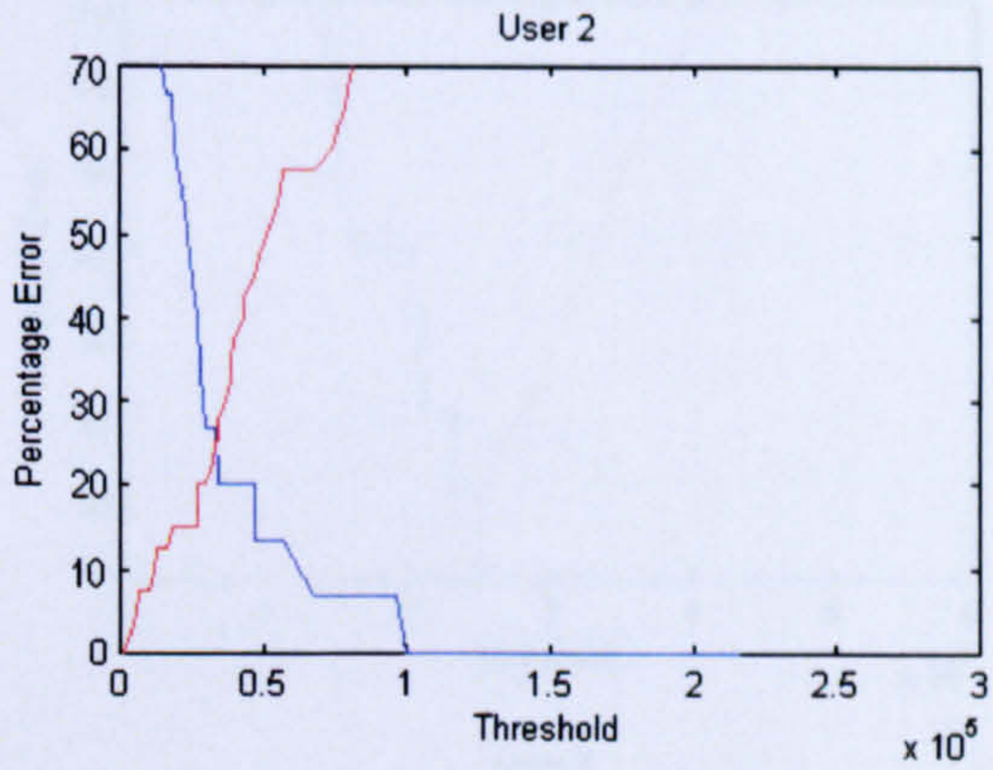

User 19
Percentage Error
Threshold
x 10^5

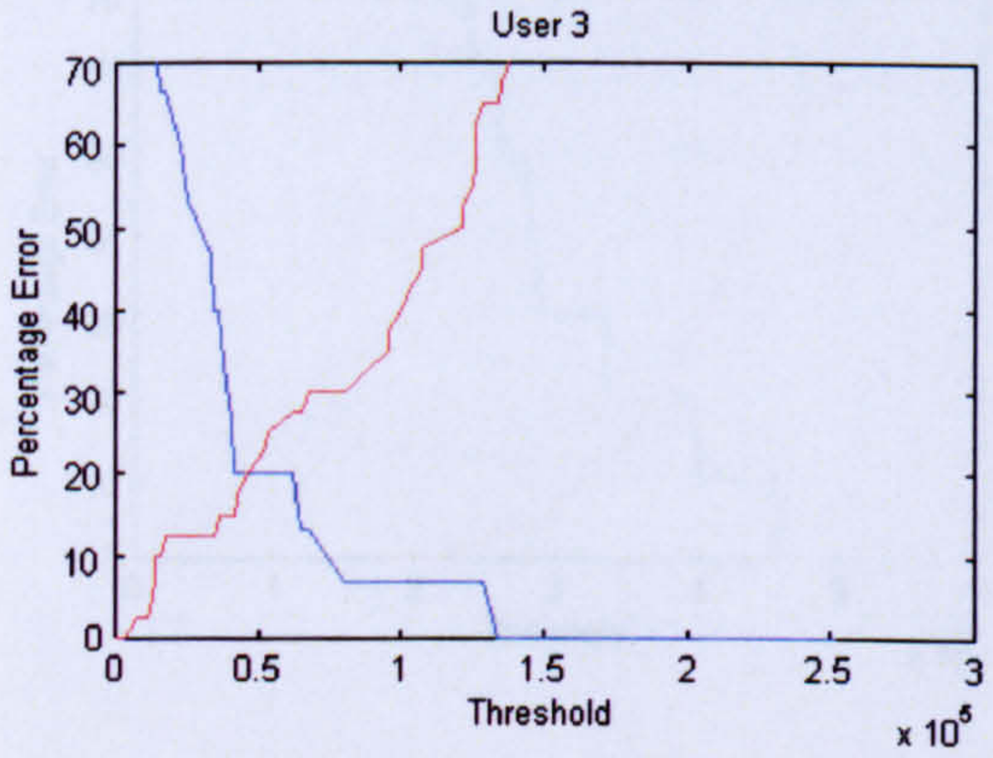

User 20
Percentage Error
Threshold
x 10^5

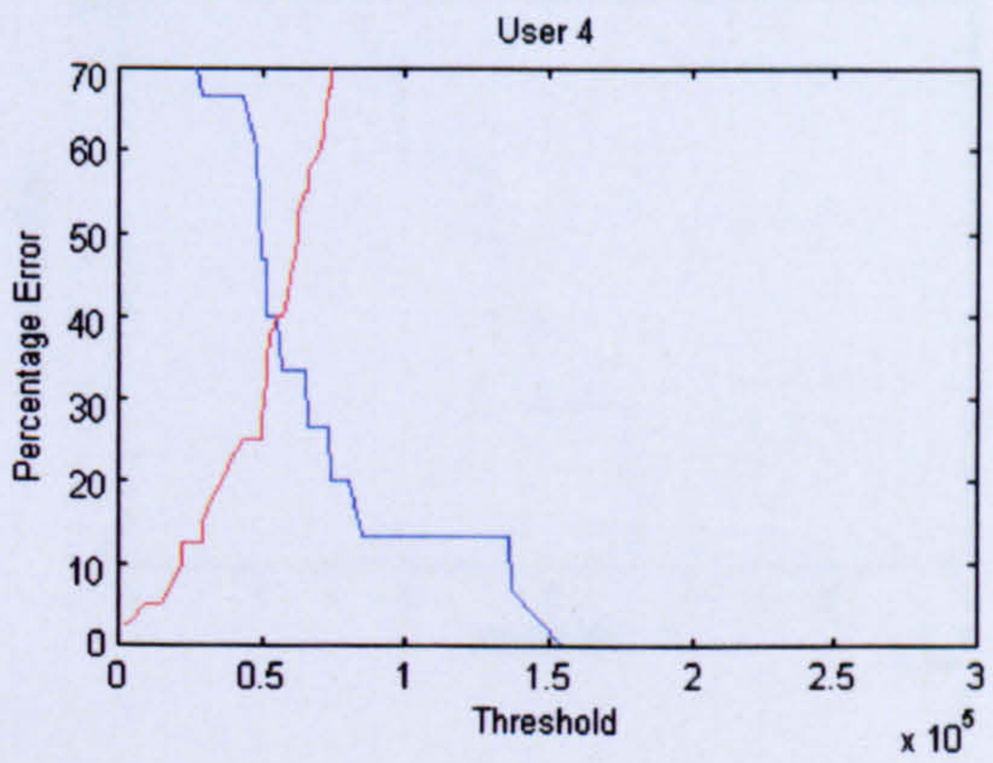


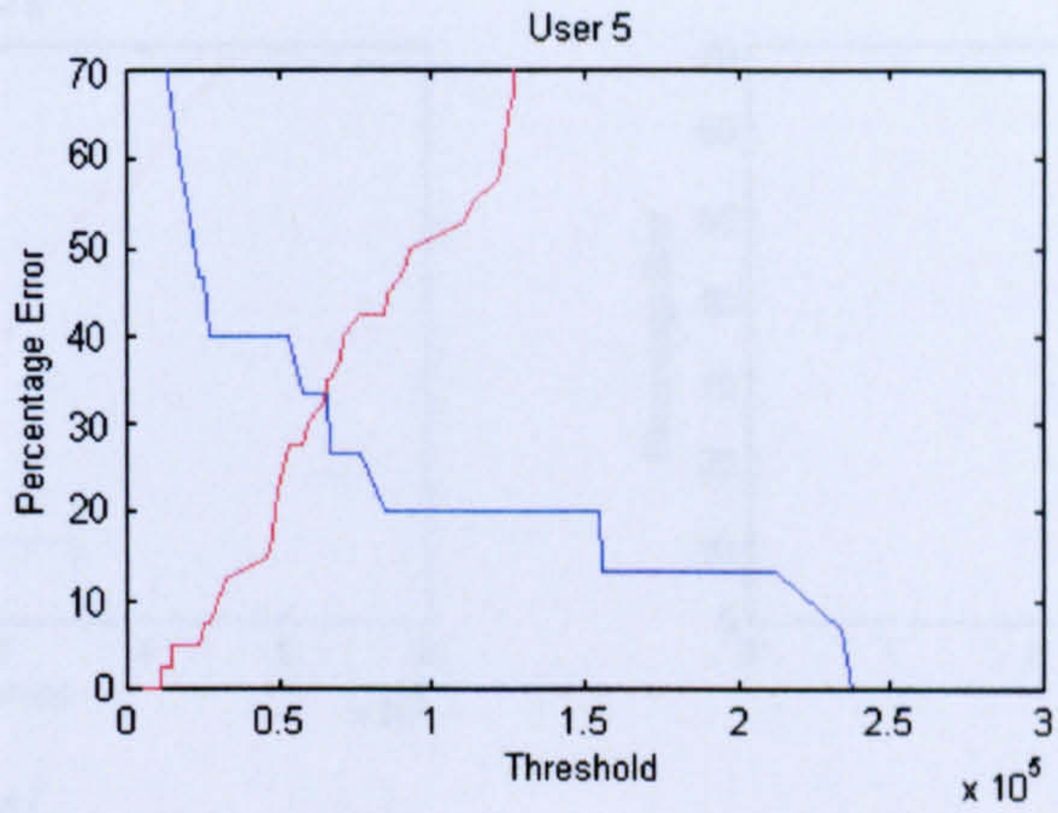
8.3.2 Euclidean Biased Data



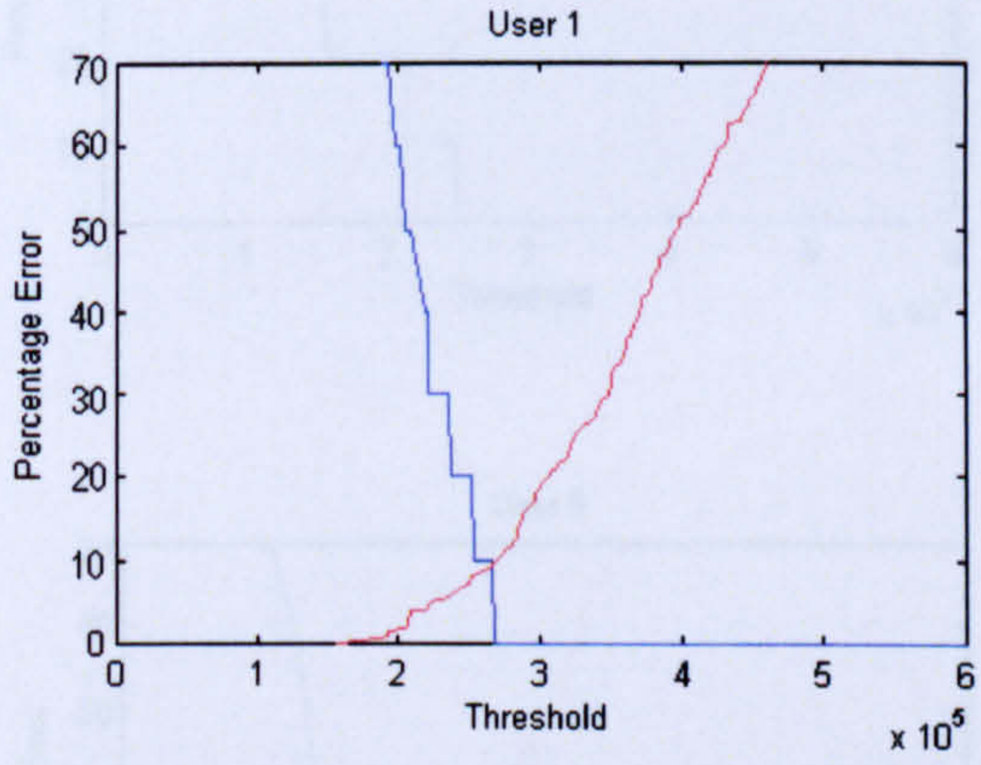


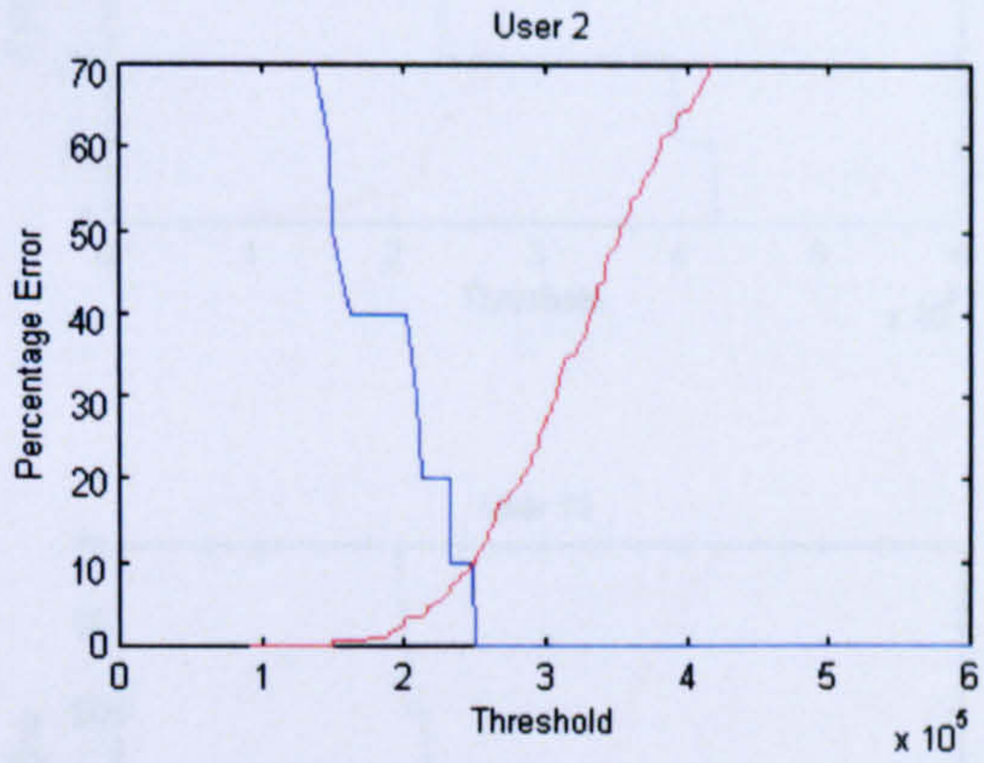


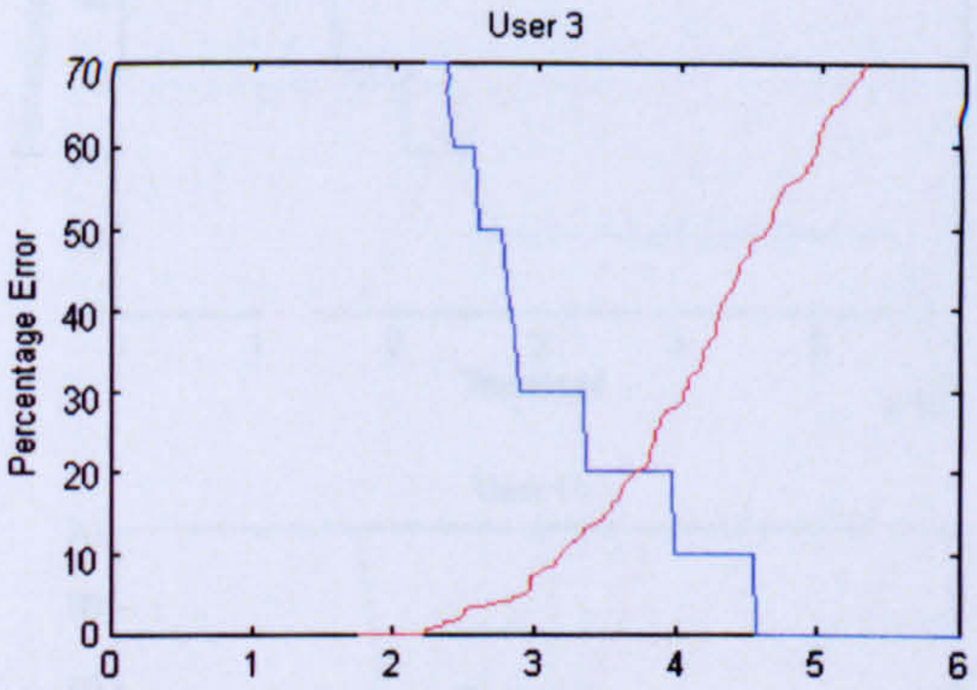


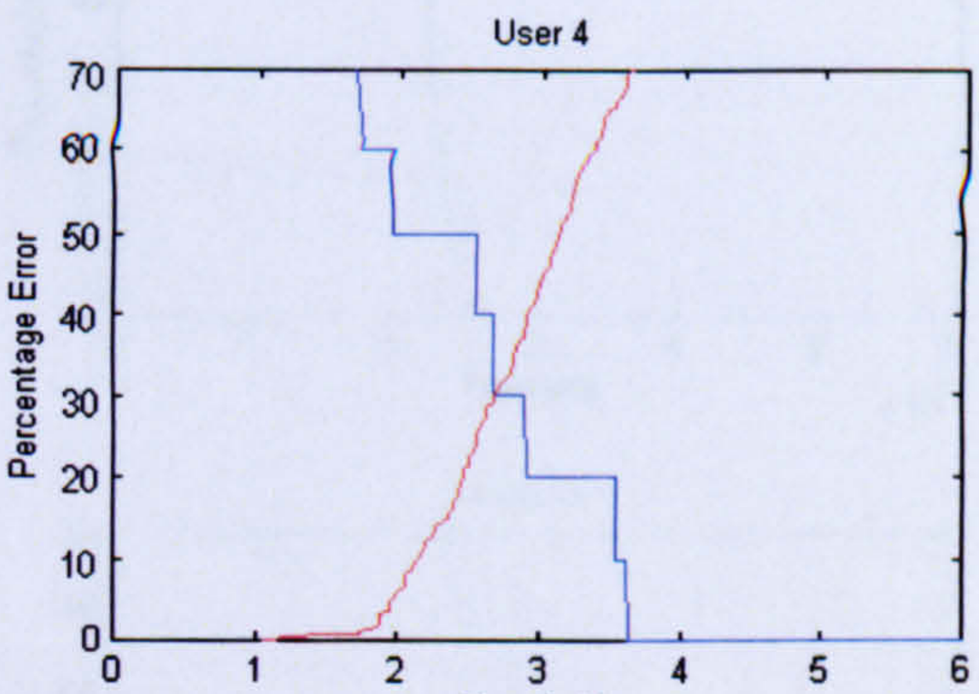


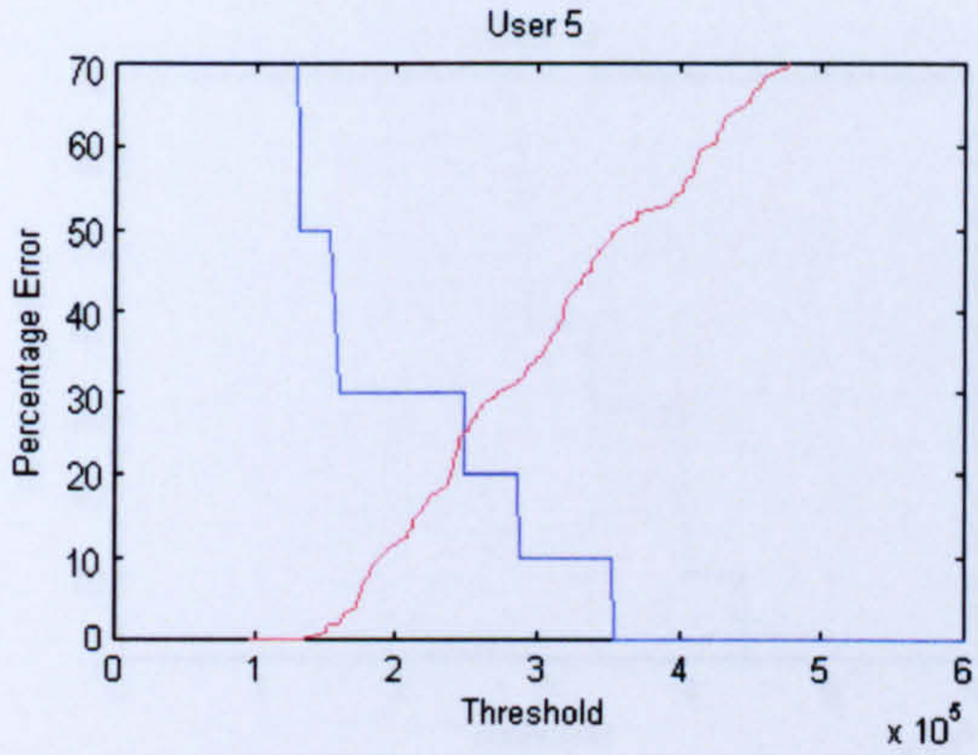
8.3.3 Manhattan Unbiased

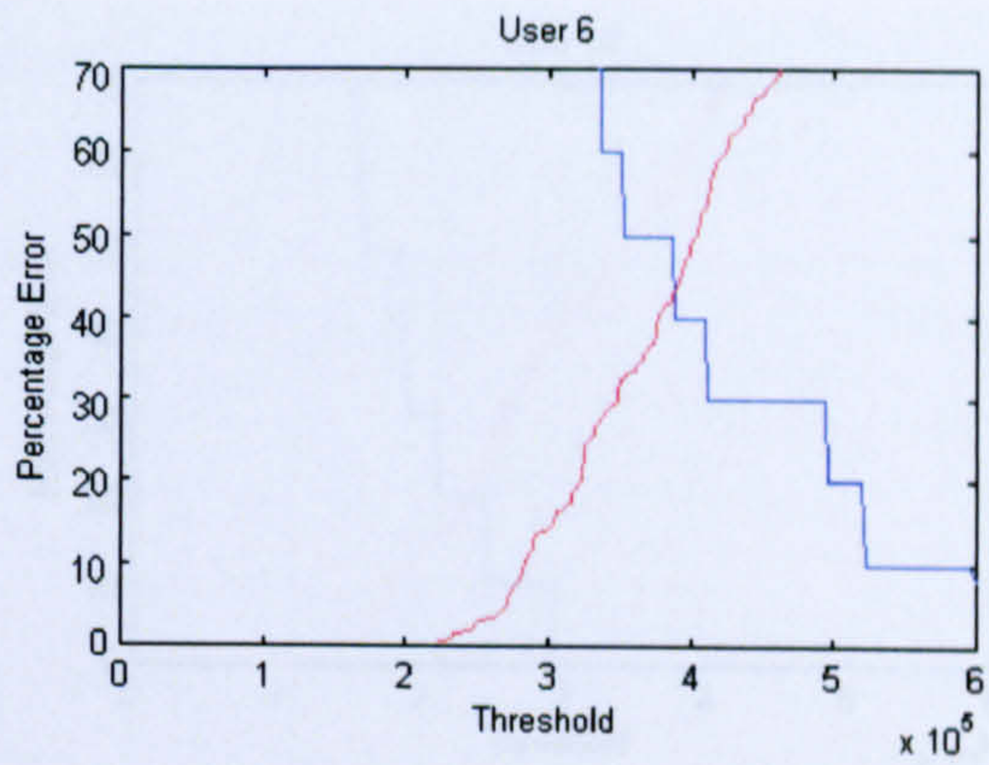


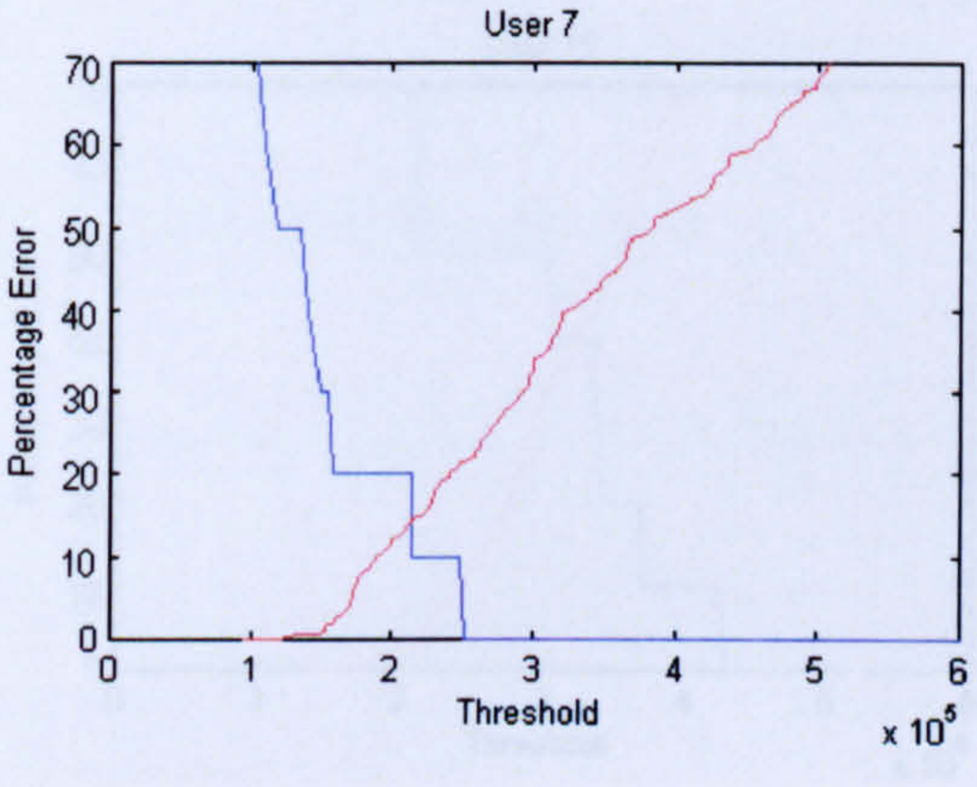


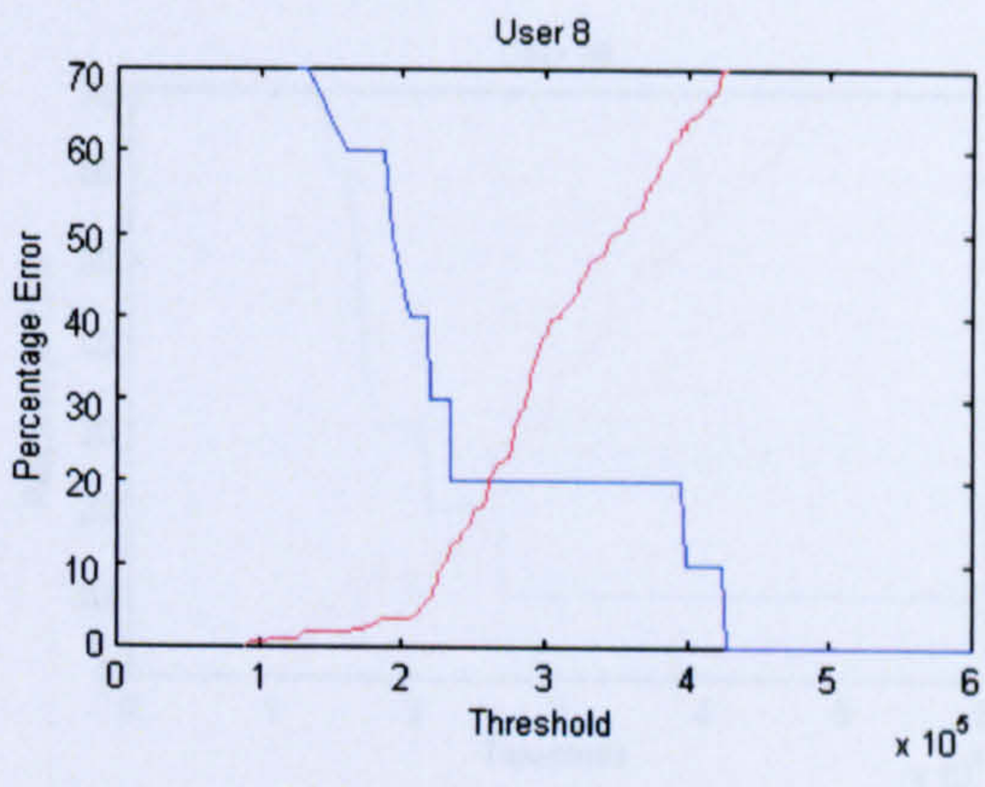


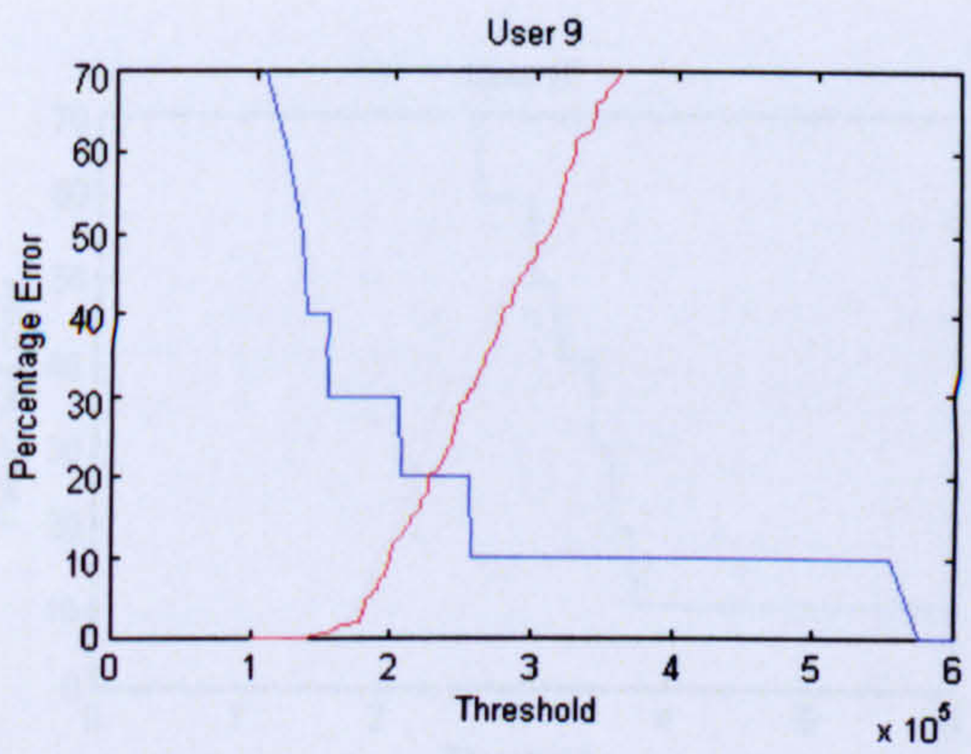


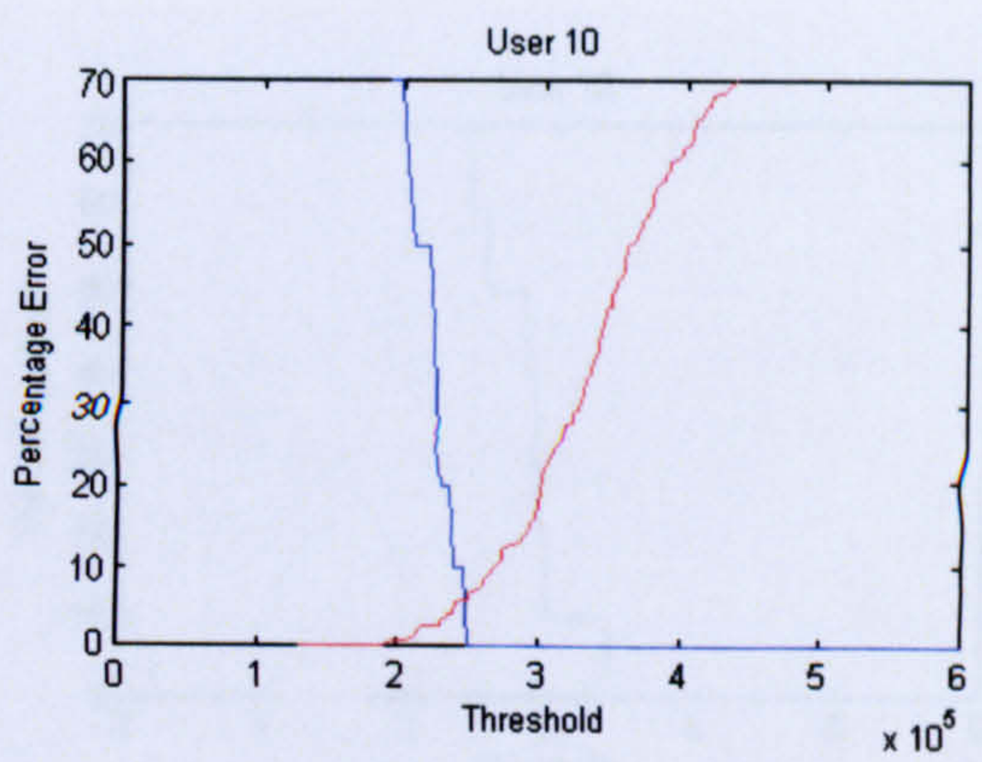

User 5
Percentage Error
Threshold
x 10^5

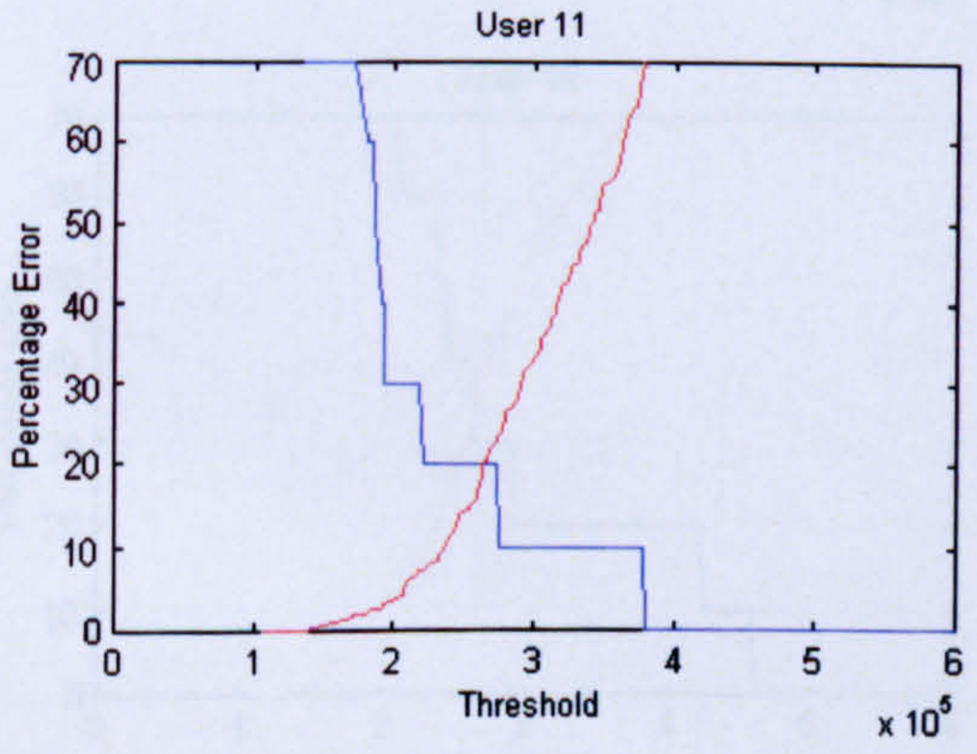

User 6
Percentage Error
Threshold
x 10^5


User 7
Percentage Error
Threshold
x 10^5

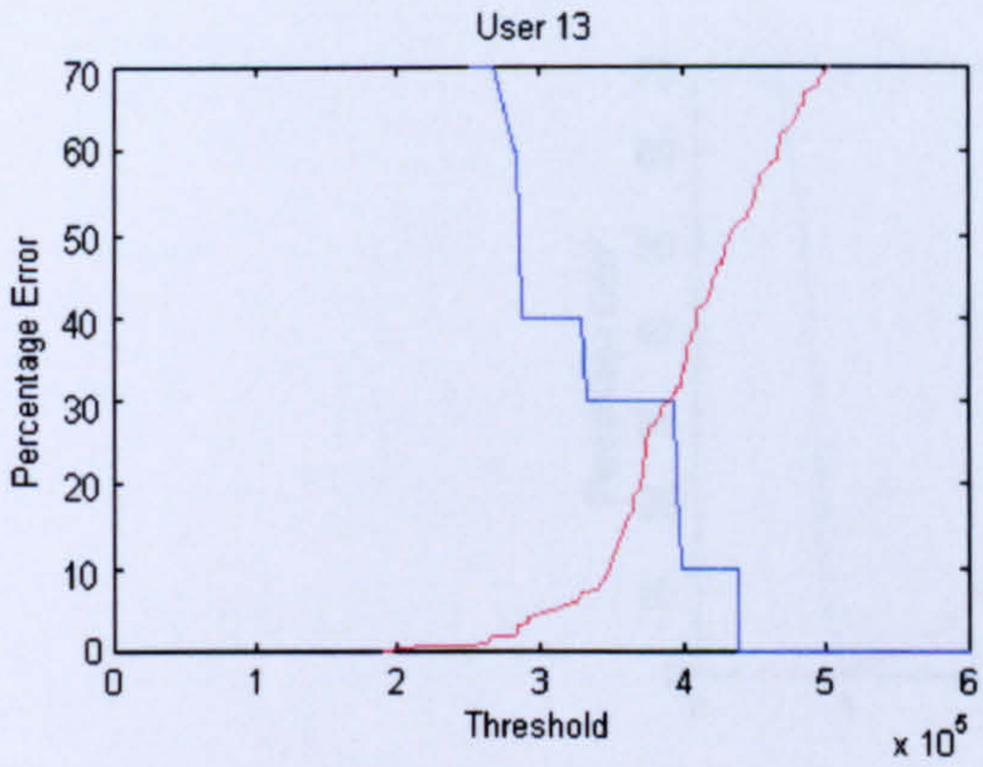

User 8
Percentage Error
Threshold
x 10^5

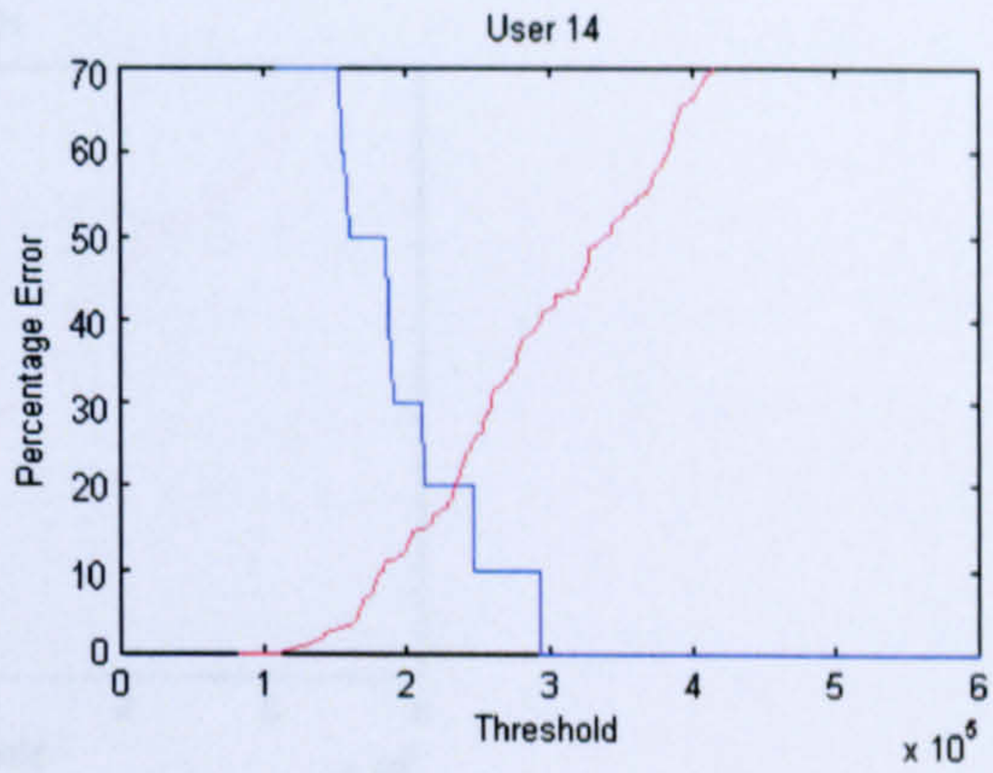

User 9
Percentage Error
Threshold
x 10^5

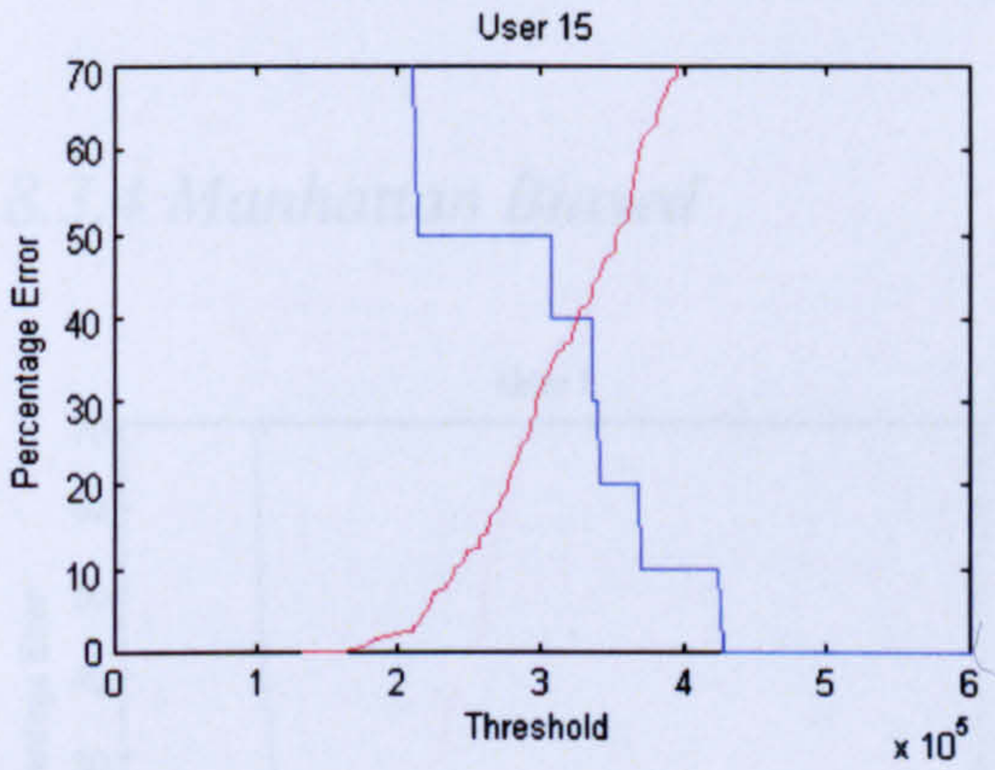

User 10
Percentage Error
Threshold
x 10^5

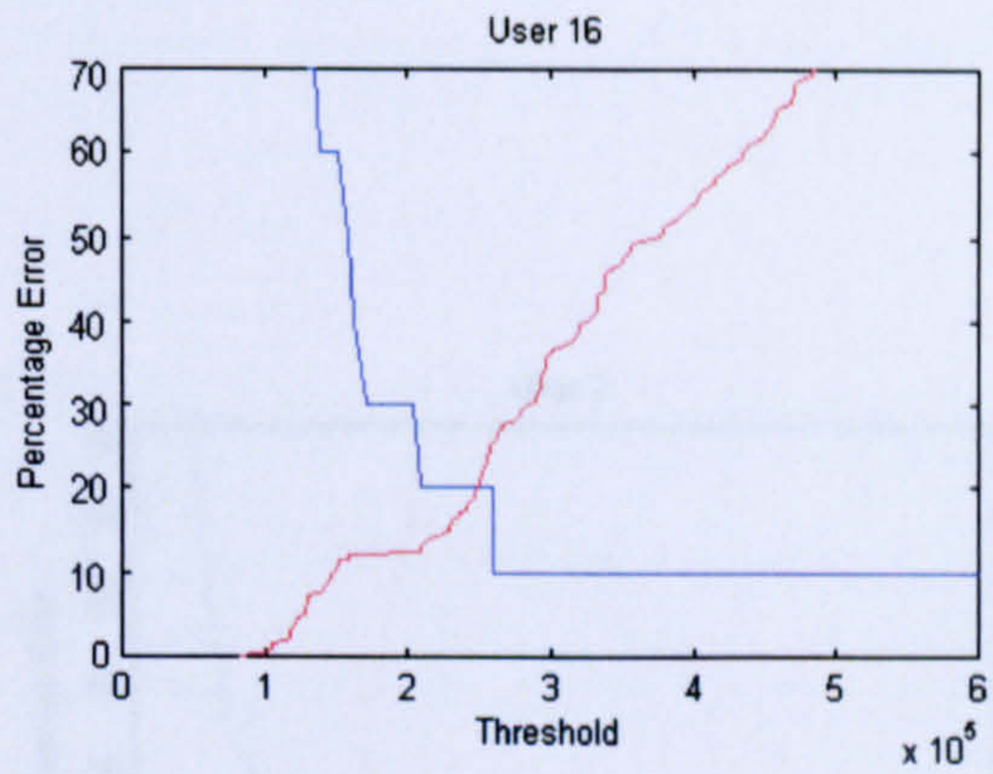

User 11
Percentage Error
Threshold
x 10^5

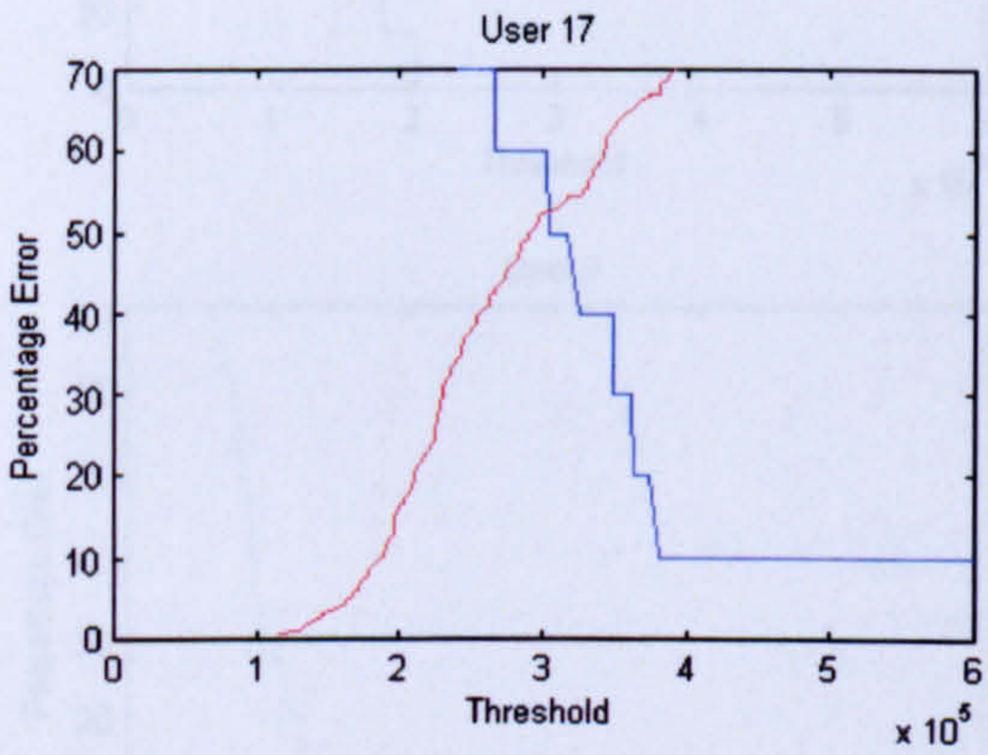

User 12
Percentage Error
Threshold
x 10^5

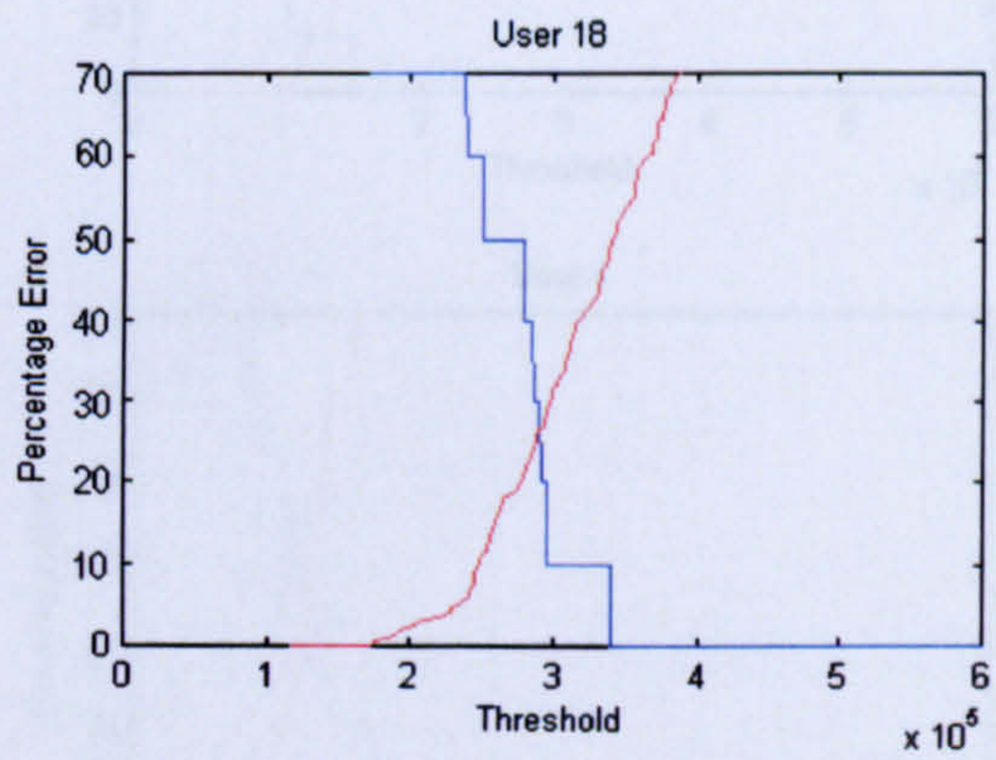

User 13
Percentage Error
Threshold
x 10^5

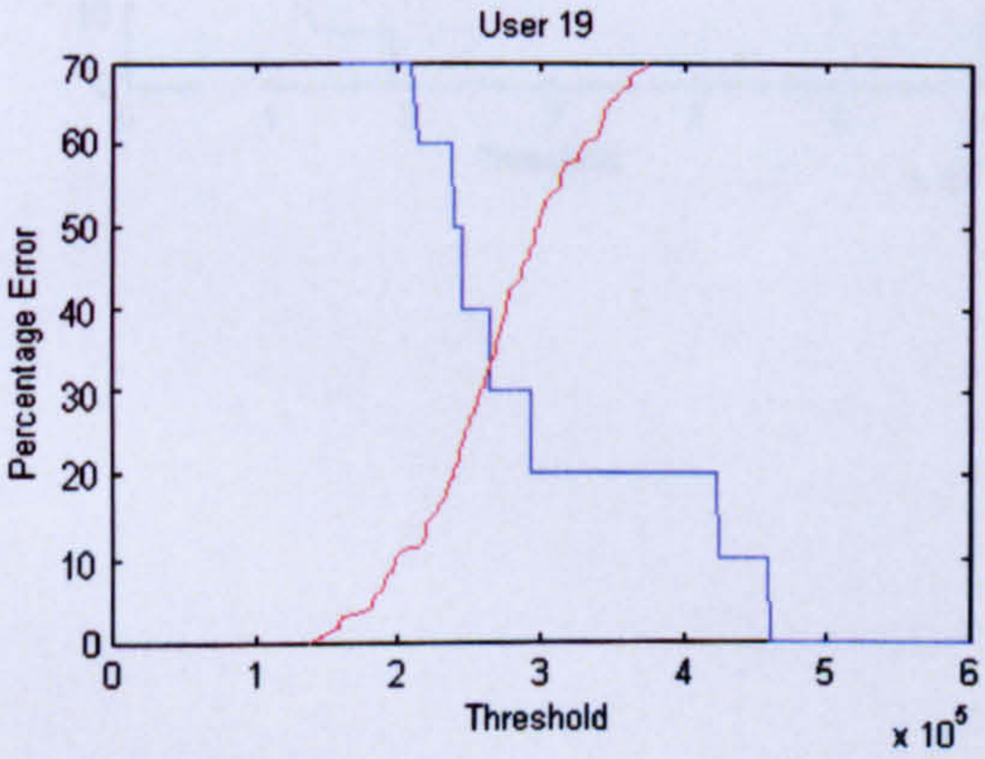

User 14
Percentage Error
Threshold
x 10^5

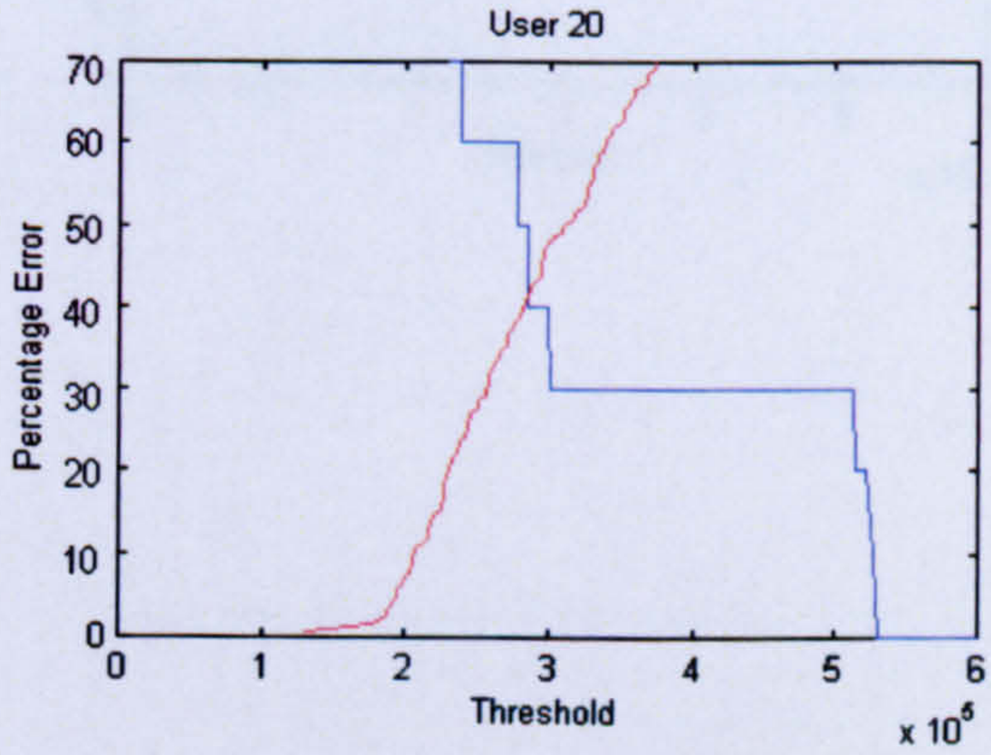

User 15
Percentage Error
Threshold
x 10^5

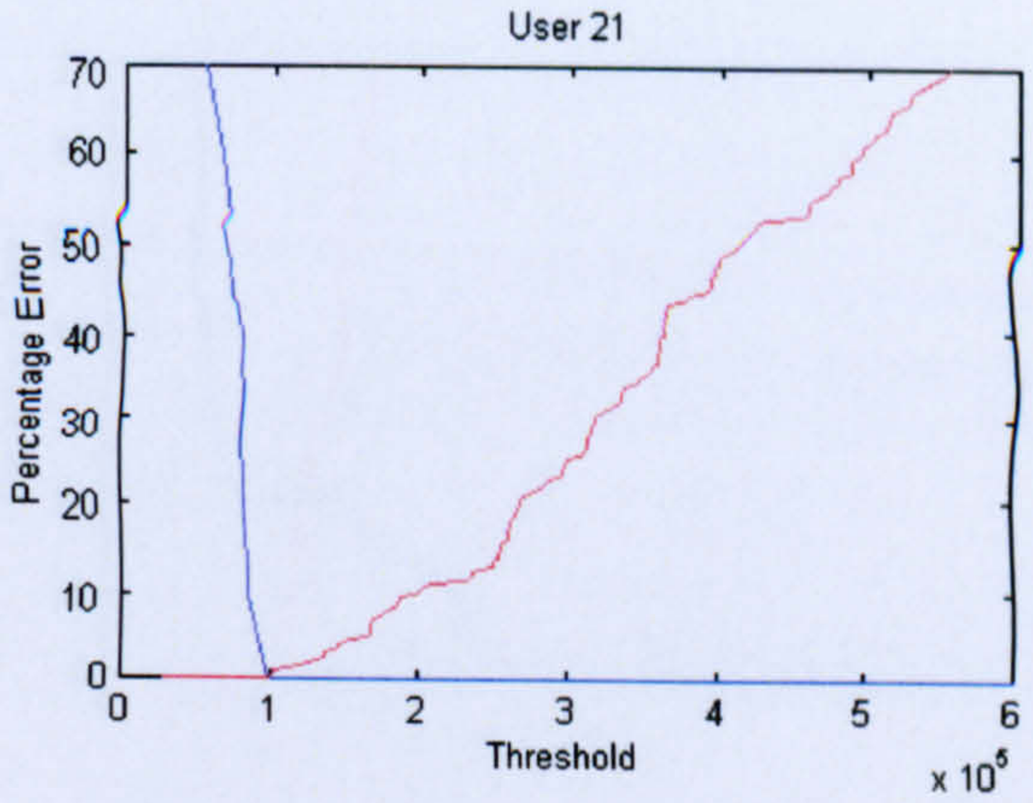

User 16
Percentage Error
Threshold
x 10^5


User 17
Percentage Error
Threshold
x 10^5

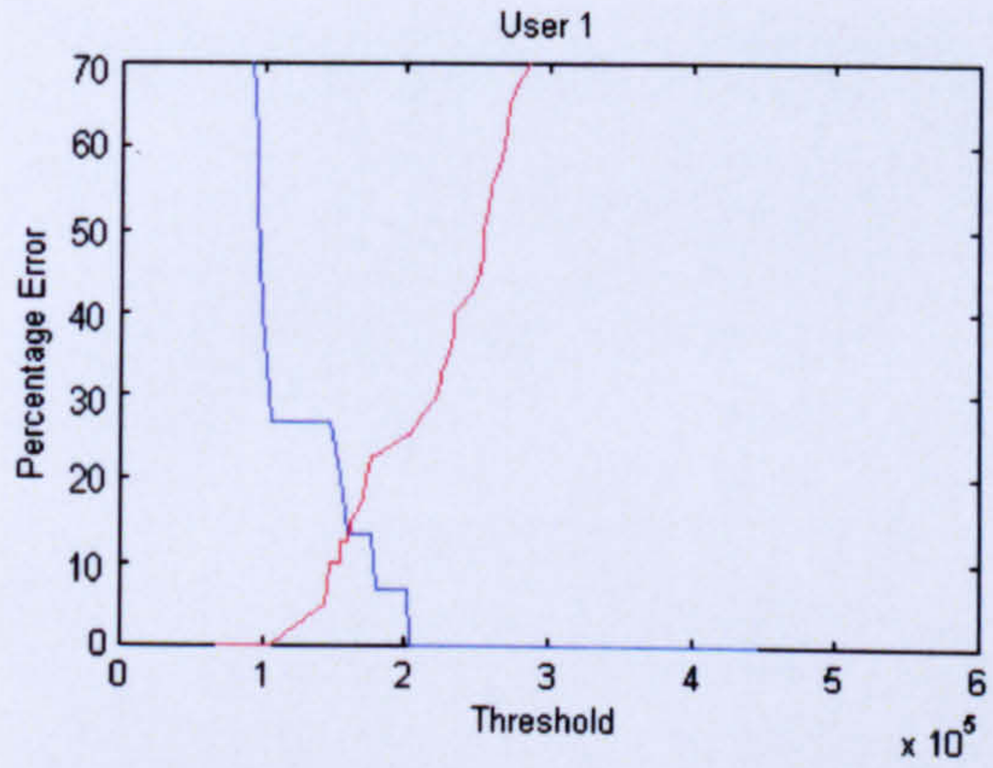

User 18
Percentage Error
Threshold
x 10^5

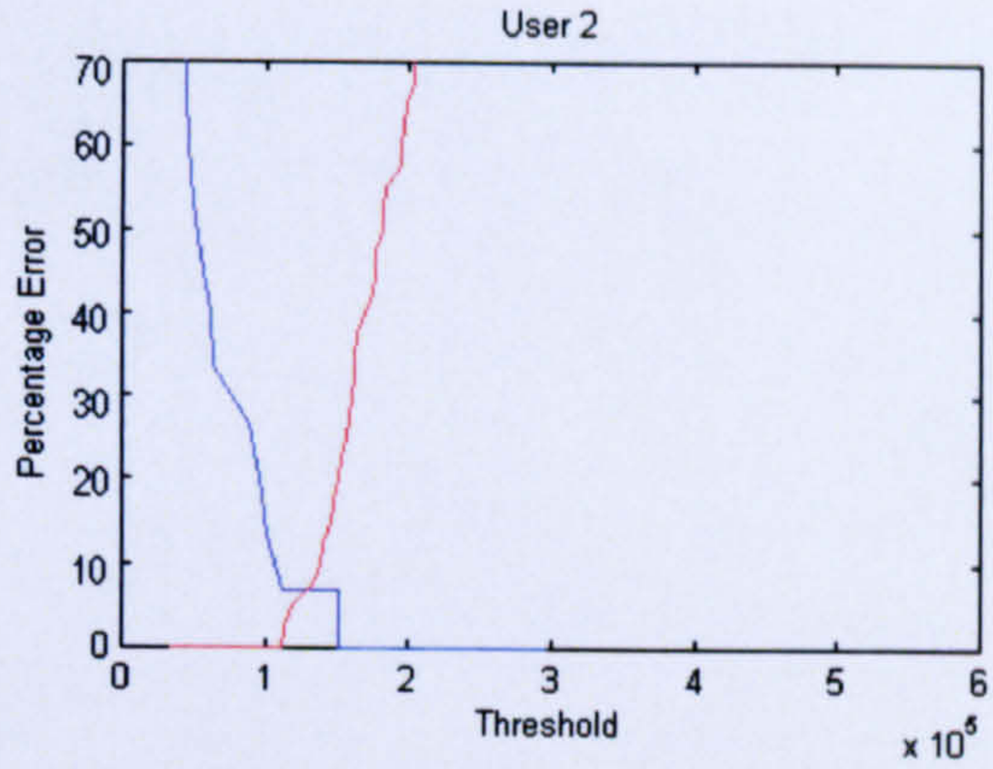

User 19
Percentage Error
Threshold
x 10^5

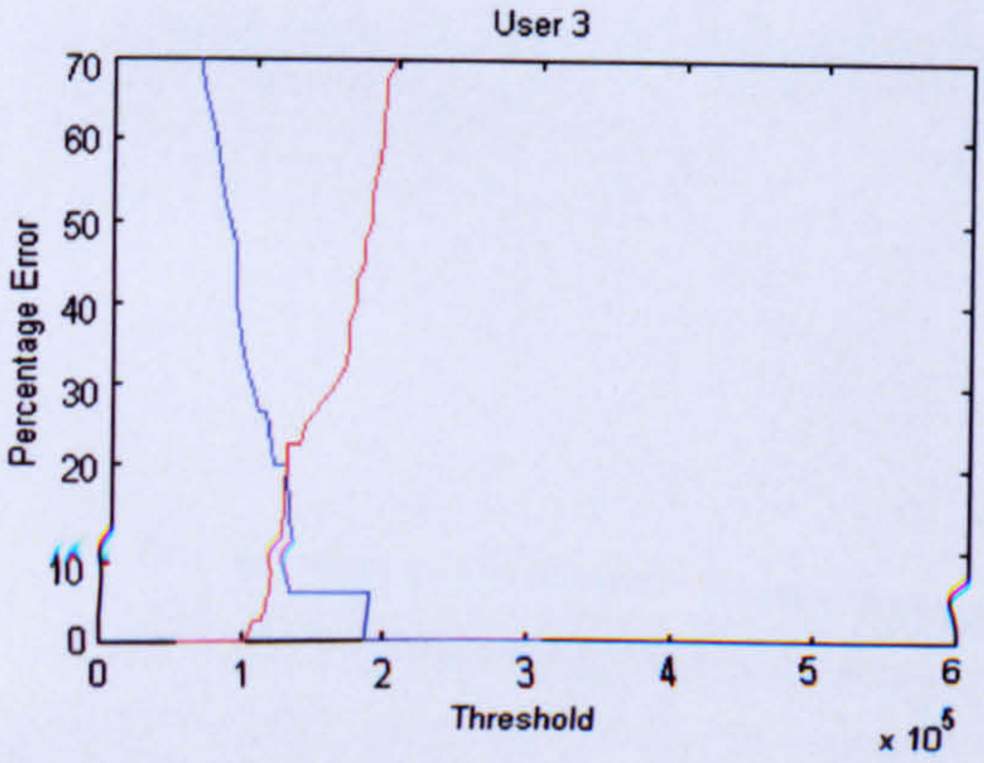

User 20
Percentage Error
Threshold
x 10^5

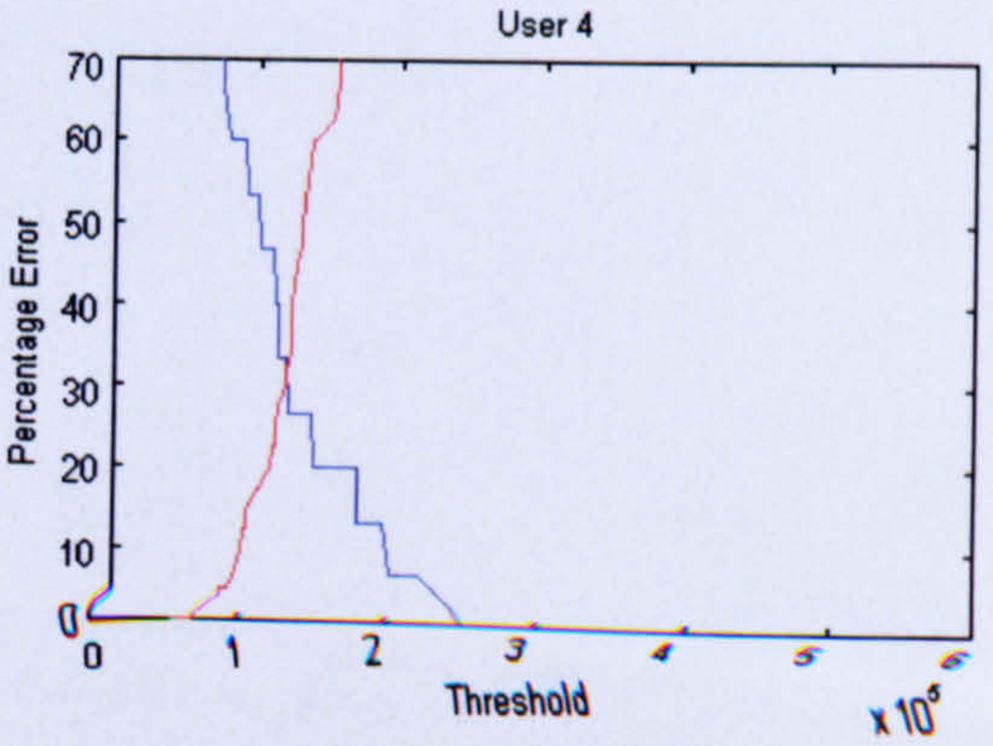


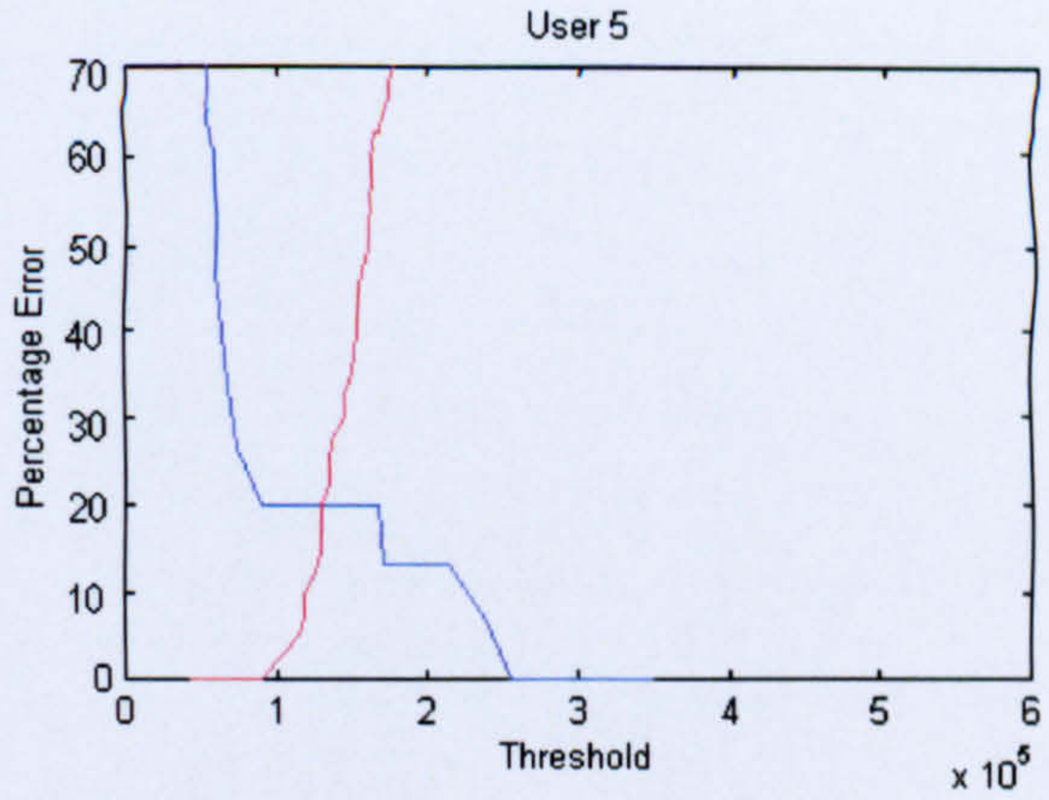
8.3.4 Manhattan Biased










User 5
Percentage Error
70
60
50
40
30
20
10
0
0
1
2
3
4
5
6
Threshold
x 10^5