# MACHINE LEARNING BASED CLASSIFICATION SYSTEM FOR OVERLAPPING DATA AND IRREGULAR REPETITIVE SIGNALS

**SIT WING YEE**

(B.Sc. (Hons) NUS)

**A THESIS SUBMITTED**

**FOR THE DEGREE OF MASTER OF SCIENCE**

**DEPARTMENT OF MATHEMATICS**

**NATIONAL UNIVERSITY OF SINGAPORE**

# Acknowledgements

I would like to thank my supervisor, Dr Mak Lee Onn of DSO, for the opportunity to work on this project, for his generosity, patient guidance and all the time and attention he had been giving me despite his busy schedule.

I am also most grateful to Dr Ng Gee Wah for making it possible for me to embark on this journey. None of this would have been possible without support from him and DSO. It has been a pleasure to work at CCSE under the watch of Prof Chen Yuzong, who has given much valuable advice to better prepare me for the road ahead. My heartfelt gratitude also goes to A/P Bao Weizhu for his support, guidance and assistance in so many ways. I would also like to thank Daniel Ng of DSO, who went through much trouble to allow me to enter this project.

My appreciation goes to my friends and colleagues at CCSE and Mathematics department, who made my experience at NUS a very enjoyable and special one.

This project is sponsored by the NUS-DSO Collaboration under contract No. DSOCL06147.

# Table of Contents


Chapter 1. Introduction ..... 1
1.1. Classification ..... 1
1.2. The Problem ..... 2
1.3. Main Results ..... 3
1.4. Contributions ..... 3
1.5. Sequence of content ..... 4
Chapter 2. Fuzzy ARTMAP Classification ..... 5
2.1. Fuzzy ARTMAP Architecture ..... 5
2.2. Problem of Overlapping Classes ..... 12
2.2.1. Category Proliferation ..... 12
2.2.2. Difficulty of Classification ..... 14
2.3. Methodology ..... 16
2.3.1. Classification and Accuracy Measure ..... 16
2.3.2. Measures to Reduce Category Proliferation ..... 24
2.4. Results ..... 29
2.4.1. Results for UCI Datasets ..... 31
2.4.2. Results for Synthetic Data ..... 34
2.5. Discussion ..... 36
Chapter 3. Signal Sorting by TOA ..... 40
3.1. Existing Method ..... 41
3.1.1. Sequence Search ..... 41
3.1.2. Difference Histogram Method ..... 42
3.2. Implementation of Sequence Search and Histogram Methods ..... 48
3.2.1. Implementation Issues ..... 49
3.2.2. Algorithm for Sequence Search using Bin Interval ..... 50
3.2.3. Problems Encountered ..... 53
3.3. Use of Prior Knowledge ..... 55

3.3.1. Selecting the Tolerance Parameter ..... 57
3.3.2. Selecting the Threshold Parameters ..... 57
3.3.3. Trial Train Construction in Sequence Search ..... 63
3.4. Results ..... 64
3.4.1. Results for Sample with 2 classes ..... 66
3.4.2. Results for Sample with 3 Classes ..... 67
3.4.3. Results for Sample with 4 Classes ..... 69
3.5. Discussion ..... 70
Chapter 4. Conclusion ..... 74

# Summary

Two classification modules in an overall system are looked into – one that does classification for data from overlapping classes using the fuzzy adaptive resonance theory map (fuzzy ARTMAP), and another which sorts repetitive signals, separating them into their respective sources. When faced with overlapping data, fuzzy ARTMAP suffers from the category proliferation problem on top of a difficulty in classification. These are overcome by a combination of modifications which allows multiple class predictions for certain data, and prevents the excessive creation of categories. Signal sorting methods such as sequence search and histogram methods can sort the signals into their respective sequences with a regular interval between signals, but effectiveness of the methods is affected when the intervals between signals in the source are highly deviating. Using available expert knowledge, the signals are effectively and accurately separated into their respective sources.

# List of Tables

Table 1: Results from class by class single-epoch training of 2-D data from Figure 5 .... 27

Table 2: Accuracy of UCI data using different classification methods ..... 30

Table 3: Combinations of modifications..... 30

Table 4: Comparisons between modifications on results for yeast data ..... 32

Table 5: Comparisons between modifications on results for contraceptive method choice data ..... 33

Table 6: Comparisons between modifications on results for synthetic data without noise ..... 34

Table 7: Comparisons between modifications on results for synthetic data with noise ... 35

Table 8: Summary of results using fuzzy ARTMAP with and without modifications..... 36

Table 9: Results for ordered incremental learning using UCI data ..... 37

Table 10: Improvements in performance from using merging ..... 38

Table 11: Example of information used in adaptation of x and k..... 62

Table 12: Values of x and k before and after adaptation ..... 62

Table 13: Class and PRI knowledge of data used..... 65

Table 14: Classes C and I with deviation 2.5% and 5% ..... 66

Table 15: Classes D and E with deviations 14% and 14% ..... 67

Table 16: Classes K, C and H with deviations 0%, 2.5% and 2%..... 67

Table 17: Classes D, E and H with deviations 14%, 14% and 2% ..... 68

Table 18: Classes C, I, D and E with deviations 2.5%, 5%, 14% and 14%..... 69

Table 19: Classes E, F, L and M with deviations 14%, 10%, 7% and 7% ..... 69

# List of Figures

Figure 1: Basic architecture of fuzzy ARTMAP ..... 6

Figure 2: Basic architecture for simplified fuzzy ARTMAP ..... 6

Figure 3: Flowchart for simplified fuzzy ARTMAP training process ..... 8

Figure 4: Flowchart for simplified fuzzy ARTMAP classification process ..... 11

Figure 5: Class distribution and hyperbox weight distribution after learning with $\rho$=0.75. (a) Class distribution of the data from 2 classes (b) Position of hyperboxes after 1 training epoch (c) Position of hyperboxes after training until convergence of training data, which required 9 epochs ..... 13

Figure 6: 2D view of hyperplane separating the hyperbox into two halves. (i) The input pattern lies on side P of the hyperplane (ii) The input pattern lies on the side Q of the hyperplane ..... 18

Figure 7: Shortest distance from the input pattern to the hyperbox when position of input pattern a on side P coincides with $u \wedge a$ ..... 19

Figure 8: Shortest distance from the input pattern to the hyperbox when the position of input pattern a on side P does not coincide with $u \wedge a$ ..... 20

Figure 9: Hyperboxes of different sizes ..... 21

Figure 10: Patterns with more than one predicted class ..... 23

Figure 11: CDIF histograms up to difference level 4 ..... 45

Figure 12: SDIF histograms up to difference level 2 ..... 46

Figure 13: Example of sequence search using bin interval – Search for first signal ..... 51

Figure 14: Example of sequence search using bin interval – Search for signal within tolerance allowance ..... 51
Figure 15: Example of sequence search using bin interval – No signal found within tolerance allowance ..... 52
Figure 16: Example for sequence search using bin interval – Search for next signal after a missing signal is encountered ..... 52
Figure 17: Example for sequence search using bin interval – Selection of next signal based on supposed PRI ..... 53
Figure 18: Difference histogram of sample with higher deviation ..... 55
Figure 19: Graphical view of information drawn from database ..... 56
Figure 20: Position of threshold function if chosen bin is taller than those before it ..... 58
Figure 21: Position of threshold function if chosen bin is shorter than one before it ..... 59
Figure 22: Simplified view of original and desired threshold function ..... 60
Figure 23: Thresholds before and after adaptation ..... 63
Figure 24: Histogram peaks at values less than smallest PRI ..... 72

# Chapter 1. Introduction

## 1.1. Classification

Classification methods have found their way into various applications because of their many uses. They can learn rules and classify new data based on previously learnt examples, speed up processes (especially for large amounts of data), or decision making and diagnosis, where human error and biasness can be avoided by using a classification system [1]. Various methods are available, and they have been utilized in different applications, such as fuzzy adaptive resonance theory map (fuzzy ARTMAP) in handwriting recognition [2], support vector machines in computer-aided diagnosis [3], multi-layer perceptrons in speech recognition [4], etc.

Different classification methods will have their own limitations, which may become more apparent or pronounced with certain types of data or under particular situations. Effective application of these methods will inevitably involve some extent of adaptation. However, many modifications made to the methods tend to introduce drastic changes to the original architecture, or impose much additional computational costs. This may result in some of the initial benefits and strengths of the method to be lost. As such, we are interested in finding ways to overcome the limitations with minimal changes. This can be done by considering a particular given situation, or by considering the characteristics and knowledge of the data involved.

## 1.2. The Problem

This project looks into part of an overall classification system. The entire system consists of different modules, each with a certain objective to attain. Our focus is on two of these modules – the fuzzy ARTMAP classification module and signal sorting module.

The fuzzy ARTMAP module classifies data according to their attribute values. However, some class distributions in the attribute space are overlapping, such that data lying in the overlapping region may belong to either class. Yet during classification, only one class is predicted by the system, leading to a difficulty in classification. In addition, the overlapping classes also lead to the category proliferation problem. There are various existing methods that aim to reduce this problem, but they tend to involve major changes to the fuzzy ARTMAP architecture or introduce considerable computational costs. It is therefore in the interest of this project to find ways to reduce the category proliferation problem and also deal with the classification of data in overlapping classes without significantly changing the architecture, and using minimal additional computational costs.

The signal sorting module deals with repetitive data. Signals from the same source occur at regularly spaced intervals, and the sample consists of signals from various sources. Signal separation methods such as sequence search and difference histograms can be used to sort the signals into their respective sources, but they face limitations when the regular intervals between signals from the same source deviate from the average value. As expert knowledge on the sources is available in the system, it becomes desirable to find a way to incorporate this knowledge into the existing process to improve it without introducing major changes to the original method.

## 1.3. Main Results

To deal with the problem of overlapping classes in fuzzy ARTMAP, we modify the classification process so that more than one class can be predicted for certain data. The decision control of which classes to predict is built into the system and there is no need for a separate parameter to be selected by the user. The extent of category proliferation is eased significantly with the introduction of some modifications, used in conjunction with an existing proposed variation called match tracking - (abbreviated as MT-, where the dash is read as 'minus'). These modifications work well even for large datasets with higher amounts of noise.

Expert knowledge on the signals and their sources is available for the signal sorting module in the form of a database. The information in the database is not exclusive and consists of irrelevant information as well, but can still improve the performance of the existing methods on our data. The selection of parameters for the difference histogram method is automated and appropriately chosen, while the sequence search process is completed with greater certainty and accuracy by referring to the database. Although the time taken by the signal sorting method is longer than before, the overall process is actually faster due to the elimination of the need to scan for the right parameter values.

## 1.4. Contributions

The contribution of the work on the fuzzy ARTMAP problem of overlapping classes is to offer a simple way that is straightforward to implement, which can overcome the difficulty in classifying data from overlapping classes, as well as the problem of category proliferation. It does not introduce much additional computational costs and the whole training and classification process is completed much faster than before for such overlapping data. A paper containing this work has been submitted [5].

In signal sorting, expert knowledge is successfully incorporated into the existing methods without making drastic changes. It enables signals to be separated effectively and accurately. There is a reduced need to scan values for user-selected parameters, which are difficult to determine since they vary with different data. This effectively reduces the total time needed for the complete process.

## 1.5. Sequence of content

The thesis is arranged as follows. Chapter 2 describes the work done for the fuzzy ARTMAP module. It will first introduce the classification process and the problem of overlapping classes. With an understanding of how the method works and how the problems arise, modifications can be made. The results and discussions following the testing of the modifications are then shown.

Chapter 3 focuses on the signal sorting module. It illustrates the existing methods and elaborates on the problems that are encountered upon implementation of the methods on actual data. Expert knowledge that is available will be used in overcoming the problem, so the format of the knowledge used is first described, followed by the way it can be used to improve on the existing methods. Results and discussions are then presented, as well as the potential concerns with incorporating expert knowledge into the method.

Finally, Chapter 4 concludes the work done in the project and the possible future directions to further the investigations conducted and results obtained here.

# Chapter 2. Fuzzy ARTMAP Classification

Fuzzy adaptive resonance theory map (fuzzy ARTMAP) is a supervised clustering algorithm that can be used for classification tasks. It has many strengths that make it very appealing, such as incremental learning as new data becomes available [6], fast learning dynamic neuron commitment , and the use of few training epochs to achieve reasonably good performance accuracy [7]. Together with various modifications, fuzzy ARTMAP has performed well when applied to areas such as radar range profiles [8], online handwriting recognition [9], classification of natural textures [10], genetic abnormality diagnosis [11], wetland classification [12], etc. However, fuzzy ARTMAP suffers from the category proliferation problem [13], which is a drawback that is of concern to us. This will be further investigated in the following sections.

## *2.1. Fuzzy ARTMAP Architecture*

The overall structure of fuzzy ARTMAP consists of two adaptive resonance theory (ART) modules – $ART_a$ and $ART_b$, and a mapping field called the MAP module (see Figure 1). $ART_a$ and $ART_b$ cluster patterns in the input space and output space respectively. Clusters from $ART_a$ are mapped to $ART_b$ through the mapping field.

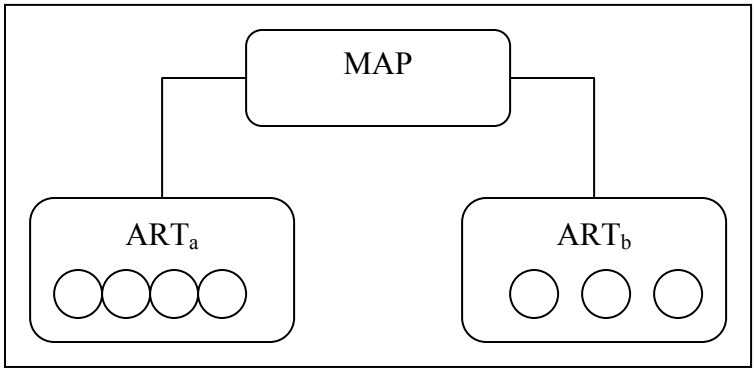


**Figure 1: Basic architecture of fuzzy ARTMAP**

For classification problems, each input pattern is mapped to an output class, so $ART_b$ becomes redundant. We can remove the $ART_b$ module and map categories from $ART_a$ directly to their respective classes in the MAP field (Figure 2). This simplified fuzzy ARTMAP was introduced by Kasuba in [14] will be used in this project. More details of the algorithm can be found in [15] and [16].

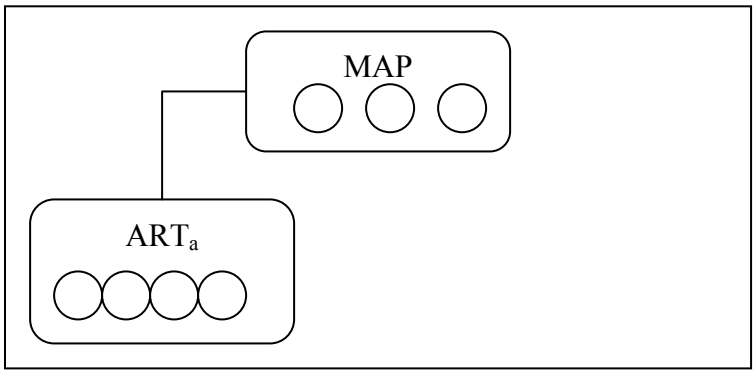


**Figure 2: Basic architecture for simplified fuzzy ARTMAP**

Input data consists of vectors representing the attribute values of each sample. The values are scaled such that they are in the range [0,1]. Before being presented to the network, the input data undergo complement coding, such that an input

$$a = (a_1, ..., a_M)$$

with M attributes will be represented as a vector

$$\begin{aligned} I &= (a, 1-a) \\ &= (a_1, ..., a_M, 1-a_1, ..., 1-a_M). \end{aligned}$$

The length of the vector will then be doubled.

The ART module consists of nodes which can cluster similar input patterns together, and all the patterns clustered by the same node will be mapped to the same class, although there may be more than one node mapped to the same class. These nodes are commonly referred to as categories, and they are represented by their own weight vectors. The weights of a category is given by

$$\begin{aligned} W &= (u, 1-v) \\ &= (u_1, ..., u_M, 1-v_1, ..., 1-v_M), \end{aligned}$$

where $u = (u_1, ..., u_M)$ and $v = (v_1, ..., v_M)$, where $u_i, v_i \in [0,1]$. Geometrically, the category can be represented as a hyperbox in M-dimensional hyperspace, with $u$ and $v$ representing the lower and upper endpoints of the hyperbox respectively. Therefore the categories are also often simply referred to as hyperboxes.

A simple basic idea of fuzzy ARTMAP classification is as follows: during training, when an input pattern in presented, the hyperbox nearest to the input point in hyperspace will code (or cluster) that point if it is mapped to the same class as the rest of the points coded by the same hyperbox. In order to code that input point, the hyperbox grows just enough to contain it. Then when an unknown input pattern is presented during classification, the hyperbox nearest to it in hyperspace will code it, so the output class will be the same as all the other points coded by that same hyperbox.

With a brief overall idea in mind, we shall now take a closer look at the algorithm of training and classification of the fuzzy ARTMAP. The following operators will be used in the algorithm.

The fuzzy min $\wedge$ and max $\vee$ operators are defined as follows:
For vectors $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_n)$,

$$A \wedge B = (\min(a_1, b_1), ..., \min(a_n, b_n)),$$

$$A \vee B = (\max(a_1, b_1), ..., \max(a_n, b_n)).$$

The size $|\bullet|$ of any weight or input vectors is defined as $|A| = |a_1| + |a_2| + ... + |a_n|$.

## Training

Input patterns are presented to the network one at a time and the full presentation of the whole set of training input is known as one epoch. For every input pattern that is presented after complement coding has been carried out, the network learns by following the process given in the flowchart in Figure 3.

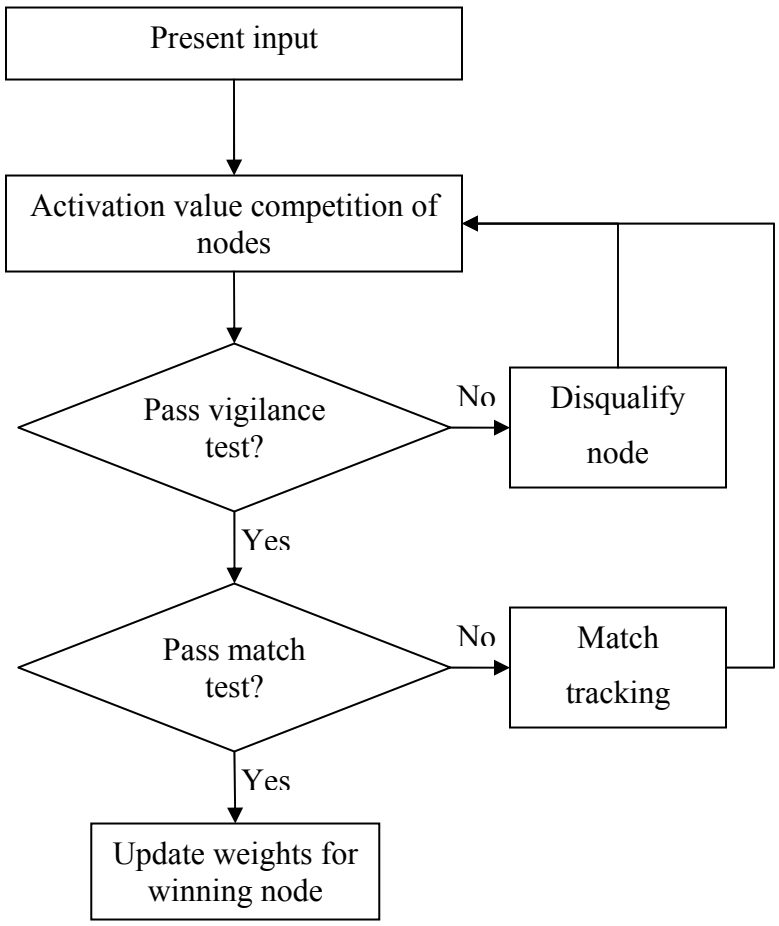


**Figure 3: Flowchart for simplified fuzzy ARTMAP training process**

When the r$^{th}$ input pattern $I_r$ is presented, all the categories undergo competition based on their activation values. The activation value for category j with weights $W_j$ is defined as

$$T_j = \frac{\left| I_r \wedge W_j \right|}{\alpha + |W_j|},$$

where $\alpha$ is a small positive value called the choice parameter. The activation value is a measure of how close the input data is to each of the existing hyperboxes. The choice parameter biases the measure towards smaller hyperboxes in case of a tie in value between two hyperboxes of equal distance from the input data. The use of activation value for competition among the nodes can be understood as clustering together the input patterns which are most similar in terms of their attribute values. The node with highest activation value will be the one which clusters input patterns with highest similarity to the given input pattern.

Based on the competition of activation values, the node or category with highest value will be the winner. This winning category with weights $W_{max}$ will then undergo a vigilance test

$$\frac{\left| I_r \wedge W_{max} \right|}{\left| I_r \right|} \geq \rho,$$

where $\rho \in (0,1)$ is known as the vigilance parameter. This test is a measure of how much the hyperbox has to grow in order to contain the input pattern. A hyperbox that is already very large or one that is far from the input pattern will be more likely to fail the vigilance test, and this vigilance parameter is a restriction on the size of the hyperbox. If the hyperbox category fails the test, one with next highest activation value is considered, until one which passes the vigilance test is found.

The match test is a check performed on the class of the winning category against the corresponding output class of the input pattern $I_r$. A class-mismatch triggers off a match tracking process, whereby the current category is disqualified and the remaining categories compete based on activation value again. The procedure is repeated but the vigilance parameter is temporarily raised for this input-output pair to

$$\frac{\left| I_r \wedge W_j \right|}{|I_r|} + \varepsilon,$$

where $\varepsilon$ is usually a small positive value.

But if the category passes the vigilance test and the match test, it will code the input pattern $I_r$ and its weights are updated according to

$$W_j \leftarrow W_j \wedge I_r .$$

The new weights are given as

$$\begin{aligned} W_j^{new} &= W_j \wedge I \\ &= (u, 1-v) \wedge (a, 1-a) \\ &= (u \wedge a, (1-v) \wedge (1-a)) \\ &= (u \wedge a, 1-(v \vee a)). \end{aligned}$$

Geometrically, the new hyperbox after growing just enough to contain the input point will have the lower and upper end points $(u \wedge a)$ and $(v \vee a)$ respectively.

Fuzzy ARTMAP adopts the winner-take-all strategy, so only this winning category has its weights updated to include the input pattern. The training process is stopped when a maximum number of epochs are reached, or when there are no more changes to the weights of the categories within a single epoch.

Over the training process, new categories may need to be created at certain times. In the beginning when training first started and there are no nodes, a new node is created to start coding the first input pattern. The weights of a new category are given by

$$(W)_i = 1, \qquad i = 1, ..., 2M .$$

If all categories fail the vigilance test, or if match tracking fails to return a winning category, a new one will be created.

## Classification

The classification process of the fuzzy ARTMAP is similar to the training process. The flowchart in Figure 4 depicts the classification process.

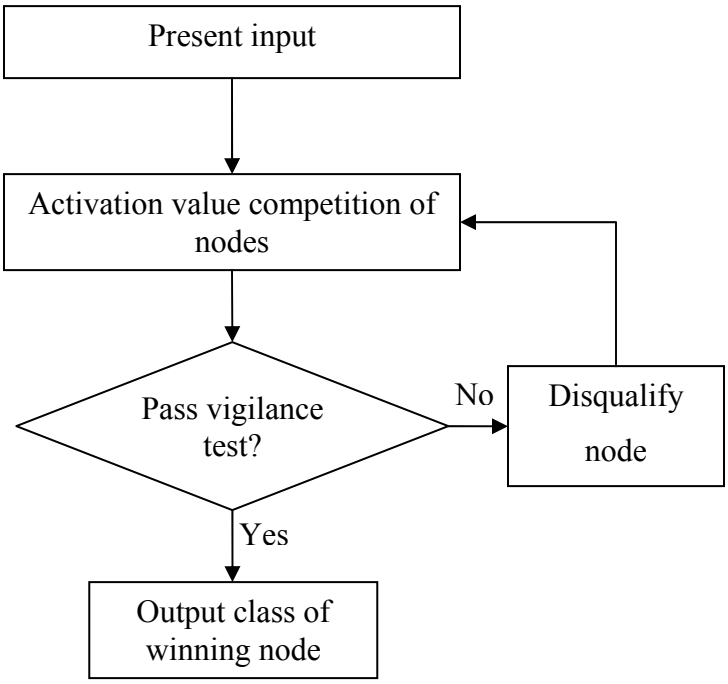


**Figure 4: Flowchart for simplified fuzzy ARTMAP classification process**

For a given input pattern, the activation values of the categories are computed and the one with highest value undergoes the vigilance test as it did in the training process. If the category passes the vigilance test, it will be the winner and its class is predicted as the output class of that input pattern. Otherwise, the test is repeated for the next category with highest activation value until one is found. In the event that none of the categories pass the vigilance test, the input pattern is treated as unclassified.

## 2.2. Problem of Overlapping Classes

The fuzzy ARTMAP module has to deal with data from overlapping classes, but certain problems arise from the use of such kind of data. Data from overlapping classes is difficult to classify in itself since it can belong to either class. In addition, training the network with such data also leads to category proliferation.

### 2.2.1. Category Proliferation

Category proliferation is a well known drawback of fuzzy ARTMAP. It refers to the excessive creation of categories during training which does not necessarily improve the performance of the network [17]. More resources will be required, in terms of storage for the large number of categories, as well as the amount of time needed to carry out training and testing. Moreover, the generalization capability of the network may be adversely affected [18].

Different factors may lead to category proliferation, such as noisy data [7] or simply training with a large data set [18]. However, the problem is most severe when training with data from overlapping classes [19].

When the distribution of two (or more) classes overlap, input patterns lying in the overlapping region cannot be accurately nor reliably classified. Fuzzy ARTMAP training terminates when there are no more changes to the weights of categories in a single epoch, which means the network will try to correctly classify all of the training input data. As a result, in the overlapping region between classes, a large number of granular categories will be created. This will allow all the training input to be correctly classified so there are no changes to the weights, but these categories may not contribute to overall predictive accuracy when other data is presented. With more training epochs, more categories are created to map out the overlapping region, which is illustrated in Figure 5c.

(a)

(b)

(c)

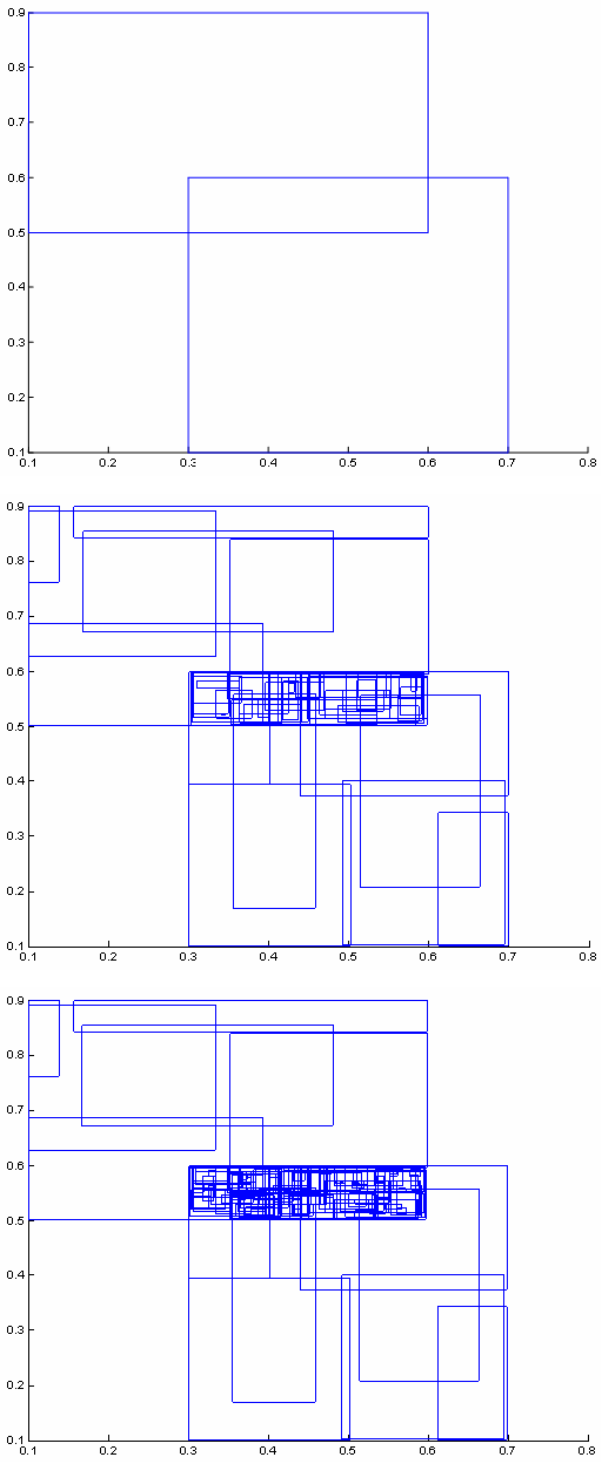


**Figure 5: Class distribution and hyperbox weight distribution after learning with ρ=0.75. (a) Class distribution of the data from 2 classes (b) Position of hyperboxes after 1 training epoch (c) Position of hyperboxes after training until convergence of training data, which required 9 epochs**

Over the training process, new categories are created under certain circumstances, such as when there are input data from classes that have not been encountered yet, or when all the existing nodes fail the vigilance test. However, it is the match tracking process that is the largest contributor to major increases in the number of categories.

As input patterns lying in the overlapping region may belong to either class, many hyperbox categories will be created in that region, and they can also be mapped to either class. Class mismatches are more likely to occur for these data and match tracking will be triggered off more frequently, and the vigilance parameter is raised. For an input pattern presented, many class mismatches may occur due to the large number of categories present in that overlapping region, leading to a magnified temporary increment in the vigilance parameter brought about by match tracking. A higher value of the vigilance parameter translates to increased difficulty for existing categories to pass the test, and hence a new category is more likely to be created to code the input pattern.

Various methods have been proposed to cope with the problem of overlapping classes and they can be widely classified into two types – post-processing methods that operate on the network after training has been completed, or modifications to the learning method to reduce the creation of categories in the first place [13]. The former includes methods such as rule pruning [20] which removes excess categories based on their usage frequency and accuracy. The latter includes modifications in the learning method [21] and weight updating schemes [22], as well as fuzzy ARTMAP variants such as distributed ARTMAP [23], Gaussian ARTMAP [24] and boosted ARTMAP [25].

### *2.2.2. Difficulty of Classification*

Classification of input patterns are done based on the attribute values of the patterns. However, when the class distributions of two classes overlap with each other, it is difficult to classify an input pattern that lies in the overlapping region. Fuzzy ARTMAP only makes one prediction for the input pattern though it can belong to either class. Even

if the activation values for two categories are the same and both pass the vigilance test, only one will be the winner. This choice is usually made by selecting the category with smaller index, or simply selecting one at random.

As a result, it would be unfair to expect the network to assign a class prediction accurately to such an input pattern. Rather than predicting only one class which has a high chance of being the incorrect one, we seek to modify the network such that it can predict more than one class, particularly for input patterns that lie in the overlapping region between classes. Based on the predicted output classes, users can make better decisions. Other information besides the given attributes can be used, or expert knowledge can be combined to determine the class from the predicted list.

Certain variants of fuzzy ARTMAP such as probabilistic fuzzy ARTMAP [15] compute the probability with which a test pattern can belong to each class and predict the output as the class with highest probability. Fuzzy ARTMAP with relevance factor [26] also gives a value of confidence in the classification. The distributed ARTMAP [23] uses distributed learning instead of winner-take-all, and the output class prediction is implemented using a voting strategy. It is possible to make slight modifications to these variants so that they can return more than one output class, but implementing these networks already require major changes to the learning method or architecture and introduce additional computational costs. The dynamics of the system are also no longer as straightforward or intuitive as the original fuzzy ARTMAP. This project thus aims to retain as much of the original architecture as possible and minimize the additional computational effort introduced by the modifications, yet still enable the network to output the possible classes in which the pattern lies, and at the same time reduce the extent of category proliferation.

## 2.3. Methodology

In view of the problems faced by fuzzy ARTMAP due to the use of data from overlapping classes, several modifications were introduced to enable the network to better cope with such kind of data. The classification process of the fuzzy ARTMAP was modified to allow it to predict more than one output class if the data falls in the overlapping region between classes. Consequently, the classification accuracy measure was also modified to adapt to this kind of classification. In addition, other changes were made to help deal with category proliferation.

### 2.3.1. Classification and Accuracy Measure

In fuzzy ARTMAP classification, only the category with highest activation value and which also passes the vigilance test is the winner and will classify the input pattern. In order to predict more than one class, the activation values can be considered such that the hyperboxes with sufficiently high activation values are all considered winners, as long as they also pass the vigilance test. A number of classes can then be predicted based on these winning hyperboxes. A minimum activation value parameter can be introduced, such that any value above this threshold is considered high enough. The key is then to find a fair way to determine when an activation value can be considered to be high enough.

Since multiple class predictions should only occur for data from the overlapping region between classes, there is a need to take a closer look at such data. In the overlapping region, many category hyperboxes are generated during the training process. These hyperboxes are also overlapping with one another and they may be mapped to different classes. As the input patterns which require multiple class predictions lie in this region, they would tend to be contained within more than one category hyperbox.

The activation value of a category hyperbox to a given input pattern determines how close the hyperbox is to that input. It will be higher for hyperboxes that actually contain the input pattern. By considering that input patterns in overlapping regions would lie in hyperboxes which may be mapped to different classes, we can find a threshold activation value, above which we can determine that the hyperbox contains the input pattern.

## Threshold Activation Value

In this section, we find a threshold activation value by considering the activation value function as well as the vigilance test. The activation value is a measure of how close the input pattern is to a particular hyperbox, and will be highest for a hyperbox that contains it. Among those that contain the input pattern, the activation value function is biased towards smallest hyperboxes, which will have higher values. The vigilance test actually imposes a restriction on the maximum size of the hyperbox, so among the hyperboxes that contain an input pattern, the one with largest size will have smallest activation value. Based on this, we can derive the minimum activation value corresponding to the largest hyperbox that contains the input data.

We will first see how the activation value is a measure of how close the input data is to the hyperbox. Suppose the input data is $I = (a, 1-a)$ and the weights of the hyperbox are given as $W = (u, 1-v)$. For input data consisting of M attributes (before complement coding), the hyperbox and input pattern will be in M-dimensional hyperspace, and the vector size $|W|$ can be rewritten as

$$\begin{aligned} |W| &= |(u, 1-v)| \\ &= |(u_1, \ldots, u_M, 1-v_1, \ldots, 1-v_M)| \\ &= u_1 + \ldots + u_M + (1-v_1) + \ldots + (1-v_M) \\ &= u_1 + \ldots + u_M + M - (v_1 + \ldots + v_M) \\ &= |u| + M - |v|, \end{aligned}$$

and the vector size $|I \wedge W|$ can be rewritten as

$$
\begin{aligned}
\left|I \wedge W\right| &= \left|(u, 1-v) \wedge (a, 1-a)\right| \\
&= \left|(u \wedge a, (1-v) \wedge (1-a))\right| \\
&= \left|(u \wedge a, 1-(v \vee a))\right| \\
&= \left|u \wedge a\right| + M - \left|v \vee a\right|.
\end{aligned}
$$

Let $d = \left|W\right| - \left|I \wedge W\right|$, which we want to simplify and show that it is the shortest distance from the input pattern to the hyperbox.

$$
\begin{aligned}
\left|W\right| - \left|I \wedge W\right| &= \left|u\right| + M - \left|v\right| - \left|u \wedge a\right| - M + \left|v \vee a\right| \\
&= \left|u\right| - \left|u \wedge a\right| + \left|v \vee a\right| - \left|v\right|.
\end{aligned}
$$

From Chapter 2.1, we have seen that a hyperbox which expands just enough to code an input pattern will have weights $W_j^{new} = (u \wedge a, 1-(v \vee a))$, where $(u \wedge a)$ and $(v \vee a)$ are the lower and upper end points of the hyperbox. In Figure 6, the illustration is shown in 2-dimensions although the input pattern and hyperboxes are in M-dimensional hyperspace.

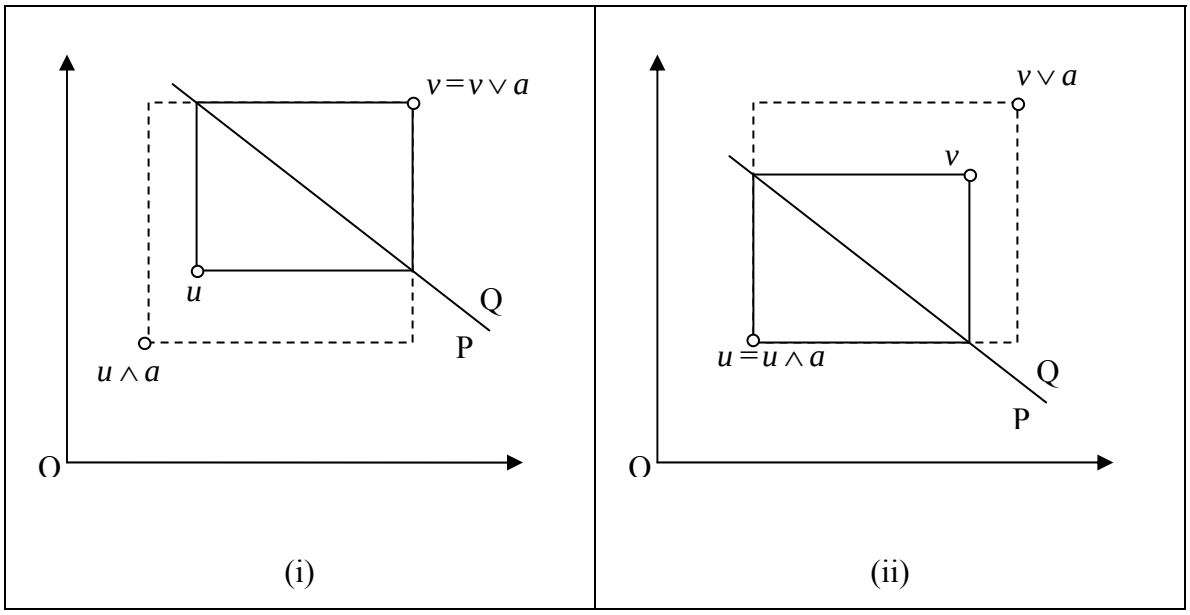


**Figure 6: 2D view of hyperplane separating the hyperbox into two halves. (i) The input pattern lies on side P of the hyperplane (ii) The input pattern lies on the side Q of the hyperplane**

As shown in Figure 6, a hyperplane can be drawn to separate the hyperbox into half, each side containing either the lower or upper end point. If the input pattern lies on the side P in Figure 6, the point $(v \vee a)$ will be the same as $v$ itself. Then

$$|v \vee a| - |v| = 0.$$

We now consider $|u| - |u \wedge a|$.

$$\begin{aligned} |u| - |u \wedge a| &= |(u_1, \ldots, u_M)| - |(\min(u_1, a_1), \ldots, \min(u_1, a_M))| \\ &= u_1 - \min(u_1, a_1) + \ldots + u_M - \min(u_M, a_M), \end{aligned}$$

which is the shortest distance in each dimension between the points $u$ and $u \wedge a$. This distance measure is called the Manhattan or block distance between $u$ and $u \wedge a$. In 2-dimension, it can be illustrated more clearly in the Figures 7 and 8.

In Figure 7, the position of the input pattern $a$ is such that $u \wedge a$ will coincide with it. The distance $d = d_1 + d_2$ measures the block distance from $a$ to $u$, which is the nearest point on the hyperbox to $a$.

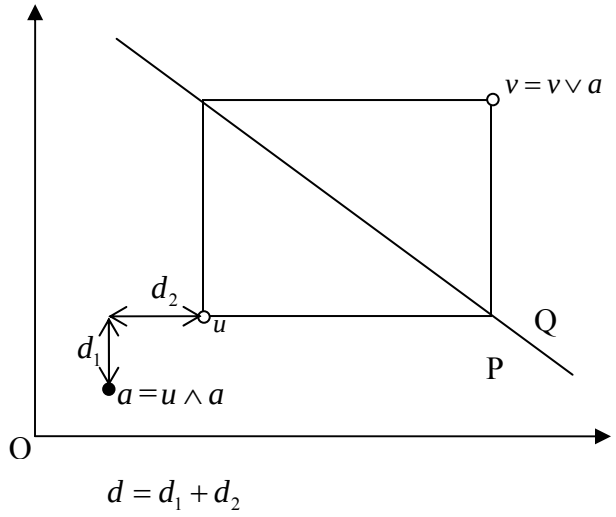


**Figure 7: Shortest distance from the input pattern to the hyperbox when position of input pattern a on side P coincides with u∧a**

In Figure 8, the point $u \wedge a$ does not coincide with the input pattern $a$. The distance $d$ between $u$ and $u \wedge a$ is the same as the shortest block distance between the input pattern $a$ and the hyperbox.

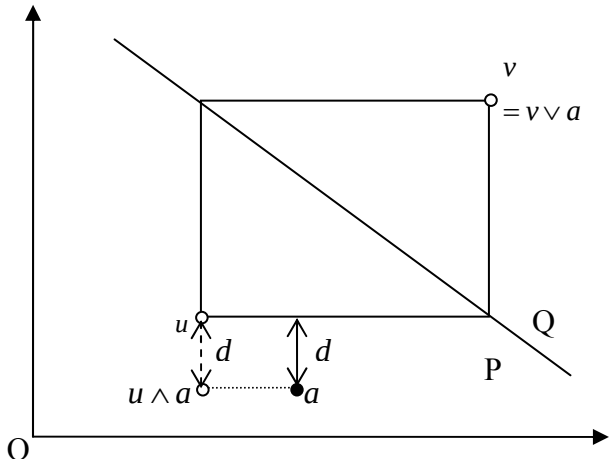


**Figure 8: Shortest distance from the input pattern to the hyperbox when the position of input pattern a on side P does not coincide with u∧a**

All other cases are similar to either of these cases. Given that $d = |W| - |I \wedge W|$, we can rewrite the activation value as

$$\begin{aligned} T &= \frac{|I \wedge W|}{\alpha + |W|} \\ &= \frac{|W| - d}{\alpha + |W|} \\ &= \frac{|W| + \alpha - \alpha - d}{\alpha + |W|} \\ &= 1 - \frac{\alpha + d}{\alpha + |W|}. \end{aligned}$$

The nearer the input pattern $a$ is to the hyperbox, the larger the activation value will be. Since $\alpha, d, |W| \geq 0$, the activation value is largest when $d = 0$, which is when the hyperbox contains the input pattern.

Among the hyperboxes that contain an input pattern $a$, a smaller hyperbox will have higher activation value. This can be shown by considering two hyperboxes $H_A$ and $H_B$ with respective weights $W_A = (u_A, 1-v_A)$ and $W_B = (u_B, 1-v_B)$. Hyperbox $H_A$ has maximum size and they can be positioned in 2-d in Figure 9 without loss of generality.

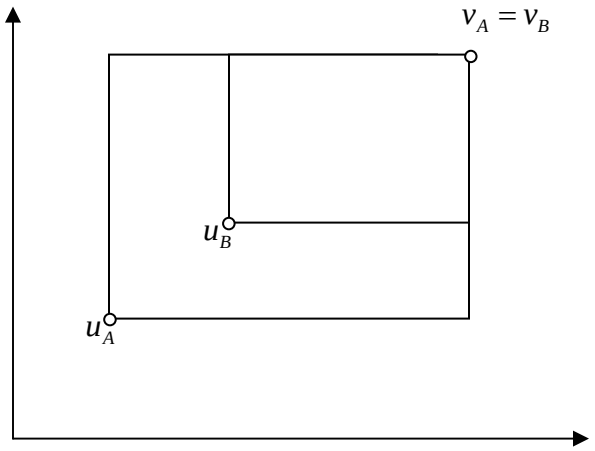


**Figure 9: Hyperboxes of different sizes**

The end points $v_A$ and $v_B$ coincide but $u_A$ is closer to the origin than $u_B$ since $H_A$ is larger. This will give $|u_A| < |u_B|$, so we have

$$\begin{aligned} |W_B| - |W_A| &= |u_B| + M - |v_B| - |u_A| - M + |v_A| \\ &= |u_B| - |u_A| \\ &> 0. \end{aligned}$$

Therefore, a hyperbox with larger size will have smaller weight vectors.

Let the activation values for $H_A$ and $H_B$ containing the input pattern be

$$T_A = \frac{|W_A|}{\alpha + |W_A|}$$

and

$$T_B = \frac{|W_B|}{\alpha + |W_B|}.$$

Then

$$\begin{aligned} T_B - T_A &= \frac{|W_B|}{\alpha + |W_B|} - \frac{|W_A|}{\alpha + |W_A|} \\ &= \frac{|W_B|(\alpha + |W_A|) - |W_A|(\alpha + |W_B|)}{(\alpha + |W_B|)(\alpha + |W_A|)} \\ &= \frac{\alpha(|W_B| - |W_A|)}{(\alpha + |W_B|)(\alpha + |W_A|)} \\ &> 0, \end{aligned}$$

since

$$|W_B| - |W_A| > 0.$$

Therefore, the activation value for a smaller hyperbox is higher than for a larger hyperbox if they both contain the input pattern. In turn, the activation value of the largest hyperbox which contains the hyperbox is still larger than one which does not.

It now remains to find the size of the weights $|W_{max}|$ for the largest hyperbox, in order to obtain a threshold for the activation value. This size can be found by considering the vigilance test. A hyperbox with weights $|W|$ can code an input pattern only if it satisfies the vigilance test, after which the new weights will be updated according to

$$W^{new} \leftarrow W \wedge I.$$

The updated hyperbox will contain the input pattern and still satisfy the vigilance test, so we have

$$\begin{aligned} \frac{|I \wedge W^{new}|}{|I|} &= \frac{|W^{new}|}{|I|} \\ &\geq \rho. \end{aligned}$$

$|W^{new}|$ is therefore bounded below by $\rho|I|$.

The size of an input pattern is given by

$$\begin{aligned} |I| &= \left|\left(a_1,\ldots,a_M,1-a_1,\ldots,1-a_M\right)\right| \\ &= M, \end{aligned}$$

where M is the number of attributes of the data. This gives the bound

$$|W_{max}| \geq \rho M$$

for the largest hyperbox size. This gives us the minimum activation value for a hyperbox containing an input pattern as

$$\frac{M\rho}{\alpha + M\rho}.$$

Any hyperbox with activation value above this threshold will be containing the input pattern.

With this threshold, category hyperboxes whose activation value exceeds it are allowed to classify the input pattern, and their classes will be among those predicted for the input pattern.

Using the data from the class distribution as shown in Figure 5(a), the network was trained and testing was carried out. Most of the test data had only one class predicted, but a portion of them had two classes predicted and they are depicted in Figure 10. Those data points lie only in the overlapping region between the two classes, and not in other regions where distinct class prediction is possible.

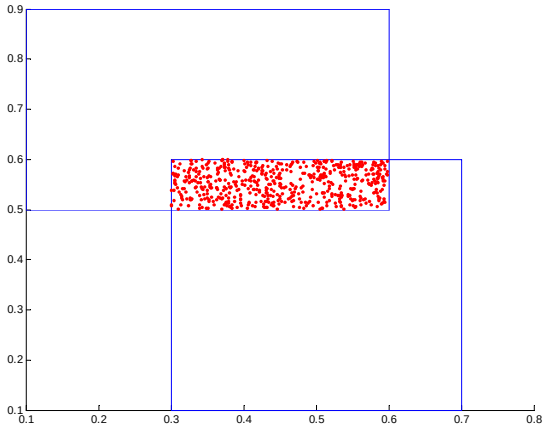


**Figure 10: Patterns with more than one predicted class**

## Change in Accuracy Measure

With the change in classification, input patterns in the overlapping region have more than one predicted class. The measure of predictive accuracy will need to be changed to adapt to this new method of classification. An input pattern is deemed as correctly classified if the actual class is one of those predicted by the network. This is because the objective is to identify the possible class predictions for data in overlapping region, and then use other methods to further distinguish them. For the simple 2d data with class distribution as shown in Figure 5(a), the network was trained until completion, which took 9 epochs and the category hyperboxes were positioned as in Figure 5(c). The original method of classification and measure gave 94.33% accuracy on the testing data generated from the class distribution, whereas the modified classification with multiple class prediction and the corresponding measure gave 100% accuracy, since the category hyperboxes have already covered the full class distribution.

### *2.3.2. Measures to Reduce Category Proliferation*

In order to reduce the category proliferation problem caused by overlapping classes, several measures were taken and investigated in this project. By using a single epoch for training, match tracking - and training the input data class by class, the number of categories created during the learning process can be limited to prevent excessive creation. In addition, some hyperboxes may be merged after the training process to further reduce the number of categories.

## Single epoch training

Various training strategies can be used in the learning of input patterns. In [18], strategies include training until convergence on training data set, single epoch training, cross

validation and training until convergence of hyperbox weight values. These criteria determine when to terminate the learning process.

As seen in Figure 5(b), a large number of categories are already created after one training epoch for data from overlapping classes. Training until completion requires even more epochs, during which the network tries to correctly classify all the training data. However, Figure 5(c) shows that for the given example, the change beyond the first epoch arises mainly in the form of the creation of more categories in the overlapping region. These categories will give a classification accuracy of 100% on the training data used since the network terminates the process only when the training data are all correctly classified, but the additional categories will not contribute much to the actual predictive accuracy of test patterns. This is especially so after a change in the accuracy measure. For the 2-d data used in Figure 5, the modified accuracy on the training and testing data was 100% for both Figure 5(b) and 5(c).

Given the weight updating rule, the network already produces a number of categories that can classify a majority of the training input patterns in the first epoch. Further epochs can better map the class boundaries or handle populated exceptions. But as the main concern here is overlapping classes rather than a complex decision boundary, further epochs are less desirable due to category proliferation as granular categories are created in the overlapping region of classes. To reduce this problem, only one training epoch will be used. A later modification will further demonstrate the benefit of single epoch training.

## Match Tracking - (MT-)

A class mismatch during the training process triggers off the match tracking process, which temporarily raises the vigilance parameter while searching for an alternative winning hyperbox that is mapped to the same class as the predicted input pattern. This makes the vigilance test harder to pass and a new category is more likely to be created. Since data from overlapping classes is equivalent to inconsistent cases mentioned in [27],

the variant match tracking - suggested in [27] is employed here. Fewer categories are likely to result from class mismatches. Instead of using a small positive value for $\varepsilon$, it is set to a negative value, hence the method is termed MT- (minus). By changing the value for $\varepsilon$, the temporary vigilance test is not too difficult to pass, which allows more chance to the existing categories before creating a new one.

## Ordered Presentation of Training Input

Besides MT-, another measure is taken to reduce the creation of new categories resulting from class mismatches. Instead of shuffling the training data and presenting them to the network in random order for learning, the data is first sorted by class and presented one class at a time. The order of training input pattern presentation influences the number of categories and generalization capability [28]. There are no restrictions on the order of input presentation within each class, nor on which class to present first. The key is to present data from the same class in bulk. The number of class mismatches leading to match tracking can be reduced and thus prevent the excessive creation of categories. This can be explained by considering the training process of data from two classes which overlap with each other.

During the presentation of input patterns from the first class, only categories mapped to that class are present, so naturally there are no class mismatches. Hence by the end of the input presentation for that class, the category hyperboxes have been allowed to grow in size. When the input patterns from the second class are presented, the overlapping region between the two classes may still see the creation of new categories, but the number will be reduced. This is because the existing hyperboxes mapped to the first class have already grown to a larger size, so the activation value for the newly created smaller hyperboxes of the second class will have comparatively higher activation values. With a better chance of winning the competition, the number of class mismatches can be reduced. In addition, even when class mismatches occur and trigger off match tracking, the larger

size of the hyperboxes of the first class will result in a smaller value of $|I_r \wedge W_j|$ and subsequently a temporary new vigilance value given by

$$\frac{|I_r \wedge W_j|}{|I_r|} + \varepsilon,$$

which is lower than the one resulting from a mismatch occurring from a hyperbox with large size. This vigilance test is less difficult to pass, which will then reduce the likelihood of the creation of a new category. This result can be better illustrated in Table 1.

**Table 1: Results from class by class single-epoch training of 2-D data from Figure 5**

| Train set 4000 | Vigilance = 0.5 | | Vigilance = 0.75 | |
|---|---|---|---|---|
| | Rand-train | Sort-train | Rand-train | Sort-train |
| Total number of categories | 30 | 2 | 52 | 17 |
| Number of categories created from match tracking | 28 | 0 | 43 | 4 |
| Average raised vigilance | 0.8776 | 0.5505 | 0.9059 | 0.7673 |
| Number of times match tracking is triggered | 327 | 5 | 369 | 34 |

In Table 1, rand-train denotes training the network with a randomly shuffled order of training data and sort-train denotes training class by class. The layout of hyperboxes shown in Figure 5 was obtained using vigilance parameter of value 0.75. As the large number of hyperboxes could arise from a high vigilance parameter value, Table 1 includes the results from using a lower vigilance parameter value of 0.5 as well. In both cases, match tracking (MT) was triggered off more frequently for rand-train than for sort-train. At $\rho = 0.5$, match tracking was triggered off 327 times for rand-train as compared

to only 5 times for sort-train. At $\rho = 0.75$, it was triggered off 369 times as compared to only 34 times for sort-train. This frequent match tracking gave rise to majority of the categories that were created. For $\rho = 0.5$, 28 of the 30 categories created using rand-train was a result of match tracking; and for $\rho = 0.75$, 43 out of the 52 categories created arose from match tracking. In addition, the average value of the increased vigilance parameter during match tracking was also higher for rand-train as compared to sort-train: 0.88 as compared to 0.55 using $\rho = 0.5$ and 0.91 as compared to 0.77 using $\rho = 0.75$. It should be noted, however, that if additional training epochs were to be used, a large number of categories could still be created for the sort-train method as the network attempts to classify all training input patterns.

## Merging Categories

Although MT- and training class by class can reduce match-tracking occurrences and category proliferation to a certain extent, the overlapping classes will still lead to the creation of some additional granular categories within the overlapping region. To deal with them, post-processing methods can be employed to reduce the number of categories even further after training. Pruning is a popular strategy which can reduce the number of categories based on factors such as usage frequency or predictive accuracy [20]. But due to the modified accuracy, as well as the use of categories to predict more than one class, pruning is no longer an effective strategy for use here. This is because the small categories in the overlapping region may not be frequently used as they may not win the competition, and the predictive accuracy may be low since the patterns in that region may belong to either class. According to the pruning criteria, these categories would be removed, yet they are necessary for the network to predict more classes for certain input patterns. Rather than pruning and removing them, these categories can be merged instead.

Merging can be carried out based on different conditions, and the following are selected. Two category hyperboxes are merged together if: (a) they are mapped to the same class, (b) the centroids of the two hyperboxes are near to each other, satisfying a certain

distance threshold, and (c) the resultant hyperbox after merging, which contains both original hyperboxes, does not exceed the maximum size imposed by the selected vigilance parameter. The centroid of a category hyperbox is taken as the midpoint of each dimension of the hyperbox, and the distance between the centroids is computed using the Manhattan distance, which is the measure used to compute the maximum hyperbox size [29]. If a hyperbox is larger than this size, it will certainly fail the vigilance test and cannot classify an input pattern. In this situation, even those that were initially classified by the original hyperboxes are no longer classified by the resultant hyperbox after merging.

## 2.4. Results

The modifications suggested are tested out on various data. Initial testing is done on two datasets from the UCI Machine Learning Repository, to verify the efficacy of the methods in handling the category proliferation problem. Further validation is carried out on a massive simulated data set with a large number of classes as well as training and testing patterns.

All the datasets used have some degree of overlap in terms of the range of values which their attributes can take. The attributes are all a mixture of discrete and numerical values. Each dataset is split into 70% for training and 30% for testing. Data from overlapping classes may be difficult to classify accurately. Table 2 shows the results of classification of the UCI data using fuzzy ARTMAP (FAM) without the modifications, as well as multilayer perceptron (MLP) and learning vector quantization (LVQ). Details of the UCI datasets – yeast and contraceptive method choice data – will be given later in this section. The results shown here are those obtained by using the parameter values of each method after scanning to find the most suitable values that gave the best performance.

**Table 2: Accuracy of UCI data using different classification methods**

| | FAM | MLP | LVQ |
|---|---|---|---|
| Yeast Data | 45.39% | 44.04% | 45.17% |
| Contraceptive Method Choice Data | 45.93% | 43.67% | 49.32% |

All three methods could not achieve accuracies above 50% despite various combinations of the parameters used. It is difficult to classify the patterns because the class distributions are overlapping and they can belong to more than one class, and this is why the classification is modified to return more than one output class where relevant, and the accuracy measure changed accordingly. The results are shown in the following sections, depicting the average number of categories formed and the accuracy for different modifications used to reduce category proliferation.

**Table 3: Combinations of modifications**

| | MT- | Train class by class | Merge categories | Single epoch training |
|---|---|---|---|---|
| FAM | | | | ✓ |
| FAM-MT- | ✓ | | | ✓ |
| FAM-new | | ✓ | ✓ | ✓ |
| FAM-combined | ✓ | ✓ | ✓ | ✓ |

Having seen how the number of categories grows with the number of training epochs used, only one training epoch is used even if there are no other modifications made to the training process, in order to facilitate comparisons on the number of categories. The combinations of modifications used and their corresponding names are shown in Table 3. FAM is the original fuzzy ARTMAP classification using single epoch training. FAM-MT- uses matching tracking – with negative $\varepsilon$ instead of the usual match tracking with positive $\varepsilon$. FAM-new employs only the new modifications suggested (training class by class and merging of categories). FAM-combined uses all the modifications mentioned.

The results to these different combinations of the modifications are shown to compare the improvements made by the different measures.

For ordered input presentation, the classes were simply presented in the order of their class label indices. All the accuracies shown below are for the modified classification and accuracy measure, and results are averaged over 100 simulations. The vigilance value was chosen as $\rho = 0.5$ for all the data since it gave the best results after testing with values from 0.1 to 0.9 with step size of 0.1. The match tracking parameter was set as $\varepsilon = 0.000001$ for normal match tracking and $\varepsilon = -0.01$ for MT-. The choice parameter was set to $\alpha = 0.000001$.

### *2.4.1. Results for UCI Datasets*

## Results for Yeast Database

The yeast database from Machine Learning Repository was donated by Paul Horton [30]. Proteins from yeast were classified into 10 classes based on their cellular localization sites, such as cytoskeletal, nuclear and mitochondrial, vacuolar, peroxisimal, extracellular, localized to lumen of endoplasmic reticulum, membrane proteins with cleaved signal, uncleaved signal, or no N-terminal signal. The 8 attributes are calculated from the amino acid sequences and include scores from McGeoch's method and von Heijne's method for signal sequence recognition, score of ALOM membrane spanning region prediction program, results of discriminant analysis of amino acid content of 20-residue N-terminal region of mitochondrial and non-mitochondrial proteins, discriminant analysis of the amino acid content of vacuolar and extracellular proteins, discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins, peroxisomal targeting signal in the C-terminus, and the presence or absence of an HDEL substring. The final attribute is binary. All the attribute values already lie in the range 0 to 1. A total of 1484 samples were available, with no missing attribute values in the dataset.

**Table 4: Comparisons between modifications on results for yeast data**

| | Number of categories | Accuracy |
|---|---|---|
| FAM | 175 | 81.09% |
| FAM-MT- | 90 | 82.91% |
| FAM-new | 23 | 84.51% |
| FAM-combined | 20 | 83.07% |

By targeting the creation of categories from the match-tracking process, the number of categories could be reduced. Simply employing MT- could reduce the number of categories by almost 50%, but the other modifications suggested here gave a more significant reduction of up to 86%. The accuracy also improved from 81.09% to 84.51%, indicating that the reduction in categories was primarily in the overlapping region since the reduced categories did not adversely affect accuracy.

## Results for Contraceptive Method Choice Data

The contraceptive method choice data [31] on Machine Learning Repository is taken from the 1987 National Indonesia Contraceptive Prevalence Survey. Women were interviewed on their choice of contraceptive methods and classified into 3 classes – short term use, long term use, or no use of contraceptive methods. The 9 attributes reflected demographic and socio-economic characteristics such as age, religion, employment, media exposure, standard of living, number of children, husband's occupation, and both the wife and husband's education. The samples belong to either of the 3 classes, but the attributes used to describe the samples are not entirely sufficient to separate them into the classes. Two women sharing very similar characteristics could choose different contraceptive methods, so the class distributions are overlapping. Most of the attributes are discrete, but they are represented by fixed values between 0 and 1. The size of this dataset is 1473 and there are no missing attribute values.

**Table 5: Comparisons between modifications on results for contraceptive method choice data**

| | Number of categories | Accuracy |
|---|---|---|
| FAM | 233 | 78.88% |
| FAM-MT- | 169 | 82.94% |
| FAM-new | 57 | 89.90% |
| FAM-combined | 56 | 87.10% |

MT- reduced the number of categories by about 27%, but the suggested modifications could reduce it by about 75%, and markedly improving the accuracy from 78.88% to 89.90% at the same time. A combination of MT- and the suggested modifications yielded the fewest categories like for the yeast dataset, but the accuracy was slightly affected.

In the above two datasets from the UCI Machine Learning Repository, the combined modifications reduced the number of categories by at least half while the accuracy improved. We shall now test out the modifications on larger simulated data.

### *2.4.2. Results for Synthetic Data*

## Results for Synthetic data without noise

This data is simulated and has a size of 49500. There are 165 classes and the data has 7 attributes before complement coding. Three of the attributes are numerical and the other four are discrete.

**Table 6: Comparisons between modifications on results for synthetic data without noise**

| | Number of categories | Accuracy |
|---|---|---|
| FAM | 2046 | 95.19% |
| FAM-MT- | 177 | 99.78% |
| FAM-new | 416 | 99.76% |
| FAM-combined | 165 | 99.82% |

Due to the higher number of classes and the higher degree of overlap between the classes, a single epoch of training led to the creation of up to 2046 categories. Without any modifications, one epoch of training yielded 2046 categories. By changing $\varepsilon$ from 0.000001 to -0.01 in MT-, the vigilance parameter was controlled during the match tracking process, and the number of categories created dropped by more than 90%. This could imply there had been many occurrences of class mismatches during training, so by controlling the vigilance parameter and preventing it from growing too much, the creation of new categories was inhibited. Unlike the two UCI datasets, MT- worked better than the suggested modifications for this dataset. A reason for this may be the large number of classes and training data. There are overlaps between many classes and the large number of patterns in the overlapping region will trigger match tracking many times. Despite presenting the patterns in order during training, class mismatches still occur quite frequently, leading to match tracking and creation of categories. Nevertheless, the best performance was achieved by employing all the modifications. The number of categories

dropped further to 165, which was the minimum number required given the number of classes the data was distributed into. Predictive accuracy improved further to 99.82%.

## Results for Synthetic data with noise

This dataset is similar to the previous one, consisting of 49500 data with 7 attributes each. However, this dataset is introduced with Gaussian noise using 20% standard deviation.

**Table 7: Comparisons between modifications on results for synthetic data with noise**

| | Number of categories | Accuracy |
|---|---|---|
| FAM | 6115 | 84.50% |
| FAM-MT- | 2482 | 92.19% |
| FAM-new | 2165 | 93.26% |
| FAM-combined | 1325 | 94.62% |

With the introduction of noise to the data, the predictive accuracy was badly affected and the number of categories created grew drastically. MT- reduced the number of categories by about 60%, while the suggested modifications reduced it by about 65%. Unlike for the previous dataset, the reduction of categories was more significant using the suggested modifications than MT-. This could be due to the presence of noise. Nevertheless, it was still a combination of all the modifications that gave the best performance, reducing the number of categories by more than 78% and improving the predictive accuracy significantly from 85% to 95%, an indication that the modifications work well with noisy data.

## 2.5. Discussion

**Table 8: Summary of results using fuzzy ARTMAP with and without modifications**

| | No modifications | With modifications |
|---|---|---|
| Yeast | | |
| Number of categories | 175 | 20 |
| Accuracy | 81.09% | 83.07% |
| Contraceptive Method Choice Data | | |
| Number of categories | 233 | 56 |
| Accuracy | 78.88% | 87.10% |
| Synthetic Data without Noise | | |
| Number of categories | 2046 | 165 |
| Accuracy | 95.19% | 99.82% |
| Synthetic Data with Noise | | |
| Number of categories | 6115 | 1325 |
| Accuracy | 84.50% | 94.62% |

Table 8 summarizes the results from testing on the various data. The combined use of all the modifications can reduce the number of categories created and at the same time improve the accuracy. However, there are certain concerns about the modifications introduced that are investigated further.

Although additional epochs of training lead to category proliferation in overlapping data, they are needed to approximate the boundaries separating the class distributions where they are not overlapping. By reducing the number of training epochs to only one, patterns lying near the boundary between classes may not be properly classified. Therefore, the method is better suited to data whose classes have a wider margin of separation where they are not overlapping. This could translate to suitability to data with more attributes

since non-overlapping class distributions are less likely to be leaning on the same boundary when in the higher dimensional hyperspace.

By training class by class, the number of categories created due to overlapping classes can be reduced. However, incremental learning is sometimes required as new data becomes available, and it is inefficient to retrain the network class by class from scratch using all the training data. Rather than doing so, the network can learn the new batch of data incrementally as long as it is also sorted by class. Although the results would not be as good as training class by class from scratch, it will still be better than training all the data in a random order. Table 9 compares the results of training a set of data in random order against the results of training in 2 and 3 batches incrementally. Rand-train denotes the training the whole set of data in random order, which is consistent with training incrementally as new data becomes available. Sort-train Case 1 trains the network in 2 batches – the first 70% followed by the remaining 30%. Sort-train Case 2 trains the network in 3 batches – the first 70% followed by 15% and then the last 15%. Training was carried out for one epoch, without MT- or merging. The parameters used are same as before and the results are averaged over 100 simulations.

**Table 9: Results for ordered incremental learning using UCI data**

| | No. of categories | Accuracy |
|---|---|---|
| Yeast | | |
| Rand-train | 175 | 81.10% |
| Sort-train Case 1 | 65 | 88.22% |
| Sort-train Case 2 | 74 | 88.11% |
| Contraceptive Method Choice | | |
| Rand-train | 236 | 78.64% |
| Sort-train Case 1 | 83 | 91.05% |
| Sort-train Case 2 | 96 | 90.81% |

As with the number of epochs, the effectiveness of training class by class degrades with the number of times incremental learning is used. However, the results are still better than training the input in a random order, as long as the bulk of the training data has been trained in order from the beginning.

The merging of categories allows the input patterns of the hyperboxes to still be classified by the corresponding resultant hyperbox after merging. With multiple class prediction and the modified accuracy measure, merging will not lead to misclassification. This is because the patterns initially classified by the hyperboxes before they undergo merging will still be classified by the resultant hyperbox. Input patterns that were originally classified by other hyperboxes will still be so, even if it is now contained in the resultant hyperbox, since both classes will be predicted. However, there may be an increase in the number of predictions for the patterns, and such patterns may not even be in the overlapping region. Therefore, additional class predictions need to be justified with an increase of reasonable proportion in the predictive accuracy.

**Table 10: Improvements in performance from using merging**

| | Percentage improvement in | |
|---|---|---|
| | No. of categories | Accuracy |
| Yeast | 6.52% | 0.13% |
| Contraceptive Method Choice | 0.23% | 0% |
| Synthetic data without noise | 0.07% | 0% |
| Synthetic data with noise | 15.37% | 0.55% |

Table 10 shows the results of merging after the other modifications have been carried out, averaged over 100 simulations. For the UCI data and the synthetic data without noise, the other modifications are already sufficient to reduce the number of categories, and

merging makes little improvement to the results. There are slightly better improvements for the synthetic data with noise, seeing a drop of about 15.37% in the number of categories. However, the increase in accuracy is only 0.55%, and further investigation is carried out on the increase in the number of class predictions for the test data.

14850 patterns were used for testing of the synthetic data with noise. On average, merging increased the total number of class predictions for all patterns by 382, but the number of correctly classified patterns increased by only 82. The large increase in the number of predicted classes could be due to the large size of the resultant hyperbox after merging, covering more space than that belonging to the class. To rectify this, a reduction in the distance threshold may be used, or a higher vigilance value to impose a stricter hyperbox size restriction can be employed during the merging process, in order to limit the size of the resultant merged hyperbox. However, the poor performance could also be a reflection of the unsuitability of merging for the data used due to the geometry of the class distributions.

# Chapter 3. Signal Sorting by TOA

The other module in the classification system which is explored is the signal sorting module. The data sample fed into this module consists of signals from various sources. Each signal has its own time-of-arrival (TOA) along with other attributes. Signals from the same source will form a sequence train with approximately the same time interval between the TOAs, and this interval is known as the pulse repetitive interval (PRI). Sequence trains from different sources will have different PRI. The purpose of the signal sorting module is to separate the signals into their various sources, which is equivalent to sorting them into their respective sequence trains.

The attributes of the signals alone are not sufficient to separate the signals into their various sources. The TOA will have to be used to sort them into their sequences with the same PRI, and this is precisely the signal sorting process. Existing methods such as sequence search and difference histogram methods can do this, but are also not entirely sufficient to do so and will face certain difficulties. One way of dealing with this is to first cluster the signals based on their attributes, and then carry out signal sorting on the signals from each cluster. Such methods have been introduced in [32], [33] and [34].

However, when expert knowledge is available, it could play a more active role in the signal sorting process. As data gets accumulated over time, this knowledge can be incorporated into the system to improve the performance [35]. In the case of our system of interest, prior knowledge of the attributes and the sources are available in the form of a database. Our goal is to make use of this knowledge in the signal sorting process to overcome the limitations and difficulties faced in the existing methods.

## 3.1. Existing Method

### 3.1.1. Sequence Search

The sample consists of signals from various sources but sorted according to their TOA. Sequence search works to identify the PRI of the sources present by attempting to construct sequence trains based on the possible values. More details of the algorithm can be found in [36].

From a starting signal, the interval to its adjacent signal is found and a trial train is constructed based on this interval. If there are enough matches between the trial train and the sample, this trial train will be extracted and its interval is the identified PRI. Sequence search is then carried out on the remaining signals in the sample and the process is repeated. If there are insufficient matches, the interval is discarded and trial train construction is attempted using the interval between the starting signal and the subsequent signal instead. This process can be more clearly illustrated using algorithm 3.1.

The sequence search method is reliable and accurate, and straightforward to implement. However, it is very processor intensive [37]. Rather than constructing trial trains for all possible PRIs, it would be more efficient to find a smaller subset of possible PRIs and construct trial trains only for those values. Such a set of possible PRIs can be found using the difference histogram method.

## Algorithm 3.1

Sample consists of signals with TOAs $\{s_1, s_2, ..., s_n\}$

**for** $i \leftarrow 1, 2, ..., n$

    Starting signal $\leftarrow s_i$

    **for** $j \leftarrow i+1, ..., n$

        interval $\leftarrow s_j - s_i$

        Construct trial train using this interval

        **if** number of matches are sufficient,

            Extract train and restart algorithm

        **end** (**if**)

    **end** (**for** j)

**end** (**for** i)

### *3.1.2. Difference Histogram Method*

A successful sequence search requires sufficient matches between the constructed train using a particular PRI and the actual sample, so the given PRI should frequently appear as an interval between signals in the sample. To find such PRIs, a histogram of the intervals between TOA of the signals in the sample can be formed. If a certain histogram bin is tall enough, its corresponding PRI value is likely to be a PRI. A threshold can be used to determine whether the bin is sufficiently tall, before the PRI value is used for sequence search. This threshold cannot simply be a fixed value, since smaller intervals are bound to appear more often in a sample containing signals from several sources, but they generally do not correspond to actual PRI values. Therefore, the threshold should be a function that takes this into consideration. The success of the histogram method will depend mainly on the way the histogram is formed and the threshold that is used. We will first look at the two different histogram methods, namely the cumulative difference

(CDIF) histogram method and the sequential difference (SDIF) histogram method, followed by the threshold function that is actually used.

## CDIF Histogram

At the first difference level, the CDIF histogram method forms a histogram (see Figure 11(i) for an illustrated example) of the TOA first differences $d_1$, which are the intervals between each signal and its adjacent signal. The count at each interval and its double interval is compared to a threshold. If the count exceeds the threshold, that interval will be a possible PRI and sequence search is carried out by constructing a trial train based on that PRI.

Figure 11 shows the histogram bins compared against the threshold up to the fourth difference level, when an interval was successfully identified for sequence search. The sample used consists of 3 trains with PRI 5, 8 and 11. Four difference levels had to be computed before an interval and its double interval both exceeded the threshold. In Figure 11(iv), the bin at PRI = 5 and at PRI = 10 both exceeded the threshold, so sequence search can be carried out for interval 5.

## Algorithm 3.2

Sample consists of signals with TOAs = $\{s_1, s_2, \ldots, s_n\}$

Set number of bins $N_{bins}$

**for** $c \leftarrow 1, 2, ...$

Find $c^{\text{th}}$ differences $d_c$, where $(d_c)_i = s_{i+c} - s_i$, for $i = 1, ..., n-c$

Form histogram for $d_c$

Accumulate histogram count from $d_i$, for $i = 1, ..., c-1$

Draw threshold $t_j$ and compare with bin heights $h_j$ for $j = 1, ..., N_{bins}$

Let $L = \{l \mid h_l \geq t_l\}$

**for** $j \in L$

**if** $2j \in L$

Do sequence search for histogram bin centre $p_j$

**if** the train is successfully extracted,

Reset $c \leftarrow 1$

Repeat CDIF algorithm on the remaining samples

**end**

**end**

**end** (**for** j)

**end** (**for** *c*)

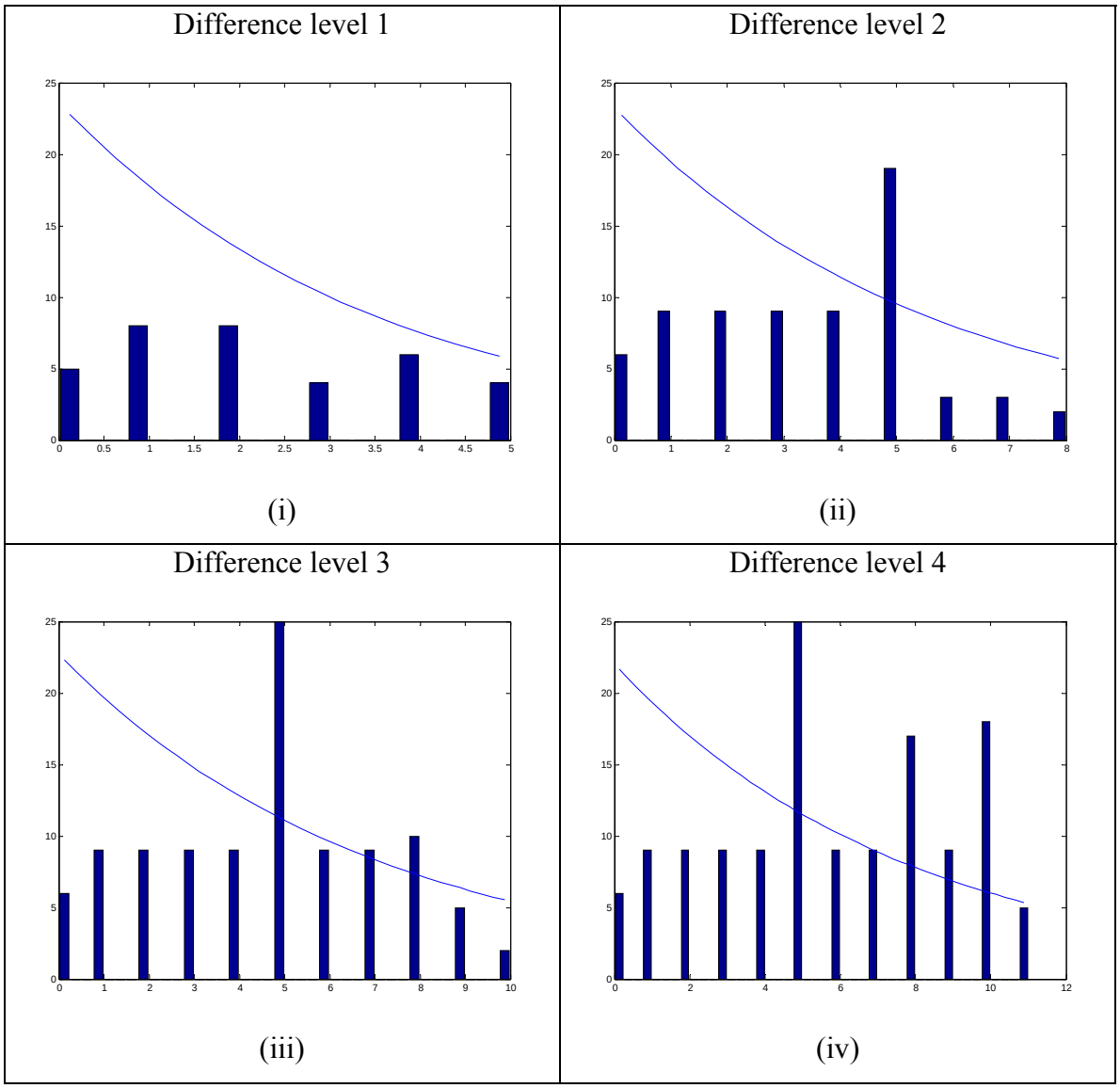


**Figure 11: CDIF histograms up to difference level 4**

## SDIF Histogram

The SDIF histogram method forms histograms like in CDIF (see Figure 12(i)) but without accumulating the count from previous difference levels. To determine the possible PRI based on which sequence search should be carried out, the histogram is not only compared to the threshold function, but an additional subharmonic check is required if the tallest histogram bin does not cut the threshold. Let $p_m$ denote the interval

represented by the histogram bin with maximum height, and $p_1$ be the interval of the first histogram bin which exceeds the threshold. If $p_1$ represents a multiple of $p_m$, then $p_m$ should be used for sequence search. Otherwise, sequence search will be carried out for all intervals whose count exceeds the threshold. Subsequent difference level histograms are formed like in CDIF (see Figure 12(ii)) but without count accumulation. The sample used here is the same as that used in Figure 11. At the second difference level, the count for interval 5 is the maximum and also exceeds the threshold, hence it passes the subharmonic check. Algorithm 3.3 presents the steps for SDIF method.

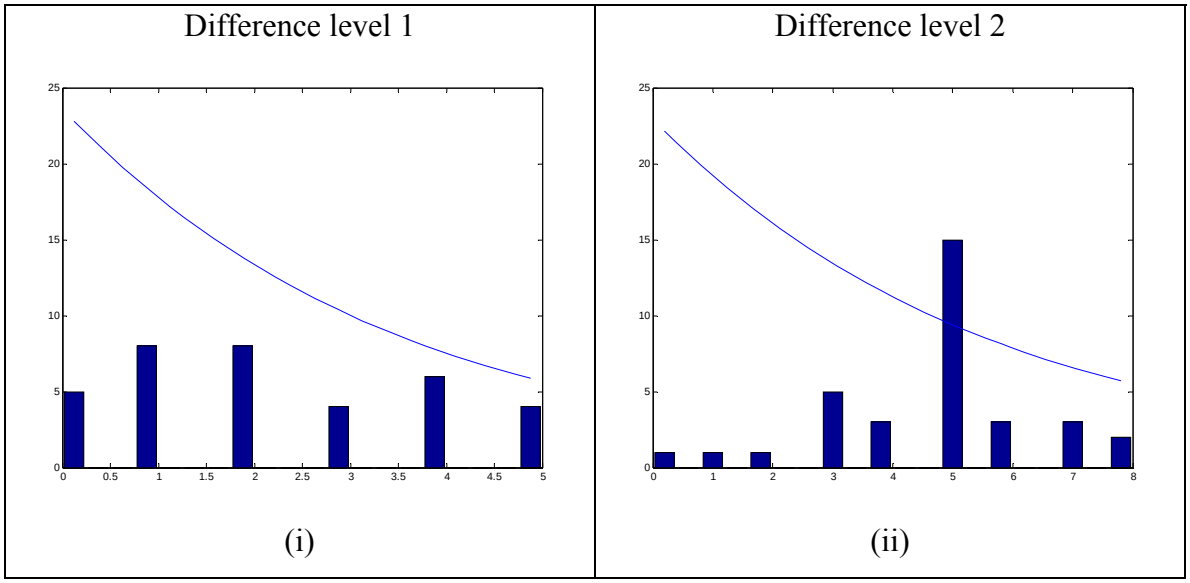


**Figure 12: SDIF histograms up to difference level 2**

## Algorithm 3.3

Sample consists of signals with TOAs = $\{s_1, s_2, \ldots, s_n\}$

Set number of bins $N_{bins}$

**for** $c \leftarrow 1, 2, ...$

Find $c^{\text{th}}$ differences $d_c$, where $(d_c)_i = s_{i+c} - s_i$, for $i = 1, ..., n-c$

Form histogram for $d_c$

Draw threshold $t_k$ and compare with bin heights $h_k$ for k $\leftarrow 1, \ldots, N_{bins}$

Let $L = \{l \mid h_l \geq t_l\} = l_1, l_{2,...}$

$p_{\max} \leftarrow$ interval represented by tallest bin

**if** $p_{l_1}$ is a multiple of $p_{\max}$

Include index of tallest bin into $L$

**end**

**for** $j \in L$

Do sequence search for histogram bin centre $p_k$

**if** the train is successfully extracted,

Reset $c \leftarrow 1$

Repeat SDIF algorithm on the remaining samples

**end**

**end** (**for** *j*)

**end** (**for** *c*)

## Threshold Function

The use of histogram methods can be more efficient than traditional sequence search because trial trains need only be constructed for certain PRI values. It is therefore important to correctly determine these PRI values so that sequence search is not carried

out unnecessarily for false PRI values. The key to doing this is to employ the right threshold function, against which the histogram bins are compared. A threshold that cuts the histogram at incorrect intervals will yield bin intervals that do not correspond to actual PRI values. Sequence search will then be performed more times than necessary, resulting in a waste of resources and undermining the effectiveness of the difference histogram method. In addition, a false PRI may be identified and the wrong signals may be extracted, causing greater difficulty later on.

As mentioned before, the threshold should be a function rather than a constant value since smaller intervals are bound to appear more often when the sample consists of signals from various sources. In [38], the intervals between two signals are considered as random Poisson points and the histogram as the estimate of the probability distribution function of a random event. The function used is

$$Thr(\tau) = x(E-c)e^{-\frac{\tau}{kN}},$$

where $\tau$ is the bin index, $E$ is the number of signals in the interleaved sample, $c$ is the difference level, $N$ is the number of bins, and parameters $x$ and $k$ are positive constants less than one which are determined experimentally. This form of the threshold function is found to follow the histogram peaks closely and was shown to give good results, so we use the same threshold here.

## 3.2. Implementation of Sequence Search and Histogram Methods

Although the difference histogram and sequence search provides a framework for signal sorting, the actual implementation of the methods gave rise to certain issues. Certain parameters had to be introduced and minor adaptations of the sequence search method were needed in order to carry out the process.

### *3.2.1. Implementation Issues*

The difference histograms give the possible intervals for which sequence search can be carried out. In the example used for Figures 11 and 12, both CDIF and SDIF histograms gave PRI = 5 for sequence search, which was indeed the PRI of one of the sequences present in the sample. However, the histogram bin intervals consist of a range of values rather than a single value. When the bin count exceeds the given threshold, the PRI to be used for sequence search can be any of the values within the bin interval since the bin count could have been contributed by any of these values. As such, this should be reflected in the construction of trial train during sequence search. The trial train consists of signals from the sample which will form a sequence with a PRI value lying within that bin interval. The algorithm will be elaborated later in the section after other issues have been addressed.

Another implication for the trial train construction in sequence search arises from the irregularity of signal intervals within PRI sequences. Although signals from the same source occur at regular intervals, this interval may not always be exact within the sequence, but instead deviate slightly from the mean PRI. So during the trial train construction, a certain tolerance should be allowed such that a signal can still be considered for the trial train even if it results in an interval lying outside the bin range, as long as it is within the given allowance. This tolerance should be closely related to the deviation of the PRI from the mean value within the sequence.

In addition to the PRI deviations, the sample may also have missing signals that makes sequence search more difficult. The trial train construction may encounter instances where no signal from the sample can be considered since the resulting PRI of the train would no longer fall within the bin interval or its tolerance. In this situation, a missing signal may be assumed and the construction can continue. However, there cannot be too many consecutive missing signals in the train construction or the process will be inaccurate, so only a certain maximum number of signals can be assumed, beyond which the train construction should be aborted.

### *3.2.2. Algorithm for Sequence Search using Bin Interval*

Given the histogram bin that contains a likely PRI value, the sequence search is implemented as follows:

- From a starting signal in the sample, project the bin interval to obtain a window and search for the next signal in the following order.
    - If there are signals in the window, the first will be used in the construction of the trial train.
    - If there are no signals in the window, search outside and find the nearest which lies within the tolerance allowance.
    - If no signals are found within the tolerance allowance, assume there is a missing signal and find the next signal.
    - If the maximum number of consecutive missing signals has been assumed, trial train construction is deemed to have failed.
- Subsequently find the next signal by following the same procedure
- If trial train construction fails for this starting signal or if the constructed train is too short, restart by using another starting signal.
- If the constructed train satisfies the minimum length, extract those signals from the sample and compute the PRI.

## Example

The algorithm can be better illustrated using an example.

The sample below consists of signals from 3 sources with PRI = 5, 8, 11. There are slight deviations of the PRI from the mean for all 3 PRIs. The bin interval being used in sequence search is 4.9638 to 5.2251 and the key is to search for the signals corresponding to PRI 5, indicated in the sample in bold.

**0**, **5.2251**, 8.1029, **10.0906**, 11.0524, (**15.1441** missing), 16.0852, **20.137**, 21.9812, 24.1221, **25.3327**, **30.4637**

Figures 13 to 17 illustrate the search for the signals. The solid boxes indicate the window obtained by projecting the bin interval, and the dashed line boxes indicate the tolerance allowances. Signals denoted by * are those that are selected for use in the train construction.

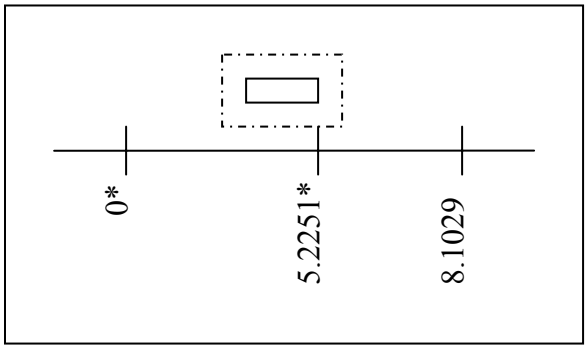


**Figure 13: Example of sequence search using bin interval – Search for first signal**

From the starting signal with TOA = 0, the bin interval was projected and the signal with TOA = 5.2251 was found.

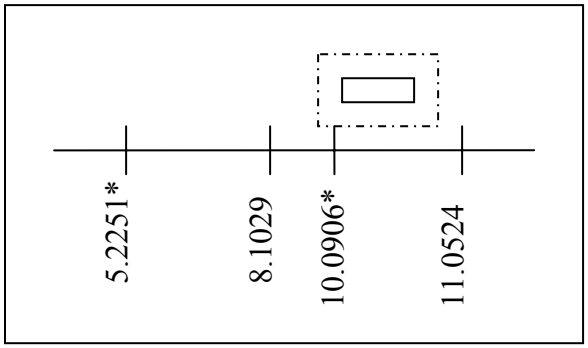


**Figure 14: Example of sequence search using bin interval – Search for signal within tolerance allowance**

From the signal with TOA = 5.2251, the bin interval was projected but no signal was found. However, the signal with TOA = 10.0906 lie within the tolerance allowance and was used in train construction.

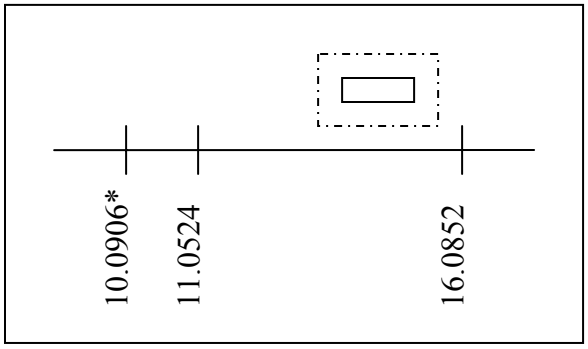


**Figure 15: Example of sequence search using bin interval – No signal found within tolerance allowance**

From the signal with TOA = 10.0906, the bin interval was projected but no bin was found in the window or the tolerance allowance.

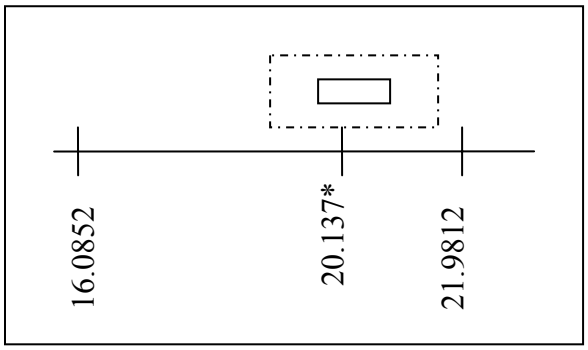


**Figure 16: Example for sequence search using bin interval – Search for next signal after a missing signal is encountered**

A missing signal is assumed and the bin interval is projected again. The signal with TOA = 20.137 was found within the window.

After a number of signals have been found in the trial train, there is a rough idea as to what the PRI might be. So instead of searching in the projected bin interval, the next signal in the trial train can be found by first computing the supposed PRI based on the trial train constructed so far, and then searching for the nearest signal to the projected PRI estimate, and which also lies within the tolerance window.

In the previous example, after those signals have been used in the train construction, the supposed PRI of the partial constructed train was computed to be 5.03425. The next signal in the constructed train can then be found as in Figure 17.

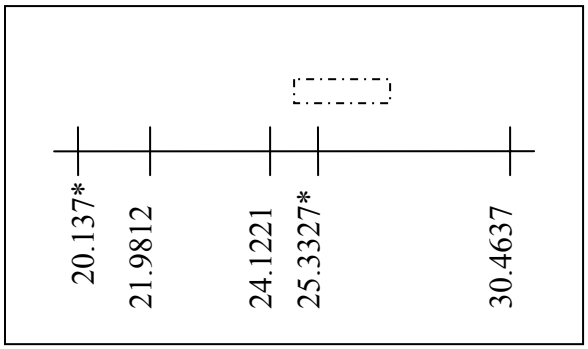


**Figure 17: Example for sequence search using bin interval – Selection of next signal based on supposed PRI**

From the signal with TOA = 20.137, the supposed PRI of 5.03425 was projected to 25.17125 where we expect the next signal to be. The nearest signal has TOA = 25.3327, which lie within the tolerance allowance.

### *3.2.3. Problems Encountered*

After the implementation issues were resolved, the process of signal sorting could be carried out. However, certain difficulties were encountered. The tolerance and threshold function play important roles in the method, as we will elaborate later in the following section, but their parameter values can only be experimentally determined. In addition, the efficacy of the method degraded as the deviations from mean PRI increased.

The tolerance parameter creates a tolerance window during sequence search to accommodate deviations of PRI from the mean value. If the tolerance value is too large, the window will be too wide and could contain too many signals, resulting in inaccurate trial train construction or making it difficult to determine which signal should be used [39]. On the other hand, if the tolerance used is too small, the trial train construction will encounter difficulties in searching for the next signal in the train, especially if there are larger deviations of the PRI from the mean value.

The threshold function is the key to finding the right PRI values for sequence search, and the right threshold parameters are required for the function to cut the right bins in the histogram. If too many irrelevant bins are cut, sequence search will have to be carried out for many false PRIs, which will undermine the histogram methods. Moreover, if the bin containing a multiple of the actual PRI is used for sequence search instead, only part of the actual PRI sequence will be extracted through sequence search and the wrong PRI will be identified. As such, the choice of threshold parameters is important in the signal sorting process and requires appropriate selection. Yet these parameters can only be determined experimentally, which also means that there has to be some knowledge of the sample in order to evaluate which parameter values are suitable. Furthermore, the parameter values for one data sample generally do not suit another data sample [40], so testing will need to be carried out frequently.

The problem of threshold parameters selection is compounded when there are large deviations of the PRI from the mean value. As the deviations increase, the histogram peaks become less distinct and harder to compare with the threshold function, which is illustrated in Figure 18. In Figures 18(a) and (b), the bin with PRI 5 is clearly taller than the other bins and is easily cut by the threshold. But as the amount of deviation from mean PRI increased, the threshold could not cut any bin in Figure 18(c) and no bin was distinctly taller than the others. Sequence search will not be carried out at this difference level for this histogram and higher difference levels will need to be computed. The threshold may cut the correct bin at higher difference levels, but it may also cut a bin containing a multiple of the PRI which could lead to incorrect extraction of the signals.

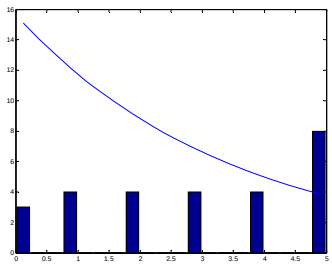

Figure 18(a): PRI 5 and 8 have no deviations. Threshold parameters $x$ = 0.7 and $k$ = 0.7

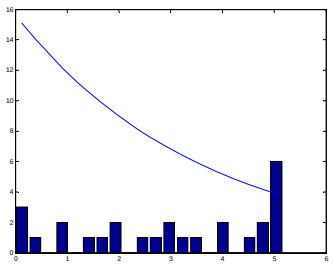

Figure 18(b): PRI 5 has deviation of 5% and PRI 8 has deviation of 2%

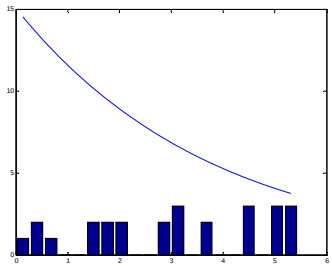

Figure 18(c): PRI 5 and 8 both have deviations of 10%

**Figure 18: Difference histogram of sample with higher deviation**

In addition, the sequence search process will also become more difficult as the deviation from mean PRI becomes larger. The wider range of values that the PRI can take has to be accounted for by a larger tolerance value for sequence search, but this also increases the likelihood of more signals within the tolerance allowance. There will need to be a way to decide which of the signals should be used in the train construction.

## *3.3. Use of Prior Knowledge*

For signal sorting, the data sample consists of signals from various sequences. Unlike generated data used for testing, in actual applications there is no way of telling exactly which signal belongs to which sequence, or how many sequences are present in the sample and what their PRIs are. On the other hand, historical data, records or expert knowledge could be available. This information may help in overcoming the difficulties encountered over the signal sorting process.

For the data used in the signal sorting module of this system, prior knowledge of the sources is available in the form of a database, along with information on other sources whose signals may not be present in the sample. The information for each source include the following: PRI range, which is a range of values within which the PRI can lie,

attributes range, which is a range of values each attribute can take, and the deviation from the mean PRI.

Information that is more relevant to the sample can be drawn from the database. Given the signals in the sample, their attributes can be checked against the attribute ranges in the database to pull out a portion of the entries, which will be referred to as the source table. From this table, higher relevance can be achieved for every bin interval obtained from the histogram method. The source table can be scanned for those entries whose PRI ranges overlap with the given bin interval, to comprise a list of possible matching sources for that bin, which will be referred to as a PRI list which can facilitate sequence search. The level of information extracted from the database can be seen more clearly in Figure 19. The information drawn from the database can then be used to help in selecting the tolerance and threshold parameters, as well as in the sequence search process after a histogram bin likely to contain a PRI is identified.

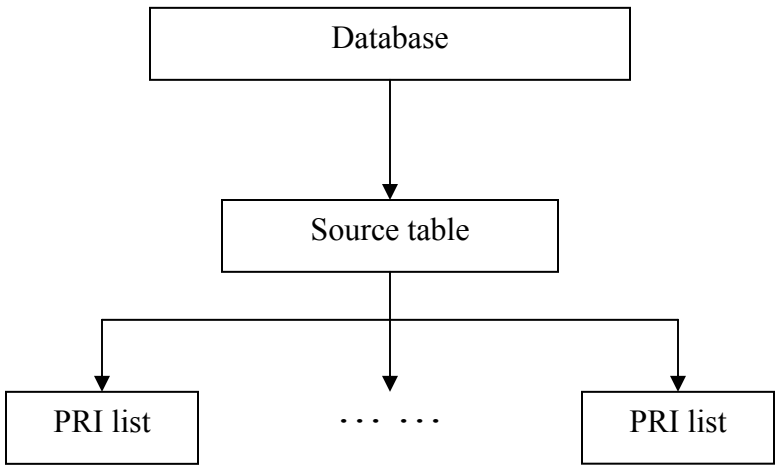


**Figure 19: Graphical view of information drawn from database**

### 3.3.1. Selecting the Tolerance Parameter

As seen earlier, the tolerance has to be appropriately set when doing sequence search, and this selection can be guided by the prior knowledge given in the database. For a given bin interval, a PRI list can be generated. This list consists of the possible sources and the probable deviation of PRI from the mean value for each source. The tolerance parameter can then be set to the highest deviation from mean PRI for the sources in this list.

### 3.3.2. Selecting the Threshold Parameters

The histogram bins are compared to the threshold function to obtain the bin intervals for carrying out sequence search. The parameters $x$ and $k$ can be adjusted using the prior information to find a threshold function that correctly cuts the histogram bins.

A bin is said to tally with the source table if its bin range overlaps with the PRI range of at least once source entry in the source table. If the threshold drawn using the initial parameter values cuts a bin that tallies with the source table, no change is made to the parameter values and sequence search is carried out as usual on the bin interval. However, if the threshold does not cut any bin, or if it cuts a bin that does not tally with the source table, adaptation of $x$ and $k$ is carried out. The values are adjusted such that the threshold cuts the tallest bin which tallies with the source table, without violating the restriction on the value range ($x$ and $k$ are positive values less than 1). The detailed steps are shown in the following section.

## Adaptation of x and k using prior knowledge

1. Find $b_t$:

   Sort the N bins $\{b_1, b_2, ..., b_N\}$ according to height and select the tallest bin $b_t$ that tallies with the source table. The threshold will later be modified such that $b_t$ is the first bin it will cut.

2. Find $b_m$:

   Since bin $b_t$ is the first bin that the threshold should cut, consider the bins before it. For each bin $\{b_1, b_2, ..., b_t\}$, compute the corresponding values $\{x_1, x_2, ..., x_t\}$ for parameter $x$ such that the threshold just cuts each bin at its tip. Identify $x_m = \max\{x_1, ..., x_{t-1}\}$, and consider its corresponding bin $b_m$.

3. Consider if $b_m$ is shorter than $b_t$:

   If bin $b_t$ is the tallest bin of $\{b_1, b_2, ..., b_t\}$, choose $x \leftarrow average\{x_t, x_m\}$ so that the threshold will cut the bin with a comfortable allowance from its tip as shown in Figure 20.

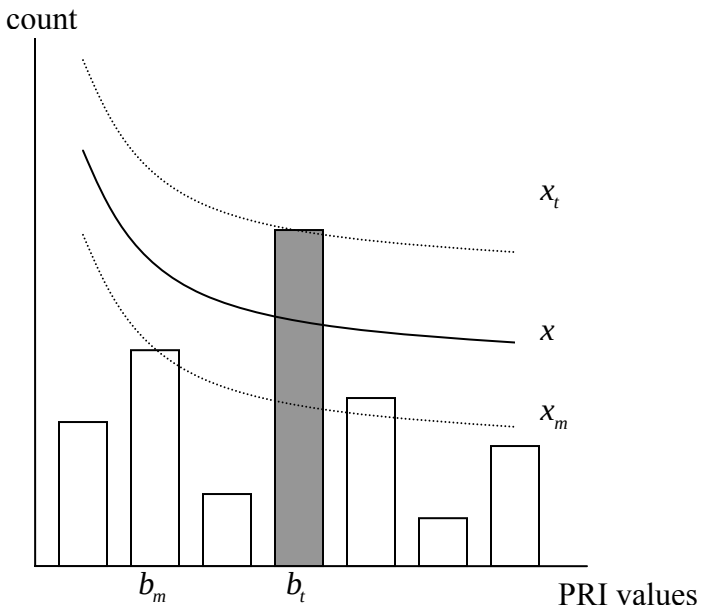


**Figure 20: Position of threshold function if chosen bin is taller than those before it**

4. Consider if $b_m$ is taller than $b_t$:

   If there is an earlier bin $b_m$ that is taller than $b_t$, change $x$ and $k$ such that the threshold still cuts $b_t$ at the same point but will not cut $b_m$. In Figure 21, the height of the bin $b_m$ is given by $h_m$.

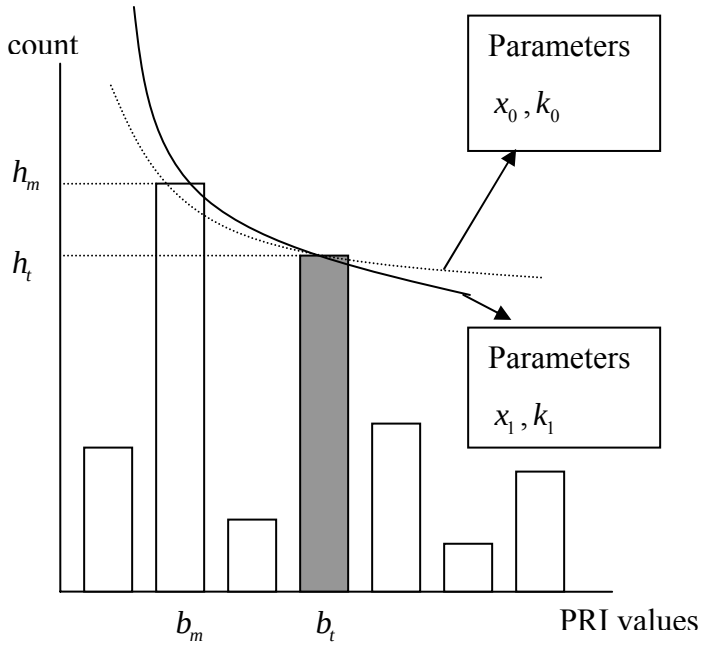


**Figure 21: Position of threshold function if chosen bin is shorter than one before it**

Using the original values $x_0$ and $k_0$, we can find new values $x_1$ and $k_1$ as follows:

a. Formulate simplified function:

   The threshold function, given by

$$Thr(\tau) = x(E - c)e^{-\frac{\tau}{kN}},$$

   can be written as the function $v = pe^{-qu}$ shown in Figure 22.

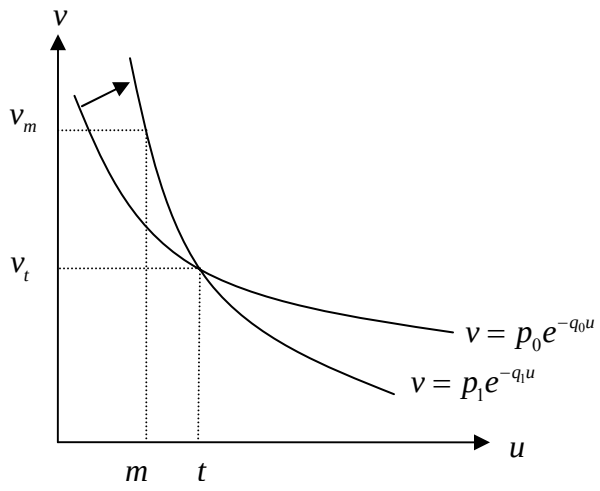


**Figure 22: Simplified view of original and desired threshold function**

On the horizontal axis, *u* represents the index of the histogram bin; on the vertical axis, *v* represents the height of the bin.

b. Matching each graph to its respective threshold function:

The graph $v = p_0 e^{-q_0 u}$ represents the original threshold with values $x_0$ and $k_0$, passing through the point $(t, v_t)$, where $v_t = h_t$. The values $p_0$ and $q_0$ can be found by

$$p_0 = x_0 (E - c)$$

and

$$q_0 = \frac{1}{k_0 N}.$$

The graph $v = p_1 e^{-q_1 u}$ represents the desired threshold with new values $x_1$ and $k_1$, passing through the same point $(t, v_t)$ and also the point $(m, v_m)$, where $v_m = h_m + \varepsilon$, for a small positive value $\varepsilon$. The values of $p_1$ and $q_1$ can be found by formulating the following equations.

c. Finding the values $p_1$ and $q_1$:

Since the graph $v = p_1 e^{-q_1 u}$ should pass through the point $(m, v_m)$, we have $v_m = p_1 e^{-q_1 m}$. In addition, both graphs pass through the point $(t, v_t)$, giving

$v_t = p_0 e^{-q_0 t} = p_1 e^{-q_1 t}$. With these two equations, the values $p_1$ and $q_1$ can be computed by

$$q_1 = \frac{1}{t-m} \log\left(\frac{v_m e^{q_0 t}}{p_0}\right)$$

and

$$p_1 = v_m e^{q_1 m}.$$

d. Computing $x_1$ and $k_1$:

The new values of the parameters can then be found using

$$x_1 = \frac{p_1}{E-c}$$

and

$$k_1 = \frac{1}{q_1 N}.$$

5. Check range of new *x* and *k*:

Both the threshold parameter values must lie in the range (0,1). If either of the values returned by the above steps violates this restriction, the adaptation of *x* and *k* is deemed to have failed.

If no values of *x* and *k* are found in the above steps, it could mean that the source is not included in the table at all, or that higher difference levels are required. No adjustments will be made to the original values of *x* and *k*. Sequence search is carried out as normal if the existing threshold cuts some bin. Otherwise, the next difference level will be computed.

Threshold parameter values *x* and *k* that lead to the successful extraction of PRI and trial trains are saved for reference in subsequent rounds of threshold comparisons. The starting values of *x* and *k* are assigned with the average of the previous two values they have took on, and they will be used without change if the threshold cuts the right bin. This way, the parameters may adapt to the suitable value over time and the need for adjustment is reduced.

Figure 23 shows the histogram peaks resulting from the data used to obtain Figure 18(c). The sample consists of two sequences with PRI 5 and 8, each with deviation 10% from the mean PRI. The original threshold function cannot cut any of the bins at all.

However, given information consisting of the possible PRIs that are included in the sample, the threshold parameters can be adapted such that the correct bin can be cut and sequence search activated. The information used in this example is shown in Table 11. Besides the two sources with PRI 5 and 8, there is information on other sources as well.

**Table 11: Example of information used in adaptation of x and k**

| Possible PRIs in the interleaved sample (range) | |
|---|---|
| Lower limit in range | Upper limit in range |
| 1.3 | 1.8 |
| 2.5 | 3.1 |
| 4.9 | 5.1 |
| 7.6 | 8.5 |
| 10 | 11.8 |

Using the original threshold parameter values, the information above was used to find new values as shown in Table 12. The new values adjusted the threshold function so that it can now cut the bin as seen in Figure 23.

**Table 12: Values of x and k before and after adaptation**

| | Initial values | New values |
|---|---|---|
| x | 0.6 | 0.41844 |
| k | 0.7 | 0.7 |

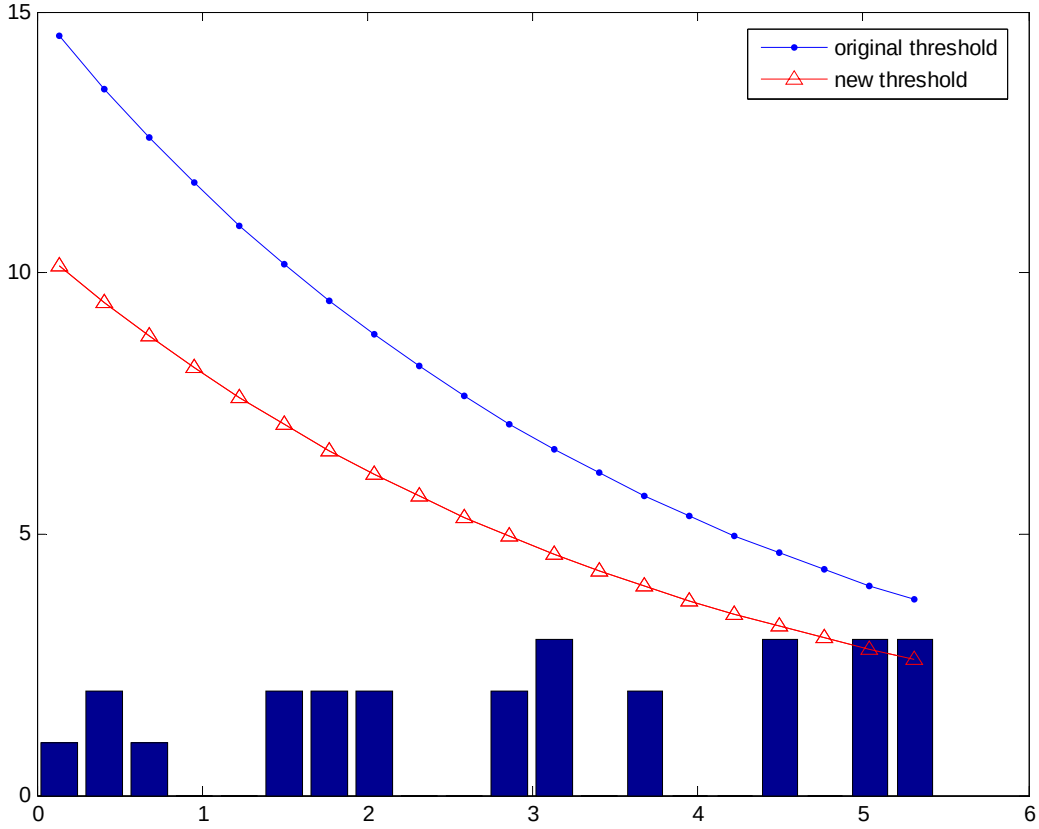


**Figure 23: Thresholds before and after adaptation**

### *3.3.3. Trial Train Construction in Sequence Search*

In the construction of a trial train during sequence search, the prior knowledge can lend greater credibility to the selection of the next signal in a trial train. Based on the bin interval, the PRI list can be generated and the attributes of the signals can be checked against the list in certain situations. Furthermore, when a starting signal for train construction has been decided on, a list of likely sources can be obtained from the PRI list by checking which source entries have attributes ranges that match the starting signal. The prior knowledge can be used in the following ways.

1. For a given histogram bin, trial train construction requires a starting signal, and the attempt is aborted only if the constructed train does not satisfy the minimum

length. The construction will restart with the next signal in the sample as starting signal. If a PRI sequence starts with a late TOA, many incorrect train constructions will need to be carried out using the wrong starting signal, resulting in a waste of resources. This number of unnecessary attempts can be reduced by using the PRI list to rule out those signals whose attributes do not match the given bin at all.

2. When searching within the window of projected bin interval, there may be more than one signal. Instead of simply selecting the signal which appears first, the attributes of these signals are checked against the list of likely sources. The first signal whose attributes match the list is selected. If no such signal is found, we will search outside the window within the tolerance allowance.
3. When searching within the tolerance allowance outside the window of projected bin interval, there may again be more than one signal lying within that tolerance. Rather than selecting the signal nearest to the original window, the attributes of these signals within the tolerance allowance are checked against list of likely sources. The signal closest to the original window which also matches the attributes will be selected. The same criterion is used when finding a signal closest to the supposed PRI (using the variation of the algorithm) after a certain length of trial train has been constructed.

Depending on the attributes of the signals, the sequence search process may or may not benefit much from the prior information. However, it could help to at least rule out signals that clearly do not belong to the PRI sequence corresponding to the current bin interval.

## *3.4. Results*

In this section, “basic sort” refers to signal sorting without using prior knowledge and “prior sort” refers to signal sorting using prior knowledge. The samples used for testing

are generated by combining signals from different classes. The PRIs of the trains in each class are given in the Table 13.

**Table 13: Class and PRI knowledge of data used**

| Class | PRI | Amount of deviation |
|---|---|---|
| K | 1,000,000 | 0% |
| H | 2,999,750 | 2% |
| C | 1,195,008 | 2.5% |
| I | 2,998,690 | 5% |
| L | 2,953,189 | 7% |
| M | 2,379,376 | 7% |
| F | 2,888,262 | 10% |
| D | 909,668 | 14% |
| E | 1,054,285 | 14% |

The signals from the respective classes are combined together and separated into smaller portions. The length of each portion is such that there are at least 10 signals from every source class, so the number of signals in the portion would depend on the PRI of the source classes. The signal sorting program is tested out on each of these portions. For basic sort, the selection of threshold parameters $x$ and $k$ will affect the ability of the algorithm to extract the right PRIs, so the parameters values are incremented from 0.4 to 0.8 at steps of 0.1 to find the most suitable values for $x$ and $k$. The values that yielded the best performance are then used and the results are recorded. Each portion is tested separately for both algorithms, but the performance of basic sort varies across different portions, so only the best result is shown here; the result for the corresponding portion is shown for prior sort.

In basic sort, the tolerance parameter is set to a default value of 0.2, while the threshold and tolerance parameters for prior sort will vary depending on the information drawn from the prior knowledge. The CDIF histogram is used for both methods, and difference

levels are computed up to a maximum of 4 levels. Tables 14 to 19 consists of the PRIs extracted and the class accuracy of the corresponding sequence extracted. For each sample, the time taken to complete signal sorting for that portion is shown, as well as the average time taken over the various portions of the sample. In both cases, the time taken is averaged over the different threshold parameters used for scanning in basic sort.

### *3.4.1. Results for Sample with 2 classes*

**Table 14: Classes C and I with deviation 2.5% and 5%**

| | Without prior knowledge | With prior knowledge |
|---|---|---|
| Correct PRIs extracted<br>(class accuracy) | 1,195,836 (100%)<br>2,983,785 (100%) | 1,195,836 (100%)<br>2,983,785 (100%) |
| Time taken | 0.0547s | 0.0672s |
| Average time required | 0.0526s | 0.0438s |

With little deviation from the mean PRI, both basic and prior sort were able to identify the PRIs and accurately extract the signals in the respective sequences. No signals remained in the sample after the signal sorting process.

The first time taken is that for the portion of the sample with best performance, using the most suitable threshold parameter values found by testing. Basic sort took less time than prior sort, which has to first check if the parameters are suitable, and also check the attributes of the signals over the course of sequence search. However, when averaged over all the portions in the sample and all the threshold parameter values tested, basic sort took slightly longer than prior sort. In the process of finding the best parameter values, many are unsuitable and triggers sequence search for irrelevant bins. This leads to failed attempts at train construction or eventually the wrong trains being extracted, thus taking up more time.

**Table 15: Classes D and E with deviations 14% and 14%**

| | Without prior knowledge | With prior knowledge |
|---|---|---|
| Correct PRIs extracted<br>(class accuracy) | 868,110 (100%)<br>1,063,645 (100%) | 886,614 (100%)<br>1,063,647 (100%) |
| Time taken | 0.0500s | 0.0784s |
| Average time required | 0.0814s | 0.0491s |

Both basic and prior sort were able to correctly extract the signals in the respective sequences, but one signal remained in the sample for basic sort, resulting in the PRI being slightly different from that found for prior sort.

Over the various portions of the sample and using different threshold parameter values, basic sort often extracted too many incorrect sequences, thus the average time taken is much more than that for prior sort.

### *3.4.2. Results for Sample with 3 Classes*

**Table 16: Classes K, C and H with deviations 0%, 2.5% and 2%**

| | Without prior knowledge | With prior knowledge |
|---|---|---|
| Correct PRIs extracted<br>(class accuracy) | 996,529 (65%)<br>1,199,091 (88%)<br>3,000,001 (100%) | 1,000,000 (100%)<br>1,195,835 (100%)<br>2,997,727 (100%) |
| Time taken | 0.1003s | 0.1197s |
| Average time required | 0.0889s | 0.1000s |

With more sequences in the sample, basic sort was still able to identify the PRIs although the sequences sometimes included signals belonging to other PRIs. Prior sort did not include any irrelevant signals within each sequence as checks are carried out on the attribute values.

In terms of the time taken to carry out signal sorting, the durations for both methods were comparable. The average time required for basic sort is slightly less than for prior sort, unlike in previous samples, since irrelevant sequences were less frequently extracted.

**Table 17: Classes D, E and H with deviations 14%, 14% and 2%**

| | Without prior knowledge | With prior knowledge |
|---|---|---|
| Correct PRIs extracted (class accuracy) | 905,965 (83%)<br>1,059,949 (82%)<br>2,986,840 (80%) | 905,965 (100%)<br>1,059,949 (100%)<br>2,984,432 (100%) |
| Time taken | 0.0269s | 0.0822s |
| Average time required | 0.1028s | 0.0839s |

In the portion of the sample that was easiest to sort the signals, the right threshold parameter values enabled the process to be completed in a much shorter time for basic sort than for prior sort. However, the other parameter values used in the process of testing gave considerably poorer results. Over the various portions and parameter values, signal sorting is carried out a number of times. The right trains were extracted only 14% of the time. For 44% of the time, the threshold function cut too many irrelevant bins, leading to the extraction of many incorrect sequences and taking up additional computational time. And in most of the remaining cases, no sequences were extracted at all. Due to the failure to apply sequence search on the right bins, a large number of failed or incorrect attempts resulted in the average time needed to be significantly more than when using the right parameter values.

### *3.4.3. Results for Sample with 4 Classes*

**Table 18: Classes C, I, D and E with deviations 2.5%, 5%, 14% and 14%**

| | Without prior knowledge | With prior knowledge |
|---|---|---|
| Correct PRIs extracted (class accuracy) | 902,925 (66%)<br>*1,216,257 (52%)**<br>1,034,477 (62%)<br>3,031,665 (44%)<br>*1,167,173 (75%)** | 902,925 (100%)<br>1,192,719 (100%)<br>1,050,118 (100%)<br>2,970,803 (100%) |
| Time taken | 0.0381s | 0.1278s |
| Average time required | 0.1477s | 0.1085s |

With basic sort, most of the PRIs could be identified. However, the histogram bin found was sometimes slightly off the actual PRI, leading to the inaccurate extraction of sequences with PRI 1,216,257 and 1,167,173 (denoted by * in Table 18) instead of just 1,192,719.

Like in the previous sample, the average time over the various portions and parameter values was considerably longer than when using the right values. In addition, the sequences could not be correctly extracted. The correct PRIs could be identified but the corresponding sequences are inaccurate, and a false PRI would also appear along with the rest.

**Table 19: Classes E, F, L and M with deviations 14%, 10%, 7% and 7%**

| | Without prior knowledge | With prior knowledge |
|---|---|---|
| Correct PRIs extracted (class accuracy) | 1,066,126 (80%)<br>2,370,882 (62%)<br>2,908,508 (80%)<br>2,952,050 (86%) | 1,066,122 (100%)<br>2,370,882 (85%)<br>2,863,179 (100%)<br>2,893,705 (78%) |
| Time taken | 0.0362s | 0.1741s |
| Average time required | 0.0806s | 0.1290s |

Although prior sort extracted the signals more accurately than basic sort, it still could not achieve 100% accuracy for two of the sequences. This is because their attributes are the same and both their PRIs lie within the same bin that was used for sequence search. However, since the sequences for the other PRIs have been ruled out, the sequence search was still fairly accurate.

Generally, the right PRIs and their sequences could be extracted rather quickly if the right threshold parameters were used. However, over the course of parameter scanning, the threshold function was often unable to cut the bins and no sequence search was carried out at all, hence no sequences were extracted and the process terminates early.

## *3.5. Discussion*

The results for basic sort are such that the portion of the sample with best performance is shown, using the most suitable threshold parameter values. Prior sort shows the results on the same portion of the sample, but the threshold parameter values are left to adaptation.

Over the sample sets tested, although both methods were able to identify about the same PRI, prior sort gave consistently better accuracy for the sequences that were extracted. This is mainly due to the check in the attributes values during the trial train construction. Despite the checks, the accuracy will not always be 100%. Given a histogram bin for sequence search, the PRI list can be found, containing the possible sources whose PRI range matches the bin interval. During the trial train construction process, the likely sources can be found based on the starting signal of the trial train. The presence of more than one likely source could lead to the selection of signals that do not belong to the current sequence. In addition, even if there is only one likely source, the attribute range may cover a large interval, leading to the selection of signals from other sources.

Although the accuracy of sequences extracted in the sample was not affected much, this gives an insight as to how the performance of prior sort is dependent on the nature and quality of the database supplied. As this knowledge is used in three main areas, let us take a closer look at how each area can possibly be affected by certain flaws of the database.

The selection of tolerance parameter is based on the PRI deviation of the sources. Given the PRI list for a histogram bin, the tolerance is set to the highest PRI deviation for the sources in the PRI list. If this list contains too many irrelevant sources, or includes certain sources whose PRI deviation is exceptionally high, the tolerance may be set too high and affect the sequence search process, as too many signals are taken into consideration. However, the consequences are not too drastic since the attributes are still taken into consideration in the sequence search process, as compared to merely selecting the first signal encountered, which would have been the case if basic sort were used.

The threshold parameters $x$ and $k$ are adapted using the PRI ranges in the source table, which contains all the possible sources, obtained by comparing the attributes of the signals to those in the database. If the first bin cut by the threshold does not tally with the source table, the parameters $x$ and $k$ will undergo adaptation. They are chosen such that the threshold can cut the tallest bin which tallies with the source table. This process works on the premise that all the sources are in the source table, so problems arise when the information about that source is missing. This may be caused by incomplete data or errors in the database. For example, there could be a disparity between the attribute range listed for the source entry and the signals themselves, resulting in its omission from the source table. As a result, threshold parameter adaptation may fail or the threshold may not be able to cut the right bin. Even though prior sort will then be unable to extract the sequence for this bin, the signals left in the sample can be processed again using basic sort. With the removal of some sequences, it will be less difficult and complicated to extract the remaining using just basic sort.

Another problem that could arise in threshold parameter adaptation is when the threshold cuts the wrong bin. Although the selection of $x$ and $k$ values depend on the source table

entries as well as the height of the bins, the presence of certain types or irrelevant entries in the source table could impede the process and result in the threshold cutting the wrong bin. This is especially so for some entries whose PRI range covers a relatively small value that corresponds to a difference between PRIs of two sequences. In Figure 24, the sample consists of sequences with PRIs 5, 8 and 11. In the first difference level, the histogram bin with highest peak is centered at 1.875 rather than the PRI value 5. If the source table contains an entry with a PRI range corresponding to 1.875, the threshold parameters will be chosen such that this bin is cut. The attributes check during sequence search could prevent the incorrect extraction of such a sequence, although resources are still wasted, or the wrong sequence may be extracted altogether and affect the process.

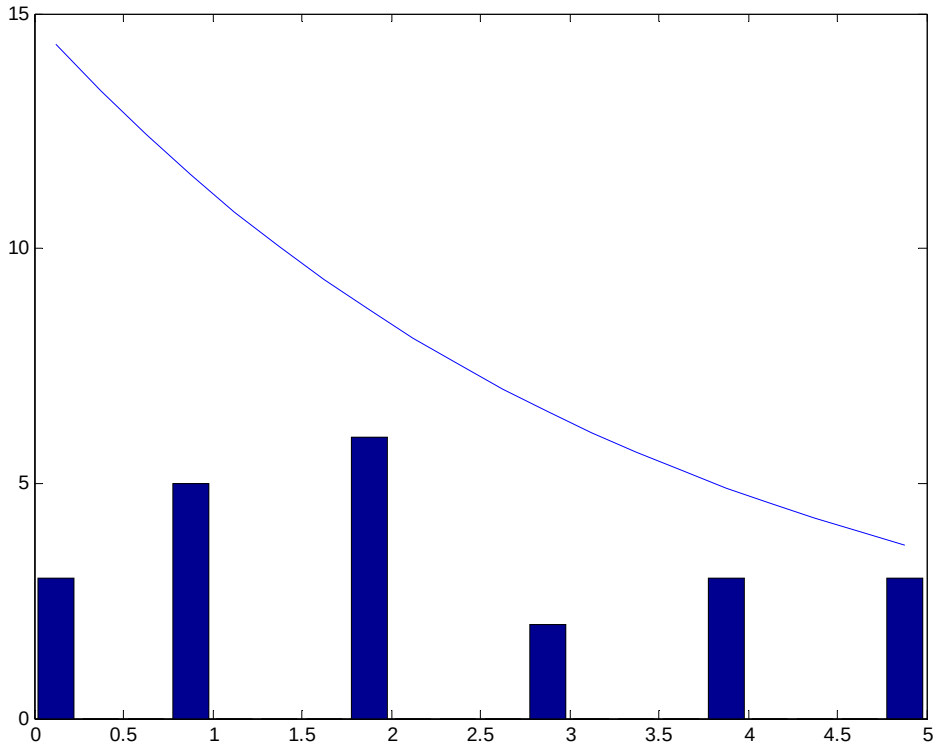


**Figure 24: Histogram peaks at values less than smallest PRI**

Finally, prior knowledge is used for attributes check in the sequence search process. Given the histogram bin, the entries in the source table are considered to find those with corresponding PRI ranges, in order to obtain the PRI list. Here, problems can arise due to

errors, or the nature of the source entry itself. In the case where the attribute range of the relevant source entry is erroneous, trial train extraction for that PRI may reject certain signals even though they belong to the sequence. Even if there are no errors in the attribute range but it covers a large interval, it will be more difficult to decide between the signals that lie within projected bin interval or tolerance window during trial train construction, hence rendering the attributes check less effective and not much different from using basic sort.

# Chapter 4. Conclusion

In this project, we looked into two components of a classification system to explore ways to improve them. The fuzzy adaptive resonance theory map (fuzzy ARTMAP) classification had difficulty dealing with data from overlapping classes, while the signal sorting module could be improved by incorporating available prior knowledge.

In dealing with data from overlapping classes in fuzzy ARTMAP, the classification process was altered to allow multiple class prediction, and the accuracy measure amended accordingly. Together with a variant known as fuzzy ARTMAP with match tracking - (read as 'minus'), the use of single epoch training and ordered training input presentation reduced the number of categories produced significantly, while keeping the classification accuracy unchanged or even improved. All these measures do not require major changes to the fuzzy ARTMAP architecture, and makes both the training and classification process more efficient than before.

However, further investigation can be carried out on certain aspects. The training input data were presented class by class in order of class index, but different results may be obtained by presenting the classes in some different order, depending on which classes are overlapping with one other. Although merging of categories produced little improvements and was eventually omitted, the method can be refined by computing the centroid of a hyperbox based on the patterns it coded rather than its geometry.

Prior knowledge was incorporated into the signal sorting process, enabling the threshold parameters to be set without scanning values for them, thus saving on computational time. Prior knowledge in the form of a database also lent credibility and greater certainty to the sequence search process by allowing a check on the attribute values on top of checking the time-of-arrival of each pulse. With the use of this information, all the pulse repetitive

intervals (PRIs) in the experimental data could be identified and their respective sequences accurately extracted, even when there is a larger deviation of the PRIs from the mean value.

Nevertheless, more work is needed to determine the effect of the quality of the prior knowledge provided on the signal sorting results. The adaptation of threshold parameter values could fail due to errors in the database or misleading entries, and the sequence search process could be impeded by errors in the attribute ranges. In addition, certain sources in the sample may be missing from the database. To allow parameter selection to proceed without the need for scanning, weights may be assigned to each histogram bin in the adaptation process. Further consideration is also needed to ensure that even with errors in the attribute ranges, the sequences extracted by using prior knowledge are at least as accurate as without using it.

For a PRI that is not listed in the database, prior sort may not be able to remove it and it could be left behind after the process is completed. Basic sort can be applied on the remaining sample then, and the selection of threshold parameter values can take into account the values that have previously been used.

Errors in the database can compromise the performance of prior sort but it can still give better results than if no prior knowledge were used at all. More work can be done to ensure that the performance or prior sort using a database containing errors is at least as good as basic sort, otherwise the process should be stopped and basic sort can be used on its own.

# References


[1] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*. New York: Ellis Horwood, 1994.

[2] M. Bote-Lorenzo, Y. Dimitriadis, and E. Gómez-Sánchez, "A hybrid two-stage fuzzy ARTMAP and LVQ neuro-fuzzy system for online handwriting recognition," in *International Conference on Artificial Neural Networks*, 2002, pp. 82-82.

[3] J. S. Suri and R. M. Rangayyan, *Recent Advances in Breast Imaging, Mammography, and Computer-Aided Diagnosis of Breast Cancer*. Bellingham, Washington: SPIE Press, 2006.

[4] A. Ahad, A. Fayyaz, and T. Mehmood, "Speech recognition using multilayer perceptron," in *Proceedings of IEEE Students Conference*, 2002, pp. 103-109.

[5] W. Y. Sit, L. O. Mak, and G. W. Ng, "Managing category proliferation in fuzzy ARTMAP caused by overlapping classes," *resubmitted to IEEE Transactions on Neural Networks,* 2008.

[6] A. Andres-Andres, E. Gomez-Sanchez, and M. L. Bote-Lorenzo, "Incremental rule pruning for fuzzy ARTMAP neural network," in *Proceedings on Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, Part 2*. Vol. 3697, Berlin: Springer-Verlag Berlin, 2005, pp. 655-660.

[7] E. P. Hernandez, E. G. Sanchez, Y. A. Dimitriadis, and J. L. Coronado, "A neuro-fuzzy system that uses distributed learning for compact rule set generation," in *Proceedings on 1999 IEEE International Conference on Systems, Man, and Cybernetics*, 1999, pp. 441-446.

[8] M. A. Rubin, "Application of fuzzy ARTMAP and ART-EMAP to automatic target recognition using radar range profiles," *Neural Networks,* Vol. 8, 1995, pp. 1109-1116.

[9] E. G. Sanchez, Y. A. Dimitriadis, J. M. Cano-Izquierdo, and J. L. Coronado, "MicroARTMAP: use of mutual information for category reduction in fuzzy

ARTMAP," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 2000, pp. 47-52.

[10] D. Charlampidis, T. Kasparis, M. Georgiopoulos, and J. Rolland, "A fuzzy ARTMAP based classification technique of natural textures," in *18th International Conference of the North American Fuzzy Information Processing Society*, 1999, pp. 507-511.

[11] B. Vigdor and B. Lerner, "Accurate and fast off and online fuzzy ARTMAP-based image classification with application to genetic abnormality diagnosis," *IEEE Transactions on Neural Networks,* Vol. 17, 2006, pp. 1288-1300.

[12] H. Min, T. Xiaoliang, and C. Lei, "An improved fuzzy ARTMAP network and its application in wetland classification," in *Proceedings on 2004 IEEE International Geoscience and Remote Sensing Symposium*, 2004, pp. 3432-3435.

[13] E. Parrado-Hernandez, E. Gomez-Sanchez, and Y. A. Dimitriadis, "Study of distributed learning as a solution to category proliferation in Fuzzy ARTMAP based neural systems," *Neural Networks,* Vol. 16, 2003, pp. 1039-1057.

[14] T. Kasuba, "Simplified fuzzy ARTMAP," *AI Expert,* Vol. 8, 1993, pp. 19-25.

[15] B. W. Jervis, T. Garcia, and E. P. Giahnakis, "Probabilistic simplified fuzzy ARTMAP (PSFAM) [and application to biosignal data]," *IEE Proceedings - Science, Measurement and Technology,* Vol. 146, 1999, pp. 165-169.

[16] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Transactions on Neural Networks,* Vol. 3, 1992, pp. 698-713.

[17] T.-H. Lin and V.-W. Soo, "Pruning fuzzy ARTMAP using the minimum description length principle in learning from clinical databases," in *Proceedings on 9th IEEE International Conference on Tools with Artificial Intelligence*, 1997, pp. 396-403.

[18] P. Henniges, E. Granger, and R. Sabourin, "Factors of overtraining with fuzzy ARTMAP neural networks," in *Proceedings on 2005 IEEE International Joint Conference on Neural Networks*, 2005, pp. 1075-1080.

[19] M. Georgiopoulos, A. Koufakou, G. C. Anagnostopoulos, and T. Kasparis, "Overtraining in fuzzy ARTMAP: Myth or reality?," in *Proceedings on International Joint Conference on Neural Networks*, 2001, pp. 1186-1190.

[20] G. A. Carpenter and A.-H. Tan, "Rule extraction: From neural architecture to symbolic representation," *Connection Science,* Vol. 7, 1995, pp. 3-27.

[21] C. J. Lee, C. G. Yoon, and C. W. Lee, "A new learning method to improve the category proliferation problem in fuzzy ART," in *Proceedings on 1995 IEEE International Conference on Neural Networks*, 1995, pp. 1393-1396.

[22] J. S. Lee, C. G. Yoon, and C. W. Lee, "Learning method for fuzzy ARTMAP in a noisy environment," *Electronics Letters,* Vol. 34, 1998, pp. 95-97.

[23] G. A. Carpenter, B. L. Milenova, and B. W. Noeske, "Distributed ARTMAP: a neural network for fast distributed supervised learning," *Neural Networks,* Vol. 11, 1998, pp. 793-813.

[24] J. R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks,* Vol. 9, 1996, pp. 881-897.

[25] S. J. Verzi, G. L. Heileman, M. Georgiopoulos, and M. J. Healy, "Boosted ARTMAP," in *(IEEE World Congress on Computational Intelligence) Proceedings on the 1998 IEEE International Joint Conference on Neural Networks*, 1998, pp. 396-401.

[26] R. Andonie and L. Sasu, "Fuzzy ARTMAP with input relevances," *IEEE Transactions on Neural Networks,* Vol. 17, 2006, pp. 929-941.

[27] G. A. Carpenter and N. Markuzon, "ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases," *Neural Networks,* Vol. 11, 1998, pp. 323-336.

[28] I. Dagher, M. Georgiopoulos, G. L. Heileman, and G. Bebis, "An ordering algorithm for pattern presentation in fuzzy ARTMAP that tends to improve generalization performance," *IEEE Transactions on Neural Networks,* Vol. 10, 1999, pp. 768-778.

[29] C. Christodoulou and M. Georgiopoulos, *Applications of neural networks in electromagnetics*. Boston, MA: Artech House, 2001.

[30] P. Horton and K. Nakai, "A probabilistic classification system for predicting the cellular localization sites of proteins.," in *Proceedings of the Fourth International Conference on Intelligent Systems in Molecular Biology*, 1996, pp. 109-15.

[31] L. Tjen-Sien, L. Wei-Yin, and S. Yu-Shan, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine Learning,* Vol. 40, 2000, pp. 203-228.

[32] A. W. Ata'a and S. N. Abdullah, "Deinterleaving of radar signals and PRF identification algorithms," *Radar, Sonar & Navigation, IET,* Vol. 1, 2007, pp. 340-347.

[33] H. K. Mardia, "Adaptive multidimensional clustering for ESM," in *IEE Colloquium on Signal Processing for ESM Systems*, 1988, pp. 5/1-5/4.

[34] V. Chandra and R. C. Bajpai, "ESM data processing parametric deinterleaving approach," in *Technology Enabling Tomorrow : Computers, Communications and Automation towards the 21st Century, IEEE Region 10 International Conference*, 1992, pp. 26-30.

[35] P. Fitch, "Is there a role for artificial intelligence in future electronic support measures?," in *Knowledge-Based Intelligent Information and Engineering Systems*, 2006, pp. 523-530.

[36] H. K. Mardia, "New techniques for the deinterleaving of repetitive sequences," *Radar and Signal Processing, IEE Proceedings F,* Vol. 136, 1989, pp. 149-154.

[37] B. E. S. Bildøy, "Satellite Cluster Consepts: A system evaluation with emphasis on deinterleaving and emitter recognition," in Masters Thesis, *Department of Electronics and Telecommunications*, Norwegian University of Science and Technology, 2006, p. 77.

[38] D. J. Milojevic and B. M. Popovic, "Improved algorithm for the deinterleaving of radar pulses," *Radar and Signal Processing, IEE Proceedings F,* Vol. 139, 1992, pp. 98-104.

[39] G. Noone and S. D. Howard, "Deinterleaving radar pulse trains using neural networks," Defence Science and Technology Organisation, Salisbury, South Australia, 1996.

[40] Y. Kuang, Q. Shi, Q. Chen, L. Yun, and K. Long, "A simple way to deinterleave repetitive pulse sequences," in *7th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering*, Sofia, 2005, pp. 218-222.