EMITTER IDENTIFICATION WITH INCREMENTAL LEARNING USING SYMBOLIC REPRESENTATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AYBÜKE EROL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

EMITTER IDENTIFICATION WITH INCREMENTAL LEARNING USING SYMBOLIC REPRESENTATIONS

submitted by **AYBÜKE EROL** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _______________

Prof. Dr. A. Aydın Alatan
Supervisor, **Electrical and Electronics Engineering, METU** _______________

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _______________

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering, METU _______________

Assoc. Prof. Dr. Fatih Kamışlı
Electrical and Electronics Engineering, METU _______________

Assist. Prof. Dr. Elif Vural
Electrical and Electronics Engineering, METU _______________

Assist. Prof. Dr. Gökhan Koray Gültekin
Electrical and Electronics Engineering, AYBU _______________

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname:	Aybüke Erol
Signature :	

ABSTRACT

EMITTER IDENTIFICATION WITH INCREMENTAL LEARNING USING SYMBOLIC REPRESENTATIONS

Erol, Aybüke
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. A. Aydın Alatan

September 2019, 74 pages

Radar receivers collect mixed signals from all electromagnetic sources in the environment. The ultimate goal of electronic intelligence is to find the types of these sources with the help of a priori information, known as emitter identification. Emitter identification system aims to find a representative for each emitter in the environment and update them over time. Hence, such a non-stationary and continuous flow of data is of this thesis concern which is beyond the scope of traditional –offline or batch- machine learning systems. Another challenge is that the system can not know all possible emitter types and does not have a priori knowledge about the number of emitters. Therefore, incremental or online learning methods should be considered for the update of emitter representatives. After obtaining a representative for each emitter in a typical incremental learning algorithm, these representatives should be compared with a list of previously available emitter types. This part requires symbolic data analysis since the radar parameters generally operate interval-based. During simulations, among incremental learning algorithms, fuzzy ART, Bayesian ART, SOM and KDESOINN are examined and several extensions are proposed for the selected online learning networks. An ART-based structure based on Jaccard index is also proposed and tested for symbolic classification. The results indicate that the proposed symbolic data analysis method has outperformed other distance metrics and that the proposed algorithmic extensions enhance the performance of the selected online learning algorithms, while KDESOINN is observed to perform the best in terms of accuracy.

Keywords: incremental/online learning, symbolic data analysis, emitter identification

ÖZ

KADEMELİ ÖĞRENME VE SEMBOLİK GÖSTERİMLER İLE RADAR KİMLİKLENDİRME

Erol, Aybüke
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan

Eylül 2019 , 74 sayfa

Radar alıcıları ortamdaki tüm elektromanyetik kaynaklardan karışık halde sinyaller alır. Elektronik harp sistemlerinin kilit noktası olan kimliklendirme, bu kaynakları birer temsilci ile göstermeyi; bu temsilcileri zaman içinde güncellemeyi ve sonuç olarak kaynakların tiplerini önsel bilgiler yardımıyla bulmayı hedefler. Dolayısıyla bu tezin kapsamında durağan olmayan ve sürekli bir veri akışı söz konusudur; ki bu veri yapısı kavram geleneksel –çevrimdışı ya da toplu- makine öğrenme yontemlerinin uygulama alanının dışında kalmaktadır. Problemin başka bir zorluğu ise kurulacak sistemin olası tüm radar tiplerini bilmesinin mümkün olmaması ve sınıf sayısının belirsiz olmasıdır. Bu sebeple, radar temsilcilerini zamanla birlikte güncelleyebilmek için kademeli (çevrimiçi) oğrenme sistemleri kullanılmalıdır. Tipik bir kademeli öğrenme yöntemi ile her bir kaynağın temsilcisi elde edildikten sonra bu temsilciler önceden var olan radar türleri ile kıyaslanmalıdır. Radar parametreleri genellikle aralık tabanlı olduğu için bu kısımda simgesel veri işleme yontemlerine başvurulmuştur. Simülasyonlar sırasında kademeli öğrenme yöntemleri arasından fuzzy ART, Bayesian ART, SOM ve KDESOINN incelenmiş ve bu yapılara yöntemsel ilave(ler) önerilmiştir. Sembolik sınıflandırma kısmı için ise Jaccard ölçeğine bağlı ART tabanlı bir yapı önerilmiştir. Sonuçlar önerilen sembolik veri işleme yönteminin diğer mesafe ölçeklerini geçtiğini, önerilen ilavelerin seçilen kademeli öğrenme yöntemlerini iyileştirdiğini ve bunlar arasından doğruluk açısından en iyi performansın KDESOINN ile elde edildiğini göstermiştir.

Anahtar Kelimeler: kademeli/çevrim içi öğrenme, sembolik veri analizi, radar kimliklendirme

To Ayten and Ömer Faruk Erol

ACKNOWLEDGMENTS

I would like to start by expressing my sincere gratitude to my supervisor, Prof. Dr. Aydın Alatan, for giving me this opportunity, for knowing me and understanding me. Prof. Dr. Aydın Alatan, besides his indisputable engineering skills, is an excellent reader of human character. I've seen him treat differently yet fair to his every student during the years I've studied with him. He always told me when I was wrong, while never failing to encourage me. For the state of mind that I am in now, I will forever be grateful.

I've been a part of METU Center for Image Analysis (OGAM) during my studies where every person helped and supported me. Most importantly, they've become my friends. Foremost, I can not count all the times Aziz Berkay Yeşilyurt saved me, in every way that a Master's student can be saved. Seeing how ambitious he is in everything that he does, how happy it makes him to learn made me love what I do even more. He changed me, in all the better ways. Then I would like to thank Oğul Can, for always listening to me, whether I was asking a question or I was sad or the worst, trying to be funny. He always told me the truth and helped me grow up. İzlen Geneci was an angel to me and she still is. I wouldn't believe that I could share so much with someone in such little time. She helped me accept the things that I can not change about myself as the way they are. I thank Ece Selin Böncü and Yeti Ziya Gürbüz for guiding me literally every step of the way. They were quite ahead of me, but they never got tired of looking back for me to support me. I would like to express my gratitude to Dr. Alper Koz, who patiently answered all my questions and concerns and made me very happy with all the food he so kindly brings to OGAM every time he visits another city. Of course, OGAM would not be the same for me if Ayberk Aydın, Esat Kalfaoğlu and Alp Eren Sarı were not there, whom I had so many great memories with.

There are two people in my life that I have to thank for my every accomplishment in life, my mother and father. I thank my father for pushing me to experience life by myself, take risks and sometimes even fail, instead of just protecting me. I also thank him for showing me, instead of telling me what to do. I thank my mother for the special bond we have. She is never scared to criticize me when she knows that I can be better, yet I only feel completely not judged with her. She taught me how to study and discipline myself when I was young, for which I am still being rewarded today.

Last but not least, I would like to thank ASELSAN for funding my thesis project and for our weekly meetings to help my progress.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xii

LIST OF TABLES xv

LIST OF FIGURES xvi

LIST OF ABBREVIATIONS xx

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem Definition 1

1.2 Types of Learning 3

1.3 Scope of the Thesis 4

1.4 Contributions and Novelties 4

1.5 Outline of the Thesis 5

2 BACKGROUND INFORMATION 7

2.1 Introduction 7

2.2 Performance Metrics 10

2.2.1 Cluster Purity Metric 11

2.2.2 Ground Truth Purity Metric 12
2.2.3 Proposed Interval Match Score Metric 13
3 INCREMENTAL LEARNING AND SYMBOLIC DATA ANALYSIS . . . 15
3.1 Adaptive Resonance Theory 17
3.1.1 Fuzzy ART 21
3.1.1.1 Components of Fuzzy ART 22
3.1.2 Bayesian ART 23
3.2 Self-Organizing Map 26
3.2.1 Self-Organizing Incremental Neural Networks 28
3.2.2 KDESOINN 31
3.3 Incremental Learning Literature for Emitter Identification 35
3.4 Symbolic Data Analysis 39
3.4.1 Hausdorff Distance 40
3.4.2 Wasserstein Distance 40
3.4.3 Jaccard Distance 41
3.5 Symbolic Data Analysis Literature for Emitter Identification 41
4 PROPOSED EMITTER IDENTIFICATION METHODS 45
4.1 Proposed Extensions on Fuzzy ART 46
4.1.1 Normalization 46
4.1.2 Activation Function and Vigilance Test 47
4.2 Proposed Extension on Bayesian ART 48
4.3 Clustering Stage of Self-Organizing Map 49
4.4 Proposed Extensions on KDESOINN 52

4.4.1 Distance Metric 52

4.4.2 Covariance Calculation 52

4.5 Proposed SDA Method 53

4.6 Tests and Simulations 56

4.6.1 Results on Incremental Learning 56

4.6.2 Results on SDA 59

4.7 Conclusions 60

5 CONCLUSIONS 63

REFERENCES 65

APPENDICES

A SIMULATOR 73

LIST OF TABLES

TABLES

Table 3.1 An example symbolic data set [1] 39

Table 4.1 The threshold parameters for each algorithm. 58

Table 4.2 SDA results with respect to various distance metrics. 61

LIST OF FIGURES

FIGURES

Figure 1.1 Deinterleaving and identification processes shown at two different time instants ($t = T_1$ and $t = T_2$). As new radar pulses are received, deinterleaving is called to divide the mixed pulses into separate pulse groups in a way that all the pulses within a group come from the same emitter. Identification takes these pulse groups as input and identifies them with the help of a priori information. Different pulse groups at different time instants may or may not be from the same emitter. 2

Figure 1.2 Block diagram for proposed emitter identification. The dashed box shows the scope of this thesis. 3

Figure 2.1 Received signal strength (PA) vs. time (TOA) of a circular scan [2]. 9

Figure 2.2 PRI modulation types. Top left is constant, top middle is jittered, top right is constant staggered, bottom left is random staggered and bottom right is dwell and switch PRI. 10

Figure 2.3 A toy example to illustrate cluster purity and ground truth purity. The first cluster has a cluster purity of 3/5 whereas the second and third clusters have a cluster purity of 1. The two ground truth classes, triangle and square, have a ground truth purity of 1 whereas the final ground truth class, star, has a ground truth purity of 3/5. 12

Figure 3.1 Incremental learning [3]. 15

Figure 3.2 An unsupervised competitive learning example. The black points show the input samples and are provided sequentially to the network. The red points are randomly initialized cluster centers and every time an input is presented, the closest one of them to the input is determined and updated towards the input. The trajectory of the cluster centers are shown with red arrows [4]. 18

Figure 3.3 ART1 structure. The input is binary and the network consists of two layers, F_1 being the input layer, F_2 being the output layer. There exists two sets of weights, $\boldsymbol{w}$ and $\boldsymbol{t}$. The input is projected to F_2 through the $\boldsymbol{w}$ weights to find the winner prototype. Then, through $\boldsymbol{t}$ weights, an expectation of the input is formed with the winner. After comparing the expectation with the input itself, if the resemblance is not enough with respect to a threshold ρ, the current winner is reset and another winner is searched. 19

Figure 3.4 The flowchart of fuzzy ART. 23

Figure 3.5 An example 6x4 SOM grid. Each node on the grid is attached to each input dimension through its weight vector. 26

Figure 3.6 An example SOM grid obtained with three Gaussian distributions having different means. Each color (white, gray and black) shows a cluster of the input data. The squares separated with grid lines are the SOM nodes and each node is associated with a weight vector that is of the same size as the input data. Close nodes on the grid represent the same cluster, i.e. carry the same color. 28

Figure 3.7 Threshold regions of SOINN (left) and KDESOINN (right) given the same input (ξ), winners (s_1 and s_2) and graph. SOINN constructs a hypersphere with radii Θ_{s_1} and Θ_{s_2}. The radii are calculated as the distance from the winner itself to its farthest neighbor as given in Equation 3.29. As it can be seen, the input is inside the regions determined by SOINN. The threshold regions in KDESOINN (calculated using kernel density estimation as in Equation 3.34), on the other hand, do not include the input [5]. 32

Figure 3.8 Clustering with KDESOINN. Each connected graph (set of nodes) represents a Gaussian distribution with a different mean. 35

Figure 3.9 Feature transfer in a progressive neural network of three columns. The first two columns on the left, with dashed arrows, are trained on first and second tasks respectively. For the final task, a third column is added which is given access to all previously learned features [6]. 37

Figure 3.10 Construction of the list of active emitters. 43

Figure 4.1 Fuzzy ART with proposed normalization compared with the original Fuzzy ART [7]. Vigilance parameter is swept. 46

Figure 4.2 The flowchart of improved fuzzy ART. 48

Figure 4.3 Proposed fuzzy ART (with both extensions) compared with the original Fuzzy ART [7]. Vigilance parameter is swept. 49

Figure 4.4 Proposed Bayesian ART compared with the original Bayesian ART [8]. S_{MAX} and c is swept for the original and proposed implementation of Bayesian ART respectively. 50

Figure 4.5 An example dendrogram. Each point starts as a single cluster and connections between them are made starting from the nearest ones until there remains only one cluster. 51

Figure 4.6 KDESOINN with cosine distance compared with the original KDESOINN [5] (Euclidean distance). k (of k-NN) is swept. 53

Figure 4.7 Proposed KDESOINN (with both extensions) compared with the original KDESOINN [5]. k (of k-NN) is swept. 54

Figure 4.8 Flowchart of the proposed SDA method. 57

Figure 4.9 Block diagram for simulator data pre-processing. The simulator data is divided into consecutive (with respect to TOA) blocks of the same pulse length. The next block in line is deinterleaved using the ground truth (GT) labels to obtain sPDWs. The sPDWs are then subject to feature extraction to obtain the input signal x for the incremental learner. 58

Figure 4.10 Results on simulator data set. (Note that these algorithms are based on the extensions proposed in this thesis.) 59

Figure 4.11 Results on external data set. (Note that these algorithms are based on the extensions proposed in this thesis.) 60

LIST OF ABBREVIATIONS

ART	Adaptive Resonance Theory
ASP	Antenna Scan Period
AST	Antenna Scan Type
BMU	Best-Matching Unit
CP	Cluster Purity
DOA	Direction of Arrival
ELINT	Electronic Intelligence
EW	Electronic Warfare
GNG	Growing Neural Gas
GP	Ground Truth Purity
GT	Ground Truth
PDW	Pulse Descriptor Word
IMQ	Interval Match Quality
k-NN	k-Nearest Neighbor
KDE	Kernel Density Estimation
KDESOINN	SOINN with Kernel Density Estimation
PA	Pulse Amplitude
PRI	Pulse Repetition Interval
PW	Pulse Width
RF	Radio Frequency
SDA	Symbolic Data Analysis
SOINN	Self-Organizing Incremental Neural Network
SOM	Self-Organizing Map
TOA	Time of Arrival

CHAPTER 1

INTRODUCTION

Radar is a system that uses radio waves for sensing remote objects. Electronic intelligence (ELINT) is intelligence-gathering from the signals transmitted by radar systems to obtain information about their capabilities [9].

Radar receivers collect signals from all electromagnetic sources in an electronic warfare (EW) environment. Some of these sources might be hostile whereas others might be neutral/irrelevant, friendly or unknown. The identification of these sources, called as emitter identification, is the ultimate goal of ELINT systems. If the identity of a source is known, it is possible to comment on its mission and operation, both of which carry critical information for the strategy onward [10].

1.1 Motivation and Problem Definition

Based on several radar pulse parameters, the type of an emitter can be recognized with the help of a priori knowledge. However, emitter identification first requires sources to be separated from one another [11]. This process, known as deinterleaving, divides the received mixed pulse sequence into groups of pulses and it is assumed that all of the pulses inside any one of these pulse groups belong to the same emitter. In other words, when the output of deinterleaving at a given time instant is considered, different pulse groups correspond to different emitters. However, two pulse groups obtained with deinterleaving at two different time instants may still correspond to the same emitter. Hence, it is the responsibility of emitter identification to find a representation for all pulse groups at any instant after deinterleaving is performed, compare them with previously existing representations or previously available emitter types and update these representations over time as deinterleaving provides new pulse groups [12]. A diagram that summarizes this cycle is provided in Figure 1.1.

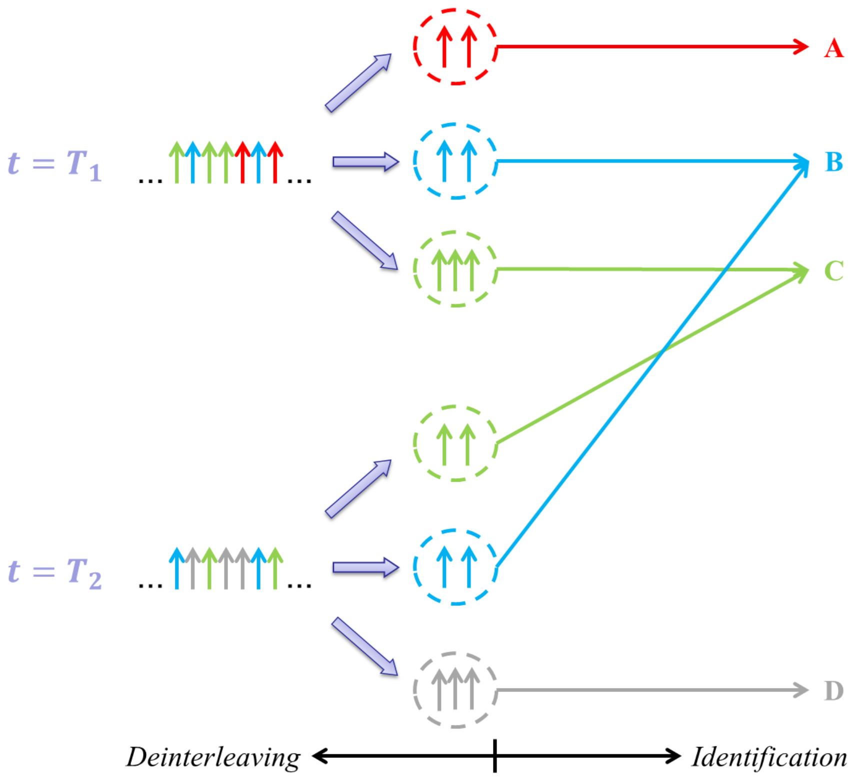


Figure 1.1: Deinterleaving and identification processes shown at two different time instants ($t = T_1$ and $t = T_2$). As new radar pulses are received, deinterleaving is called to divide the mixed pulses into separate pulse groups in a way that all the pulses within a group come from the same emitter. Identification takes these pulse groups as input and identifies them with the help of a priori information. Different pulse groups at different time instants may or may not be from the same emitter.

The proposed emitter identification system is provided in Figure 1.2. The radar pulses are received in the form of Pulse Descriptor Words (PDWs). A PDW consists of radio frequency, pulse width, time of arrival, pulse amplitude and direction of arrival information (Section 2.1). The PDWs are grouped into super-PDWs (sPDWs) by the deinterleaver. The sPDWs are subject to feature extraction in order to obtain a representative vector x from each sPDW that contains the descriptive information. Next is the incremental learning stage that updates, or when necessary unites the representatives over time. The output of this stage is a list of unnamed emitters. To be compared with a priori known emitter types and be labelled, first a symbolic representation of these emitters are obtained. This is necessary because radar parameters generally operate inside a range rather than at a single point, thus their assignment to a type is considered under symbolic data analysis. In the last part, prior information of known emitter types and their parameters is utilized to identify the emitters and finally demonstrate a list of active emitters in the environment.

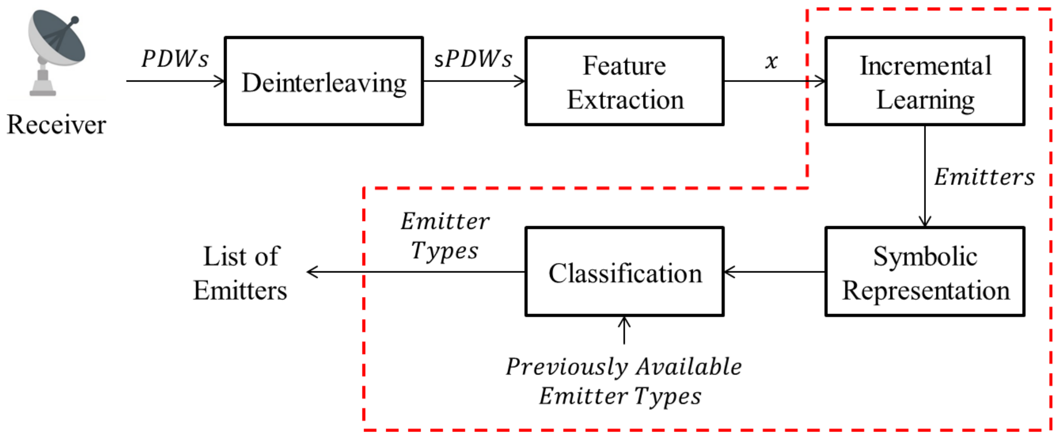


Figure 1.2: Block diagram for proposed emitter identification. The dashed box shows the scope of this thesis.

All in all, the problem of emitter identification can be regarded as an online classification problem with adaptive number of classes. The number of classes can not be taken as a constant number due to unknown emitter types. Currently, a system can not know all possible radar emitter types, and the number of emitter types keep increasing dramatically [13]. The constructed system should be able to extend its network to accommodate new classes and learn them since it is inevitable that unfamiliar emitter types are to be encountered during the electronic battle.

1.2 Types of Learning

The learning algorithms are of two types: offline (batch) learning and online (incremental) learning [14]. In offline learning, which is the traditional approach, there is access to the full data set, which is generally divided into training, validation and test sets [15]. A model is constructed and fixed using the training and validation sets before the test set is presented to the network. Therefore, a static structure is assumed among the input data set in batch learning. On the other hand, incremental learning refers to a continuous learning process based on non-stationary streaming data [16].

Emitter identification system will be provided pulse groups continuously from the deinterleaver. It will learn and update the representatives of these pulse groups over time, identify each representative by either assigning it to an already known emitter type or initializing a new type for it. Hence, continual processing of pulse groups is required, which makes emitter identification an incremental learning problem. However, not all incremental learning algorithms allow an expandable structure, i. e. suitable for a change in the number of classes. This thesis focuses on online learning networks that can as well increase the number of classes whenever unfamiliar data is observed and this concept will be examined in detail in Chapter 3.

1.3 Scope of the Thesis

Emitter identification is the final step of ELINT and comes after a deinterleaver. In this thesis, only the problem of emitter identification is considered. During the experiments, the PDWs are grouped according to ground truth labels as if there is a deinterleaver. Both the literature survey and experimental results are only dedicated to emitter identification.

1.4 Contributions and Novelties

The contributions of this thesis are as follows:

- In order to decrease the complexity of fuzzy ART, the vigilance test and activation function is combined using Jaccard index, as explained in Section 4.1.

- The vigilance test in Bayesian ART is replaced by an input-representative covariance based test in order to account for the effect of the input on the representative, as described in Section 4.2.

- A monotonically decreasing ρ parameter is suggested in KDESOINN while determining the threshold regions as the more samples are accumulated, the less smoothing is required, as discussed in Section 4.4.
- A new method for growing classification of interval based data is proposed in Section 4.5.
- A new metric called the interval match score is proposed to compare two different interval based lists in Section 2.2.3.

1.5 Outline of the Thesis

This thesis focuses on emitter identification. The rest of the thesis is organized as follows:

In Chapter 2, the parameters that constitute a radar pulse are defined. In addition, possible operation types of these radar parameters are explained, which is necessary to construct a realistic test platform. Last but not least, the performance metrics to evaluate the performance of emitter identification are introduced.

In Chapter 3, incremental learning and symbolic data analysis methods are investigated. After introducing the general concept of each, among all the methods, fuzzy ART, Bayesian ART, SOM, KDESOINN and Jaccard index based interval analysis are found to be the most convenient to emitter identification. The detailed explanation of each of these algorithms is provided.

In Chapter 4, several extensions for the selected algorithms in Chapter 3 are proposed. The advantages of these extensions are supported through experimental results and finally, all of the algorithms are compared both on simulator data and an external data set.

The thesis is concluded with final remarks and possible future works in Chapter 5.

CHAPTER 2

BACKGROUND INFORMATION

2.1 Introduction

Radar, short for Radio Detection and Ranging, is a system in which radio waves are used to obtain information about a target object such as its velocity or position. A radar pulse is received in the form of a Pulse Descriptor Word (PDW). The deinterleaver takes PDWs as input and groups them, forming super-PDWs (sPDWs) in such a way that the PDWs under an sPDW are assumed to belong to the same emitter.

A PDW may contain all or some of the following parameters [17]:

- Radio Frequency (RF)

  Radio waves carry a frequency in the 30 Hz - 300 GHz band of the electromagnetic spectrum. Due to the fact that physically close radars can not be operated on the same frequency, RF can be quite distinctive especially for emitters that are near to each other [18].

  In order to decrease the chance of being detected, radars nowadays generally change their frequency. The change in frequency can be from pulse to pulse, known as frequency agile, or from group to group, known as frequency hopping. RF can also be in multiple mode, which corresponds to the transmission of two different RF values at the same time [19].

- Pulse Width (PW)

  The time duration that a radar transmitter emits energy is called the pulse width, also known as the pulse duration. Although various modes are possible for PW as well (constant, jittered, dwell and switch), many radars operate at similar PWs. Most radars use PWs between .25 µs to 5 µs [20].

- Time of Arrival (TOA)

  TOA corresponds to the time at which a pulse is received.

- Pulse Amplitude (PA)

  The received pulses carry an amplitude that is dependent on the distance between the receiver and transmitter antennas and the time of the pulse. According to antenna scan type (AST), transmitter antennas follow different scan patterns and if the pulses are received during the main lobe of radar, PA will be much higher compared to side and back lobes. PA is also affected by the respective motion of the transmitter and receiver antennas [21].

  Although the TOA-PA sequence is essential for the estimation of AST and antenna scan period (ASP), PA will not be reliable for emitter identification by itself because it is highly influenced by some parameters as mentioned above, which do not define the characteristics of an emitter.

- Direction of Arrival (DOA)

  DOA, also referred to as angle of arrival, takes a value between 0 and 360 and shows the direction in which pulses are received. Even if the transmitter is in motion with respect to the receiver, the change in DOA between two measurements will not be significant as PRI is in the order of microseconds. Hence, DOA gives a more reliable and certain information compared to other parameters.

These parameters, together with Pulse Repetition Interval (PRI), characterize emitter types. PRI of an emitter is not a directly measured parameter, yet it is simply obtained by taking the difference of the TOA values of consecutive pulses from that emitter.

PA and TOA information together are also used to determine AST and ASP. For example, Fig. 2.1 shows a typical TOA-PA pattern for a circular scan whose periodicity is quite easy to observe.

There are several PRI modulation types; they are listed as follows [22]:

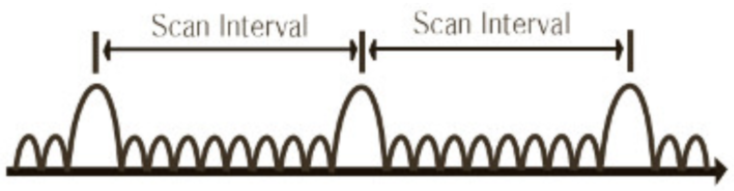


Figure 2.1: Received signal strength (PA) vs. time (TOA) of a circular scan [2].

- Constant

  Constant PRI is the simplest case and it means that the emitter sends consecutive pulses with a constant time gap.

- Jittered

  When PRI is jittered, the time gap between consecutive pulses is around a center PRI value, but may vary from it up to a certain range. In other words, jittered PRI can be considered as having a uniform distribution inside a PRI interval.

- Constant/Random Staggered

  In staggered modulation, there exists several options for a PRI value, called the stagger levels. The stagger levels can also be jittered. If modulation type is random staggered, PRI of each pulse is selected randomly among the stagger levels. If modulation type is constant staggered, the time gap between consecutive pulses follows a constant pattern between the stagger levels. The number of levels in both cases is random.

- Dwell and Switch

  If modulation type is dwell and switch, like in staggered modulation, there exists several PRI levels. However, the PRI repeats itself for a few pulses at each level. In other words, a set of consecutive pulses are separated at a constant time gap value, then this value is changed for the next set of consecutive pulses. The number of levels and the number of pulses at each level are random parameters and the levels can carry a certain amount of jitter.

PRI patterns for all of the modulation types are given in Figure 2.2.

A test platform has been constructed to produce pulses with RF, PW, DOA and TOA

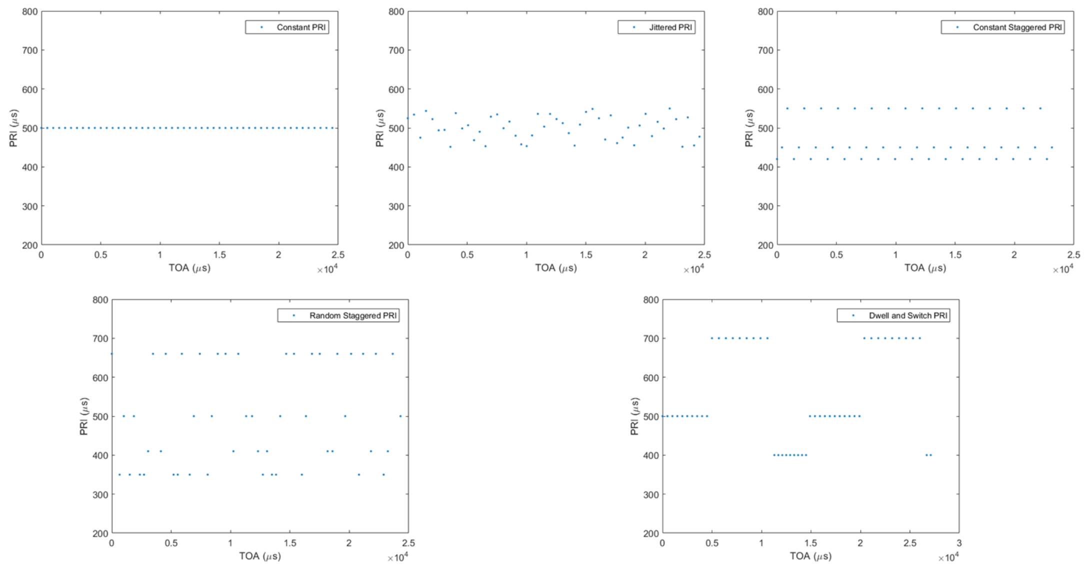


Figure 2.2: PRI modulation types. Top left is constant, top middle is jittered, top right is constant staggered, bottom left is random staggered and bottom right is dwell and switch PRI.

information in accordance with the modulation types described above. The details of the simulator can be found in Appendix A.

2.2 Performance Metrics

Two metrics are utilized to evaluate clustering. The first one is cluster purity, defining how pure a cluster is. If a cluster contains samples from only one class, it has a cluster purity of 1. The second one is ground truth purity, standing for how pure a ground truth class is clustered. If all of the samples that belong to a class are under the same cluster, that class has a ground truth purity of 1.

Instead of using one metric to evaluate the performance of clustering (such as adjusted Rand index, [23]) using these two metrics allows the user to comment on why or how clustering has failed as well. For instance, less ground truth purity means there are much more clusters than there are classes. Less cluster purity means misclassification took place. What's more, growth is controlled by a threshold parameter in all of the considered algorithms, which are explained in detail in Chapter 3 and 4. The effect of the related parameter on a selected algorithm is much easier to comment on with the use of two metrics for evaluation. For instance, generally, if the search for similarity has become too strict (meaning that the threshold parameter is too high), cluster purity increases whereas ground truth purity decreases.

To evaluate the performance of symbolic data analysis, a comparison of lists which have interval valued elements and which may have different sizes is required. For this purpose, a metric called as "interval match quality" is proposed.

2.2.1 Cluster Purity Metric

Cluster purity describes how pure a cluster is in terms of the class samples it contains. Let the clustering state be $\Omega = v_1, v_2, \dots v_k$ where k is the number of clusters. Cluster purity for the jth cluster is defined as the ratio of the most prevalent class in the cluster [24]:

$$CP(v_j) = \frac{1}{N_{v_j}}(\max_i |v_j \cap c_i|) \tag{2.1}$$

where N_{v_j} is the number of samples that belong to v_j, c_i shows ith ground truth class and $|.|$ gives the cardinality.

The overall (final) cluster purity is computed by dividing the sum of cluster purity of all clusters by the number of clusters:

$$CP_f = \frac{1}{k}\sum_{j=1}^{k} CP(v_j) \tag{2.2}$$

For example, in Figure 2.3, the first cluster is not completely pure as it contains samples from more than one class. To compute its cluster purity, the maximum number of samples that belong to the same class under this cluster is divided by the total number of samples under this cluster, which gives 0.6. The second and third clusters both contain only one class, hence are completely pure. The final cluster purity is equal to $(1 + 1 + 0.6)/3 = 0.867$.

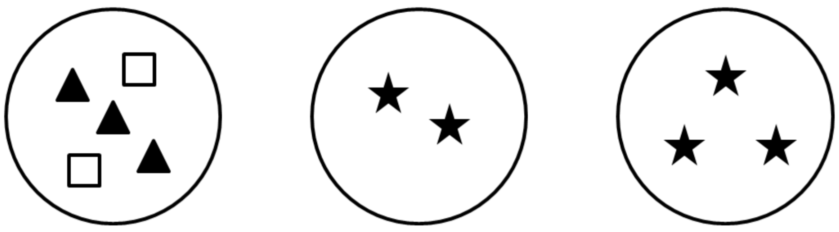


Figure 2.3: A toy example to illustrate cluster purity and ground truth purity. The first cluster has a cluster purity of 3/5 whereas the second and third clusters have a cluster purity of 1. The two ground truth classes, triangle and square, have a ground truth purity of 1 whereas the final ground truth class, star, has a ground truth purity of 3/5.

2.2.2 Ground Truth Purity Metric

Ground truth purity describes how pure a ground truth class is divided into separate clusters [25]. If all the samples of a ground truth class belongs to the same cluster, its purity is 100%. Ground truth purity for the ith class is defined as the ratio of the greatest number of its samples that belong to a single cluster:

$$GP(c_i) = \frac{1}{N_{c_i}}(\max_j |v_j \cap c_i|) \tag{2.3}$$

where N_{c_i} is the number of samples that belong to c_i.

The overall (final) ground truth purity is computed by dividing the sum of ground truth purity of all classes by the number of classes:

$$GP_f = \frac{1}{C}\sum_{i=1}^{C} GP(c_i) \tag{2.4}$$

where $i = 1, 2, ...C$ stands for the ground truth classes, with C being the total number of classes.

Going back to Figure 2.3, the class of triangles and the class of squares are both inside one cluster only, thus have 100% ground truth purity. However, the class of stars is divided into two clusters by a ratio of 2 to 3. Hence, its ground truth purity is 0.6. The final ground truth purity is equal to $(1 + 1 + 0.6)/3 = 0.867$.

The ideal case is when both cluster purity and ground truth purity is 1.

2.2.3 Proposed Interval Match Score Metric

Previous metrics consider single numeric values, as most of the literature does. However, the final output of emitter identification is a list of active emitters, whose parameters are shown in intervals. In this thesis, for the cases where ground truths are interval based lists, a new metric is proposed.

The actual list of active emitters might be of different size than the found (estimated) one. Moreover, there might be some shifts in the ranges of radars due to misclassified samples. This, of course, should be punished.

Let the true list of active emitters be $\mathcal{T}$ and the estimated list be $\mathcal{E}$. Both $\mathcal{T}$ and $\mathcal{E}$ consists of interval elements: $t_i, i = 1, 2, \dots M$ and $e_j, j = 1, 2, \dots N$ respectively, where M is the size of $\mathcal{T}$ and N is the size of $\mathcal{E}$. The interval match score (IMQ) is calculated as follows:

Algorithm 1 Calculation of IMQ

```
1: Initialize the score of each $t_i$ as zero: $s = 0, \dots, 0$
2: for each $e_j, j = 1, 2, \dots N$ do
3:     Find the closest true interval and its distance to it: $I = \arg\min_i d(t_i, e_j)$, $d_{min} = d(t_I, e_j)$
4:     Set $s(I) = 1 - d_{min}$
5: IMQ = $\frac{1}{M} \sum_{i=1}^{M} s(i)$
```

The distance measure, d is selected as Jaccard distance, which is explained in detail in Section 3.4.3. Basically, it is intersection over union subtracted from 1.

To explain this metric over an example, let the true list be {[2,6], [6,9]} and the estimated list be equal to {[2,9]}. The algorithm erroneously merged these two radars. The accuracy of the two elements of the true list are initialized as 0 ($s(1) = s(2) = 0$). The only element of the estimated list is considered and its closest match among the elements of the true list is obtained. The Jaccard distance between [2,9] and [2,6] is 3/7 whereas it is 4/7 between [2,9] and [6,9]. Hence, [2,6] is the closest match. Then,

$s(1) = 1 - 3/7 = 4/7$ and $s(2)$ is still 0, since the true list has already been assigned to a result. Hence, IMQ for this example is calculated as $2/7$.

CHAPTER 3

INCREMENTAL LEARNING AND SYMBOLIC DATA ANALYSIS

Computational systems in practice are subject to continuous streams of information and are required to learn from a data-flow, which is called as incremental learning [26] (Figure 3.1). In the machine learning literature, the term incremental learning is also referred to as online learning, lifelong learning, continual learning or evolutionary learning to emphasize learning of new information as input data becomes available over time [27] [28].

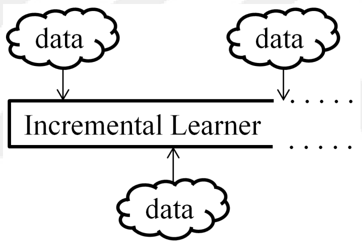


Figure 3.1: Incremental learning [3].

In the recent years, besides the need for learning from a real-time data-stream, incremental learning has also gained popularity with big data processing [29]. The traditional (batch or offline) machine learning approach is subject to a trade-off between time and accuracy, that is to obtain better accuracy, more data needs to be processed. However, with incremental learning, fast classification with reasonable accuracy and high speed is possible [30].

When a static relationship between input and output variables is assumed in a non-stationary environment, a model may perform much worse as data changes over time. This brings with the most common problem in incremental learning tasks or learning from non-stationary distributions in general, catastrophic forgetting or stability-plasticity dilemma [31]. Both refer to networks forgetting all a priori information in order to adapt to environmental changes. In life-long learning tasks, learning is supposed to continue as long as new data is available which may result in the network losing its a priori knowledge over time. However, a network should be stable, meaning it should preserve existing knowledge. On the other hand, too much stability might interfere with plasticity, which is the ability of a system to keep up with the current environment.

Growth is a crucial feature for these types of tasks since it allows adapting to changing environments while preserving old input patterns. Therefore, insertion of new classes is a very useful contribution to the stability-plasticity dilemma [32]. In addition, emitter identification itself requires a change in the total number of classes due to unknown emitter types. Consequently, the literature survey on incremental learning methods in this thesis only focuses on the ones allowing network growth.

As an another solution to the stability-plasticity dilemma, instead of updating the neurons in the network altogether with each data, only the most similar neuron(s) to the input should be updated. In other words, classes or neurons should be in competition with each other to win the right to learn the input. This way, new information can not interfere with all neuron weights, it can only update the most relevant one(s), which also helps them stay up-to-date. This type of learning is known as competitive learning.

As a completely different topic analyzed in this chapter, symbolic data analysis (SDA) is an extension of standard data analysis to be applied on lists, intervals, multi-valued variables, distributions etc. [33]. For emitter identification, a significant challenge is that describing an emitter type by single numeric values would not be fair as radar features are generally interval based. For example, emitters today do not operate on a single frequency, they rather have a frequency range in which they can operate. Hence, the representation and comparison of previously available emitter types and radar clusters are considered under symbolic data analysis (SDA).

In the upcoming sections, the fundamental algorithms related to incremental learning are examined.

3.1 Adaptive Resonance Theory

Adaptive resonance theory (ART) is an unsupervised neural architecture based on competitive learning. Competitive learning is a variant of Hebbian learning. Hebbian rule states that the weight of the connection between two neurons (input and output) should be changed in proportion to the product of the activation of the units [34]:

$$\Delta \boldsymbol{w}_i = \beta \boldsymbol{x} \boldsymbol{w}_i \tag{3.1}$$

where η is the learning rate, $\boldsymbol{x}$ is the input and $\boldsymbol{w}_i$ is the ith weight vector.

Competitive learning is a type of learning where output neurons compete with each other to learn the current input. Whenever an input sample is presented to the network, its closest match among the output neurons is selected as the winner. Unlike multilayer perceptrons where all neurons are updated, only the winner neuron stays active for learning, which is known as winner-takes-all principle [35].

The use of competitive learning is also convenient to emitter identification in the sense that a new class should be initialized in case that the input signal does not resemble any of the existing classes. The resemblance between the class that is most similar to the input signal (or for some algorithms, the two most similar classes) and the input signal itself creates a measure to determine whether a new class should be initialized or not.

Figure 3.2 shows an unsupervised example of competitive learning. There are three randomly initialized cluster centers, shown as red points. The input samples, shown as black points, are presented to the network sequentially. Each time an input sample is given, its nearest cluster center is selected as the winner. Then, the location of the winner is updated towards the input sample. The red arrows show the trajectory of the cluster centers.

ART networks are first proposed as a solution to the stability-plasticity dilemma. The first version, ART1, is used to cluster binary input samples. The ART1 structure is provided in Figure 3.3.

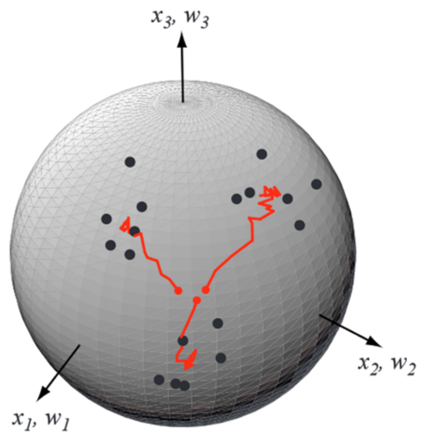


Figure 3.2: An unsupervised competitive learning example. The black points show the input samples and are provided sequentially to the network. The red points are randomly initialized cluster centers and every time an input is presented, the closest one of them to the input is determined and updated towards the input. The trajectory of the cluster centers are shown with red arrows [4].

The number of nodes in F_1 is equal to the dimension of the input and each node stands for a dimension. The nodes in F_2 are the categories, i.e. they each represent a cluster of the input data. Every node i in F_1 is connected to every node j in F_2 through bottom-up ($\boldsymbol{w}$) and top-down ($\boldsymbol{t}$) weights. When an input is presented to the network, each category j in F_2 is activated at a level according to how similar it is to the input. The activation calculation is as follows:

$$net_j^{F_2} = \sum_i w_{ji} x_i =< \boldsymbol{w}_j, \boldsymbol{x} >= ||\boldsymbol{w}_j \cap \boldsymbol{x}||_1 \tag{3.2}$$

where $||.||_1$ is the L_1-norm, $\cap$ is the binary AND operation and $< .,. >$ is the inner product. After the activation of each node j in F_2, the winner category J is determined as:

$$J = \arg\max_j net_j^{F_2} \tag{3.3}$$

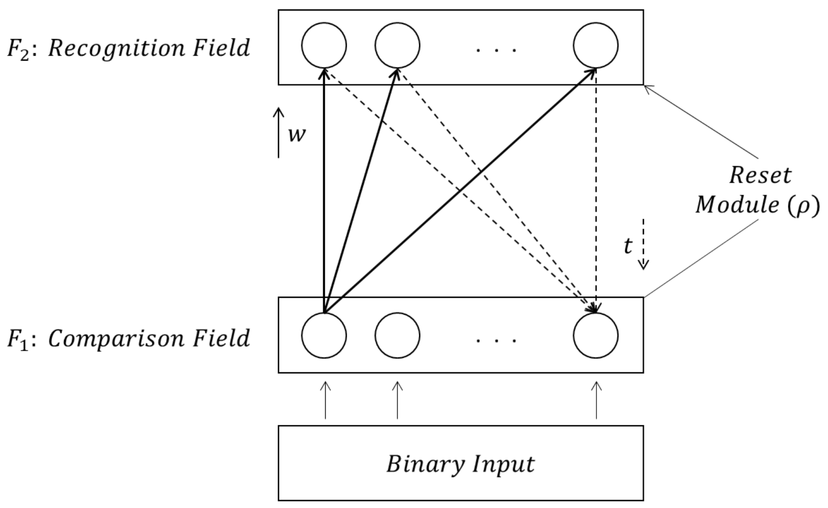


Figure 3.3: ART1 structure. The input is binary and the network consists of two layers, F_1 being the input layer, F_2 being the output layer. There exists two sets of weights, $\boldsymbol{w}$ and $\boldsymbol{t}$. The input is projected to F_2 through the $\boldsymbol{w}$ weights to find the winner prototype. Then, through $\boldsymbol{t}$ weights, an expectation of the input is formed with the winner. After comparing the expectation with the input itself, if the resemblance is not enough with respect to a threshold ρ, the current winner is reset and another winner is searched.

An output vector, $\boldsymbol{o}_J$ is constructed as follows:

$$\boldsymbol{o}_J = \begin{cases} 1, \text{if } j = J \\ 0, \text{otherwise} \end{cases} \tag{3.4}$$

$$net_i^{F_1} = t_{iJ}\boldsymbol{o}_J \tag{3.5}$$

This, after written for all dimensions i, forms the expectation. Finally, vigilance test is performed to decide if the match between the expectation and input is adequate:

$$\frac{\sum_i net_i^{F_1}}{||\boldsymbol{x}||_1} > \rho \tag{3.6}$$

If the test is passed, learning is initiated for the winner prototype. Otherwise, the current winner is reset and the next most activated prototype is selected as the new winner J and subject to Equations 3.4-3.6. If a prototype that can pass the vigilance test is not found, a new prototype is committed [36].

Learning is different for bottom-up ($\boldsymbol{w}$) and top-down ($\boldsymbol{t}$) weights [37]:

$$w_{Ji} = \frac{Lx_i}{L - 1 + ||\boldsymbol{x}||_1} \tag{3.7}$$

$$t_{iJ} = x_i \tag{3.8}$$

This way, the current or most updated version of the input cluster is stored under $\boldsymbol{t}_J$, whereas $\boldsymbol{w}_J$ is only updated by the input with a learning rate of L. As a result, top-down weights acts as a short term memory and bottom-up weights act as a long term memory for the input clusters they represent. Following this, $\boldsymbol{t}$ ensures plasticity while $\boldsymbol{w}$ ensures stability.

There are various structures derived from ART, both for supervised and unsupervised learning. Yet, they all follow a similar pattern, which is provided in Algorithm 2.

Algorithm 2 General ART Structure

```
1:  for each $x \in \boldsymbol{x}$ do
2:      Calculate activations $T_j = f(\boldsymbol{x}, \boldsymbol{w}_j, \alpha), j \in F_2$
3:      Determine the winner category $J = \arg\max_{j \in F_2} T_j$
4:      Perform vigilance test(s) $V_J$ on $J$
5:      if TRUE then
6:          Update $J$
7:      else
8:          Reset $J$ so that $T_J = 0$
9:          if $T_j = 0, \forall j \in F_2$ then
10:             Commit a new node and set $J$ as the new node
11:             Update $J$
12:         else
13:             Go to Step 3
```

ART1 (or binary ART, described above), ART2 (for continuous inputs), fuzzy ART, Bayesian ART, Gaussian ART, validity index-based vigilance fuzzy ART are a few examples for unsupervised ART networks. The supervised versions of ART networks include another field containing the classes, known as the MAP field, and therefore go by the name of ARTMAP. In ARTMAP networks, the weight matrix between the MAP field and the category layer, F_2, links every category to a class. A few example ARTMAP algorithms are fuzzy ARTMAP, Bayesian ARTMAP, Gaussian ARTMAP, μARTMAP and ARTMAP-IC [38].

3.1.1 Fuzzy ART

Fuzzy ART is the most widely used architecture adapted from ART [39]. Fuzzy ART is the combination of ART1 structure with fuzzy logic so that the binary operations in ART1 are replaced by fuzzy operations.

The fuzzy ART algorithm is described below [7]:

1. Initialization
   - Learning rate (β), bias (α) and baseline vigilance (ρ) parameters are set.
   - The input vector $\boldsymbol{x}$ is normalized to $[0, 1]$ and complement coded:

   $$\boldsymbol{A} = (\boldsymbol{x}, \boldsymbol{x}^c) \tag{3.9}$$

   For an input vector $\boldsymbol{x} = (x_1, x_2, ...x_M)$, its complement is given as $\boldsymbol{x}^c = (x_1^c, x_2^c, ...x_M^c)$ where $x_i^c = 1 - x_i$, $i = 1, 2, ...M$.

2. Prototype Selection
   - A activates F_1 and is propagated to F_2 through $\boldsymbol{w}$. Activation of each prototype j in F_2 is through Weber's Law:

   $$T_j(\boldsymbol{A}) = \frac{|\boldsymbol{A} \wedge \boldsymbol{w}_j|}{\alpha + |\boldsymbol{w}_j|} \tag{3.10}$$

   where $|.|$ is the L^1 norm and $\wedge$ is the fuzzy AND operation, which is calculated as follows:

   $$(\boldsymbol{x} \wedge \boldsymbol{y})_k = min(x_k, y_k) \tag{3.11}$$

- The winner prototype is selected as the maximally activated prototype:

$$J = \arg\max_j T_j \tag{3.12}$$

3. Vigilance Test

   - Vigilance test is performed in F_1 with the winner prototype J:

$$\frac{|\boldsymbol{A} \wedge \boldsymbol{w}_J|}{|\boldsymbol{A}|} = \frac{|\boldsymbol{A} \wedge \boldsymbol{w}_J|}{M} \geq \rho \tag{3.13}$$

   - If J passes the vigilance test, learning is performed.
   - Otherwise, J is reset (T_J is set to 0 until the next input pair) and the winner prototype is selected as the one with the latter greatest activation. This search continues until a prototype that can pass the vigilance test is found. If such a prototype does not exist, a new prototype is initialized as the input itself such that $\boldsymbol{w}_{init} = A$.

4. Learning

   Learning corresponds to updating $\boldsymbol{w}_J$ as follows:

$$\boldsymbol{w}_J^{'} = \beta \boldsymbol{w} + (1 - \beta)\boldsymbol{w}_J \tag{3.14}$$

   where β is the learning rate. Fast learning mode corresponds to β=1.

The flowchart of fuzzy ART is provided in Figure 3.4.

3.1.1.1 Components of Fuzzy ART

There are a few operations in fuzzy ART that are not quite conventional for the machine learning literature. Yet, understanding the idea behind them is necessary to be able to comment on the performance of fuzzy ART.

The first one is normalization followed by complement coding. From Equation 3.10, the activation of a prototype can maximum be 1 (or very close to 1, as α is selected as a very small number). This case occurs if $\boldsymbol{A}$ and $\boldsymbol{w}_j$ are exactly the same.

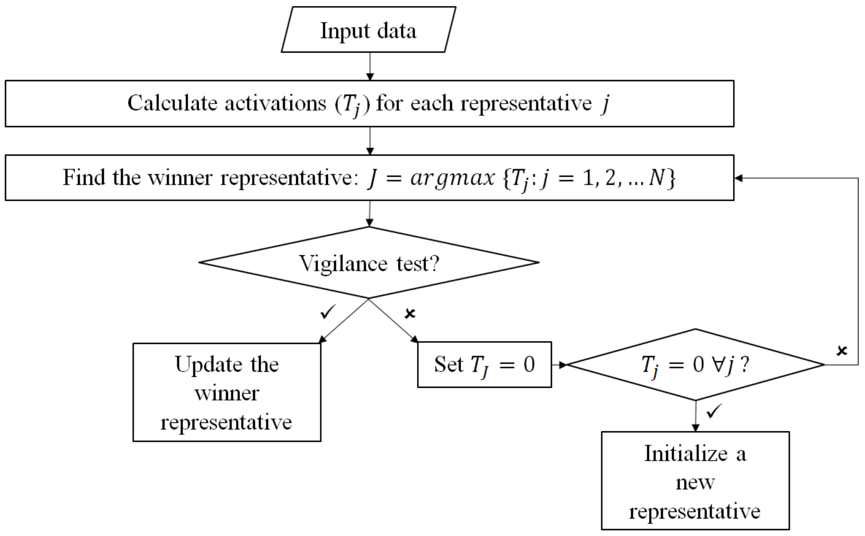

Input data
Calculate activations (T_j) for each representative j
Find the winner representative: $J = argmax\ \{T_j : j = 1, 2, \dots N\}$
Vigilance test?
✓
×
Update the winner representative
Set $T_J = 0$
$T_j = 0\ \forall j$?
×
✓
Initialize a new representative

Figure 3.4: The flowchart of fuzzy ART.

If there was no complement coding, the activation of $\boldsymbol{w}_j = (0.5, 0.5, ...0.5)$ given an input $\boldsymbol{x} = (0, 0, ...0)$ would be 0. Given the same input $\boldsymbol{x}$, the complement coded input signal will be found as $(0, 0, ...0, 1, 1, ...1)$. Hence, the activation of $\boldsymbol{w}_j = (0.5, 0.5, ...0.5)$ given $\boldsymbol{A}$ is $(0.5M)/(0.5 \times 2M) = 0.5$.

To observe the distance in maximally separated case, let $\boldsymbol{x} = (0, 0, ...0)$. Then, $\boldsymbol{A} = (0, 0, ...0, 1, 1, ...1)$. If $\boldsymbol{w}_j = (1, 1, ...1, 0, 0, ...0)$, the activation of $\boldsymbol{w}_j$ will be 0. Hence, to obtain maximum distance with complement coding, not just the positive, but also the negative ends of the input and weight vector should be opposite.

Another point to mention is the relationship between activation calculation and vigilance test. Weber's Law states that the size of the just noticeable difference (ΔI) is a constant proportion of the original stimulus value ($\Delta I/I = k$) [40]. From this point of view, it can be observed that the activation value of a weight vector $\boldsymbol{w}_j$ is a measure of the change caused by the input $\boldsymbol{A}$ on $\boldsymbol{w}_j$, whereas vigilance test compares the change caused by the winner prototype $\boldsymbol{w}_J$ on $\boldsymbol{A}$ to a threshold.

3.1.2 Bayesian ART

Bayesian ART is the combination of ART with Bayesian decision theory. The prototypes are multidimensional Gaussian distributions, thus each defined by a mean vector and covariance matrix, and one important advantage of Bayesian ART is that it does not require normalization (of input or weight vectors), thus complement coding is not included [8].

1. Prototype Selection

   First, the activation of each prototype $\boldsymbol{w}_j$ is calculated as the posterior probability of $\boldsymbol{w}_j$:

   $$T_j(\boldsymbol{x}) = P(\boldsymbol{w}_j|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{w}_j)P(\boldsymbol{w}_j)}{\sum_{k=1}^{N} p(\boldsymbol{x}|\boldsymbol{w}_k)P(\boldsymbol{w}_k)} \tag{3.15}$$

   where N is the total number of prototypes, $P(\boldsymbol{w}_j)$ is the estimated prior probability of w_j, $p(\boldsymbol{x}|\boldsymbol{w}_j)$ is the likelihood of $\boldsymbol{x}$ given $\boldsymbol{w}_j$ and the denominator term adds up to $p(\boldsymbol{x})$, known as the evidence.

   $P(\boldsymbol{w}_j)$ is calculated as the ratio of the number of input patterns that is under the prototype j to the number of input patterns received in total:

   $$P(\boldsymbol{w}_j) = \frac{N_j}{\sum_{k=1}^{N} N_k} \tag{3.16}$$

   where N_j is the number of input patterns under j and the denominator term adds up to the total number of input patterns received.

   $P(\boldsymbol{w}_j)$ allows “diminishing” the presence of a category that has not been observed for a long period, which is a property that fuzzy ART lacks. This is a useful property for emitter identification because it is unnecessary to point out a radar that is no longer active and it can even lead to a false strategy.

   The likelihood of the jth prototype that has mean $\boldsymbol{\mu_j}$ and covariance $\boldsymbol{\Sigma_j}$ is calculated by:

   $$p(\boldsymbol{x}|\boldsymbol{w}_j) = \frac{1}{(2\pi)^{M/2}|\boldsymbol{\Sigma_j}|^{1/2}} exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big) \tag{3.17}$$

   where M shows the dimension of input and weight vectors.

The winner prototype J is selected as the prototype that has the greatest posterior probability:

$$J = \arg\max_j T_j = \arg\max_j P(\boldsymbol{w}_j|\boldsymbol{x}) \tag{3.18}$$

2. Vigilance Test

The hypervolume of the winner prototype is compared to a limit S_{MAX}:

$$S_J \leq S_{MAX} \tag{3.19}$$

where S_J shows the hypervolume of J and defined as the determinant of its covariance matrix:

$$S_J = det(\boldsymbol{\Sigma}_J) = \prod_{i=1}^{M} {\sigma_{Ji}}^2 \tag{3.20}$$

The second equality of Eq. 3.20 is included only if a diagonal covariance matrix is assumed (which is the case in this thesis).

If the winner prototype satisfies Eq. 3.19, learning is performed.

Otherwise, if the hypervolume of J is beyond S_{MAX}, then it cannot accept any other input samples and it is is reset, meaning the vigilance test has failed. In this case, the prototype that has the latter greatest activation is selected as the winner prototype and passed to vigilance test.

If no prototype that passes the test can be found, a new prototype is initialized. The mean vector of the new prototype is taken as the input vector itself ($\boldsymbol{\mu}_{init} = \boldsymbol{x}$) and its covariance matrix, $\boldsymbol{\Sigma}_{init}$, is ensured to pass Eq. 3.19. In order to allow newly initialized prototypes to grow, $\boldsymbol{\Sigma}_{init}$ is started as $\sigma_{init}^2\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and $\sigma_{init}^2 \ll (S_{MAX})^{1/M}$.

3. Learning

The mean vector and covariance matrix of the winner prototype is updated as follows:

$$\boldsymbol{\mu}_J^{'} = \frac{N_J}{N_J+1}\boldsymbol{\mu}_J + \frac{1}{N_J+1}\boldsymbol{x} \tag{3.21}$$

$$\boldsymbol{\Sigma}_J^{'} = \frac{N_J}{N_J+1}\boldsymbol{\Sigma}_J + \frac{1}{N_J+1}(\boldsymbol{x}-\boldsymbol{\mu}_J^{'})(\boldsymbol{x}-\boldsymbol{\mu}_J^{'})^T * \boldsymbol{I} \tag{3.22}$$

where $*$ shows the element-wise product and included only if a diagonal covariance matrix is assumed because it ensures that the non-diagonal entries of $(\boldsymbol{x}-\boldsymbol{\mu}_J^{'})(\boldsymbol{x}-\boldsymbol{\mu}_J^{'})^T$ will be zero. The updated mean and covariance are shown with $\boldsymbol{\mu}_J^{'}$ and $\boldsymbol{\Sigma}_J^{'}$ whereas $\boldsymbol{\mu}_J$ and $\boldsymbol{\Sigma}_J$ show the previous mean and covariance respectively.

3.2 Self-Organizing Map

A self-organizing map (SOM) aims to transform the input space into nodes of a one or two dimensional grid [41]. Figure 3.5 shows a two dimensional SOM grid. Every node on the grid has a weight vector that is of the same size as the input vector. This grid is topologically ordered, meaning that closer nodes on the grid are likely to represent the same cluster of the input. SOM also utilizes competitive learning while learning the input data.

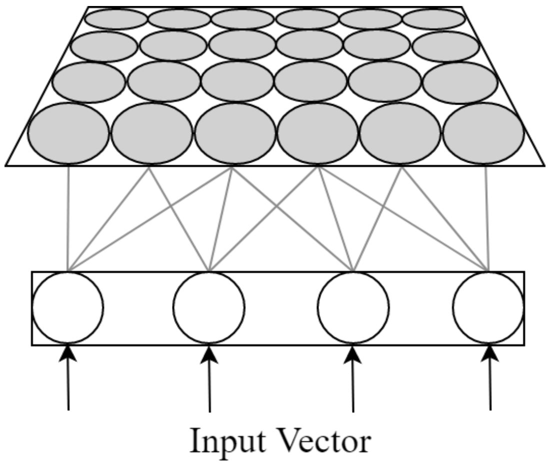


Figure 3.5: An example 6x4 SOM grid. Each node on the grid is attached to each input dimension through its weight vector.

The algorithm works as follows [42]:

1. Node weights are initialized randomly.

2. An input vector $\boldsymbol{x}$ is selected randomly.

3. The node whose weight vector is closest to the input is found. This is called as the "best-matching unit" (BMU) in SOM and here denoted by b such that:

$$||\boldsymbol{x} - \boldsymbol{w}_b|| = min||\boldsymbol{x} - \boldsymbol{w}_j|| \tag{3.23}$$

   where $\boldsymbol{w}_j$ shows the weight vector of jth node.

4. The weights of the BMU and its neighbors are updated [43]:

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + \alpha(t)h_{bi}(t)(\boldsymbol{x} - \boldsymbol{w}_i(t)) \text{ for } i \in N_b \tag{3.24}$$

   where N_b is the set of neighbors of the BMU, including the BMU itself, t shows the time, $\alpha(t)$ is the adaptation coefficient that decreases monotonically with time and $h_{bi}(t)$ is the neighborhood kernel which is given below:

$$h_{bi}(t) = exp\bigg(-\frac{||r_b - r_i||^2}{2\sigma^2(t)}\bigg) \tag{3.25}$$

   where r_b and r_i are the positions of the BMU and ith node on the SOM grid respectively and the variance $\sigma(t)$ is a monotonically decreasing function of t.

Finally, all input vectors will have been mapped to another space (the SOM grid) that is topologically ordered as it can be seen in Figure 3.6. Every color represents a cluster of input data and every square separated with grids shows a node. As it can be observed, close nodes represent the same cluster.

Although SOM uses competitive learning as well, unlike ART networks, not only the winner node, but also its neighborhood is updated, but on a less effective scale. The nodes in SOM are different from classes. A group of nodes is likely to represent a class rather than a single node as there are many nodes in SOM and not only the BMU, but also its neighborhood is updated with the input.

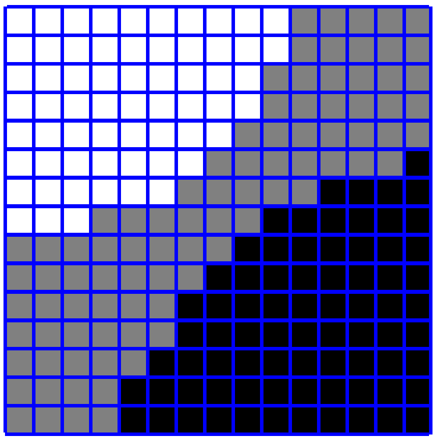


Figure 3.6: An example SOM grid obtained with three Gaussian distributions having different means. Each color (white, gray and black) shows a cluster of the input data. The squares separated with grid lines are the SOM nodes and each node is associated with a weight vector that is of the same size as the input data. Close nodes on the grid represent the same cluster, i.e. carry the same color.

3.2.1 Self-Organizing Incremental Neural Networks

The self-organizing incremental neural network (SOINN) is a prototype-based unsupervised online learning algorithm that is a combination of SOM and growing neural gas (GNG). SOINN has a quite similar structure to GNG. Both use a graph based approach and aim to perform clustering on sequential input by defining edges between similar nodes. For this, they both search for two winners that are the two closest ones to the current input and add an edge between the winners, unless it already exists. They (re)set the age of a newly added or newly used edge to zero and increment the age of all edges by one. The edges that reach a certain age, age_{max}, are deleted.

The main difference is that GNG always adds an edge between two winners or resets the age of the existing edge between them to zero. In GNG, the node insertion process is as follows [44]:

1. The distance between the input $\boldsymbol{\xi}$ and the first winner node s_1 is noted as the error of the node.

$$\Delta error(s_1) = ||\boldsymbol{w}_{s_1} - \boldsymbol{\xi}|| \tag{3.26}$$

   where $\boldsymbol{w}_{s_1}$ is the weight vector of s_1.

2. After a certain number of iterations, the node q with the most accumulated error is found.

3. The node f having the greatest error among the neighbors of q is determined.

4. A node having a weight vector w_r halfway between q and f is inserted to the graph:

$$\boldsymbol{w}_r = 0.5(\boldsymbol{w}_q + \boldsymbol{w}_f) \tag{3.27}$$

   where $\boldsymbol{w}_q$ is the weight vector of q and $\boldsymbol{w}_f$ is the weight vector of f.

As stated by [32], GNG does not check whether insertion of an edge or node is useful or not. This causes GNG to be not as suitable in non-stationary environments as too much node insertion may cause over-fitting. There should be a mechanism to check the utility of inserting a new node, considering when and how it will be inserted.

SOINN solves this problem by using two adaptive thresholds, one for the first and the other for the second winner, to decide whether to insert an edge (or reset an existing edge) or insert a node. Only if the distance between the input pattern and the first and second winner is less than these thresholds, an edge is connected between them or the existing edge in between is reset. Otherwise, the node set is extended with a node having a weight vector that is equal to the input pattern itself.

The SOINN used in this thesis is adjusted SOINN classifier as the original version has a two-layer framework and too many parameters. The adjusted SOINN proposed in [45] is much faster, requires less parameters and obtains almost the same performance. The algorithm of (adjusted) SOINN is given in Algorithm 3.

The number of wins of each node is stored in the corresponding index of the vector t as in Step 17. While updating the weight of the winner and its neighborhood, the learning rate is determined using the number of wins (Step 18 and Step 19). The weight update equation is formulated accordingly with Hebbian rule.

The learning rate of the winner node (ε_1) and the learning rate of its neighborhood (ε_2) is given by

$$\varepsilon_1(t) = \frac{1}{t} \quad , \quad \varepsilon_2(t) = \frac{1}{100t}. \tag{3.28}$$

Algorithm 3 SOINN

```
1: Initialize node set $V$ with two nodes $c_1$ and $c_2$ whose weight vectors $\boldsymbol{w}_{c_1}$ and $\boldsymbol{w}_{c_2}$
   are selected randomly from the input set: $V = \{c_1, c_2\}$
2: Initialize edge set $E = \emptyset$
3: Initialize number of processed input patterns $p = 0$
4: while there is new input pattern $\boldsymbol{\xi} \in \mathbb{R}^n$ do
5:     $p = p + 1$
6:     $s_1 = \arg\min_{c \in V} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$
7:     $s_2 = \arg\min_{c \in V \setminus \{s_1\}} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$
8:     Calculate thresholds $T_{s_1}$ and $T_{s_2}$ according to Equation 3.29
9:     if $||\boldsymbol{\xi} - \boldsymbol{w}_{s_1}|| > T_{s_1}$ or $||\boldsymbol{\xi} - \boldsymbol{w}_{s_2}|| > T_{s_2}$ then
10:        $V \leftarrow V \cup r$
11:        $\boldsymbol{w}_r = \boldsymbol{\xi}$
12:    else
13:        if $(s_1, s_2) \notin E$ then
14:            $E \leftarrow E \cup \{(s_1, s_2)\}$
15:        $age_{(s_1,s_2)} \leftarrow 0$
16:        $age_{(s_1,i)} \leftarrow age_{(s_1,i)} + 1, \forall i \in N_{s_1}$
17:        $t_{s_1} \leftarrow t_{s_1} + 1$
18:        $\boldsymbol{w}_{s_1} \leftarrow \boldsymbol{w}_{s_1} + \varepsilon_1(t_{s_1})(\boldsymbol{\xi} - \boldsymbol{w}_{s_1})$
19:        $\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \varepsilon_2(t_i)(\xi - \boldsymbol{w}_i), \forall i \in N_{s_1}$
20:        Find edges $E_{old} = \{(i,j) \mid (i,j) \in E, age(i,j) > age_{max}\}$
21:        Find nodes $i_{old} = \{i \mid i \in V, \{(i,j)\} \in E_{old}, \forall j \in N_i\}$
22:        Delete $E_{old} : E \leftarrow E \setminus\ E_{old}$
23:        Delete $i_{old} : V \leftarrow V \setminus\ i_{old}$
24:    if $p = k\lambda, k \in Z$ then
25:        Delete nodes $\{i \quad | \quad |N_i| \leq 1\}$
```

As Equation 3.28 shows, learning rate is less for nodes with more wins. This is a reasonable choice since a pattern that has coded the input many times before should not be changed a lot. The neighborhood of the winner node is also subject to less learning rate.

The threshold calculation of the winners in Step 8 of Algorithm 3 is formulated as:

$$T_i = \begin{cases} \max_{j \in N_i} ||\boldsymbol{w}_j - \boldsymbol{w}_i||, (N_i \neq \emptyset) \\ \min_{j \in V \setminus i} ||\boldsymbol{w}_j - \boldsymbol{w}_i||, \text{otherwise.} \end{cases} \tag{3.29}$$

Hence, the threshold region of a winner in SOINN corresponds to a hypersphere, centered at $\boldsymbol{w}_i$ and has a radius of T_i.

3.2.2 KDESOINN

SOINN with kernel density estimation (KDESOINN) proposes calculating the threshold regions based on the distribution of the neighborhood of the winner [5]. The threshold regions calculated with both SOINN and KDESOINN are compared in Figure 3.7.

As it can be seen in Figure 3.7, KDESOINN takes into account the shape of the neighborhood of the winner, which results in the input signal staying outside the threshold regions unlike SOINN. This is quite appropriate to radar signals as some emitters might transmit constant radar parameters whereas others might transmit jittered or staggered radar parameters and all of these cases possess different density functions. KDESOINN achieves this by adapting density estimation with Gaussian kernels.

Given training samples $\{\boldsymbol{x}_i\}_{i=1,2,\ldots N} \in \mathbb{R}^d$ from a density $p(\boldsymbol{x})$, an estimation $\hat{p}(\boldsymbol{x})$ of the density can be expressed as follows [46]

$$\hat{p}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} K_{\boldsymbol{\Sigma}}(\boldsymbol{x} - \boldsymbol{x}_i) \tag{3.30}$$

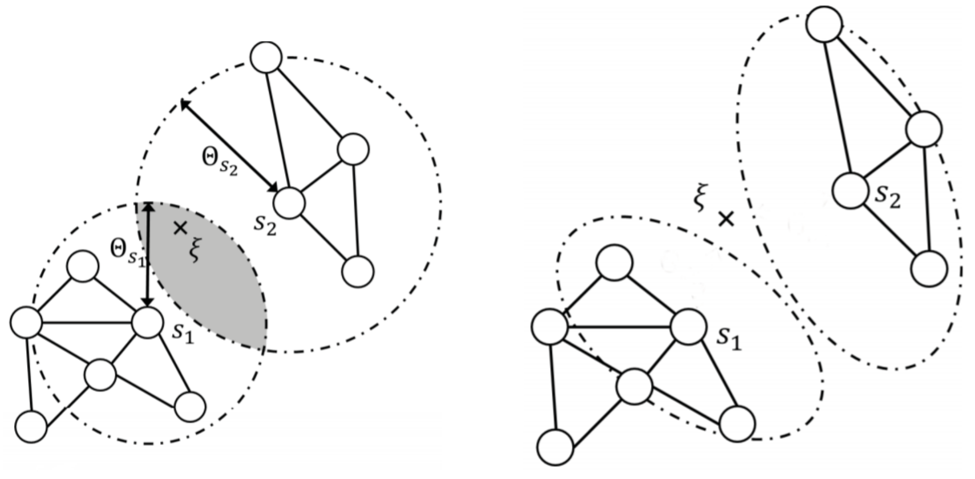


Figure 3.7: Threshold regions of SOINN (left) and KDESOINN (right) given the same input (ξ), winners (s_1 and s_2) and graph. SOINN constructs a hypersphere with radii Θ_{s_1} and Θ_{s_2}. The radii are calculated as the distance from the winner itself to its farthest neighbor as given in Equation 3.29. As it can be seen, the input is inside the regions determined by SOINN. The threshold regions in KDESOINN (calculated using kernel density estimation as in Equation 3.34), on the other hand, do not include the input [5].

where K is a kernel function with a bandwith (scale) of Σ. If it is Gaussian kernel,

$$K_{\mathbf{\Sigma}}(\boldsymbol{x}-\boldsymbol{\mu}) = \frac{1}{\sqrt{(2\pi)^d|\mathbf{\Sigma}|}} exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big) \tag{3.31}$$

where μ is the mean of samples.

In KDESOINN, Equation 3.30 is written as

$$\hat{p}(\boldsymbol{x}) = \frac{1}{T_V}\sum_{i\in V} t_i K_{C_i}(\boldsymbol{x}-\boldsymbol{w}_i) \tag{3.32}$$

where $T_V = \sum_{i\in V} t_i$. This way, The kernel located on node i is weighted by its number of wins t_n. The covariance matrix $\boldsymbol{C}_i$ is given by

$$\boldsymbol{C}_i = \frac{1}{T_{N_i}} \sum_{j \in N_i} t_j (\boldsymbol{w}_j - \boldsymbol{w}_i)(\boldsymbol{w}_j - \boldsymbol{w}_i)^T. \tag{3.33}$$

$\boldsymbol{C}_i$ is different from the original kernel density estimation in the sense that $\boldsymbol{\mu}$ is replaced by the weight vector of node i, samples are selected as the neighborhood of node i and the contribution of nodes are weighted by their number of wins.

Given the input $\boldsymbol{\xi}$, the threshold region of node i is

$$(\boldsymbol{\xi} - \boldsymbol{w}_i)^T \boldsymbol{M}_i^{-1} (\boldsymbol{\xi} - \boldsymbol{w}_i) \leq 1$$

where

$$\begin{aligned} \boldsymbol{M}_i &= \boldsymbol{C}_i + \rho \gamma_i \boldsymbol{I} \\ \gamma_i &= \begin{cases} \max_{j \in N_i} ||\boldsymbol{w}_j - \boldsymbol{w}_i||, (N_i \neq \emptyset) \\ \min_{j \in V \setminus i} ||\boldsymbol{w}_j - \boldsymbol{w}_i||, \text{otherwise.} \end{cases} \end{aligned} \tag{3.34}$$

where I is the identity matrix and ρ is the parameter for thresholds. The identity matrix is added for smoothing, i.e. a node having only a few edges may have a very strict or small threshold region without the added term.

The complete KDESOINN is provided in Algorithm 4. When compared with Algorithm 3, two other changes besides threshold calculation can be seen.

Firstly, neighborhood weights are not updated, only the winner node has the right to learn the input. This approach is preferred as kernel density estimation is also only based on the winner node.

The second and last difference is that at each λth iteration, a k-nearest neighbor (k-NN) graph with nodes in V is constructed. A k-NN graph is a directed graph where an edge from a node p to q is added if q is among the k-nearest neighbors of p [47].

The k-NN step is added to KDESOINN since the original SOINN can add an edge between two nodes only if they are the two winners for an input pattern. However,

Algorithm 4 KDESOINN

```
1: Initialize node set $V$ with two nodes $c_1$ and $c_2$ whose weight vectors $\boldsymbol{w}_{c_1}$ and $\boldsymbol{w}_{c_2}$
   are selected randomly from the input set: $V = \{c_1, c_2\}$
2: Initialize edge set $E = \emptyset$
3: Initialize number of processed input patterns $p = 0$
4: while there is new input pattern $\boldsymbol{\xi} \in \mathbb{R}^n$ do
5:     $p = p + 1$
6:     $s_1 = \arg\min_{c \in V} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$
7:     $s_2 = \arg\min_{c \in V \setminus \{s_1\}} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$
8:     if $(\boldsymbol{\xi} - \boldsymbol{w}_{s_1})^T {M_{s_1}}^{-1}(\boldsymbol{\xi} - \boldsymbol{w}_{s_1}) > 1$ or $(\boldsymbol{\xi} - \boldsymbol{w}_{s_2})^T {M_{s_2}}^{-1}(\boldsymbol{\xi} - \boldsymbol{w}_{s_2}) > 1$ then
9:         $V \leftarrow V \cup r$
10:        $\boldsymbol{w}_r = \boldsymbol{\xi}$
11:    else
12:        if $(s_1, s_2) \notin E$ then
13:            $E \leftarrow E \cup \{(s_1, s_2)\}$
14:        $age_{(s_1,s_2)} \leftarrow 0$
15:        $age_{(s_1,i)} \leftarrow age_{(s_1,i)} + 1, \forall i \in N_{s_1}$
16:        $t_{s_1} \leftarrow t_{s_1} + 1$
17:        $\boldsymbol{w}_{s_1} \leftarrow \boldsymbol{w}_{s_1} + \varepsilon_1(t_{s_1})(\xi - \boldsymbol{w}_{s_1})$
18:        Find edges $E_{old} = \{(i,j) \mid (i,j) \in E, age(i,j) > age_{max}\}$
19:        Find nodes $i_{old} = \{i \mid i \in V, \{(i,j)\} \in E_{old}, \forall j \in N_i\}$
20:        Delete $E_{old} : E \leftarrow E \setminus\ E_{old}$
21:        Delete $i_{old} : V \leftarrow V \setminus\ i_{old}$
22:    if $p = k\lambda, k \in Z$ then
23:        Delete nodes $\{i \quad | \quad |N_i| = 0\}$
24:        Create a k-NN graph $G$ whose set of nodes is $V$ and call its edges $E_k$
25:        $E \leftarrow E \cup \{(i,j) \mid (i,j) \in E_k, (j,i) \in E_k\}$
26: Create a k-NN graph $G$ whose set of nodes is $V$ and call its edges $E_k$
27: $E \leftarrow E \cup \{(i,j) \mid (i,j) \in E_k, (j,i) \in E_k\}$
```

even if such an input does not exist, two nodes might still belong to the same class if they are similar enough to one another. To be able to add an edge in such a case, it is checked whether two nodes i and j are both included in each other's k-nearest neighbors, i. e., whether a duplex edge exists between i and j in the k-NN graph.

Figure 3.8 shows a clustering example performed with KDESOINN when three different Gaussian distributions are given as input.

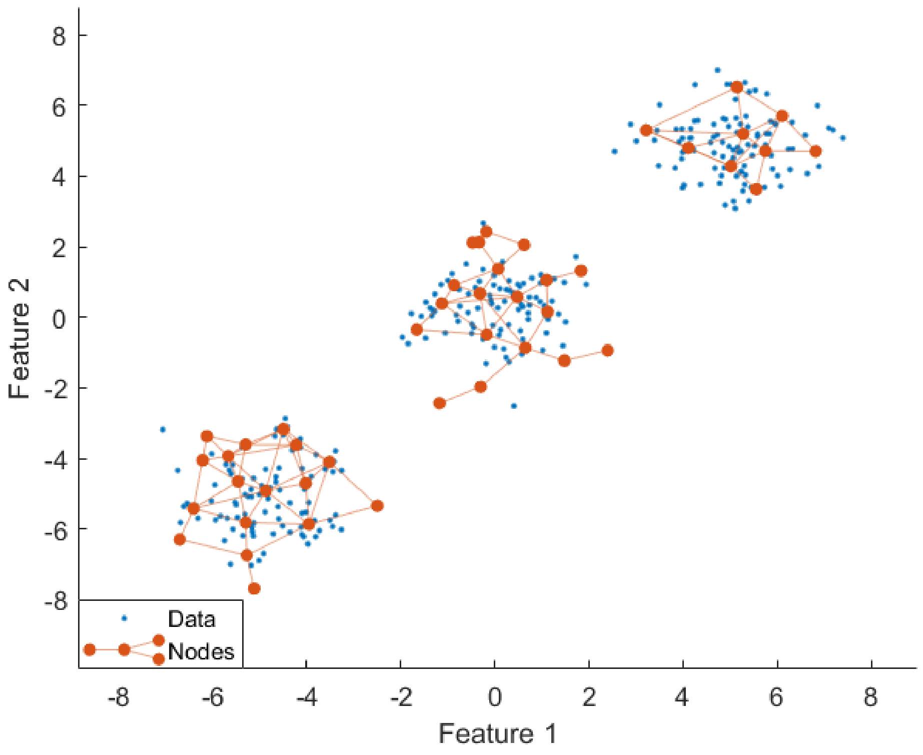


Figure 3.8: Clustering with KDESOINN. Each connected graph (set of nodes) represents a Gaussian distribution with a different mean.

3.3 Incremental Learning Literature for Emitter Identification

Incremental learning is a necessity for emitter identification as emitter representatives should be updated continuously over time. The sPDWs from the deinterleaver include many pulses, so first a feature extraction must be applied to initiate learning as shown in Figure 1.2. In order to capture the range of the useful radar parameters for emitter identification which are RF, PW, PRI and DOA; minimum, maximum and mean values of each of them are used to obtain a summarized information for the sPDW they belong to. Hence, the input data to be learned is as provided below:

$$\begin{aligned}\boldsymbol{x} =&[RF_{min}\ RF_{mean}\ RF_{max}\ PW_{min}\ PW_{mean}\ PW_{max} \\ &PRI_{min}\ PRI_{mean}\ PRI_{max}\ DOA_{min}\ DOA_{mean}\ DOA_{max}]^T\end{aligned} \tag{3.35}$$

Due to unknown emitter types, incremental learning algorithms with dynamic architectures ([26]) (i.e., expandable networks) are investigated for this thesis. One such network is ART which is described in detail in Section 3.1.

In the literature, ART (especially fuzzy ART) and SOM have been used for deinterleaving [48] [18] as well as identification [49] [50] [51]. In [52], a vector neural network is proposed that can take both intervals and scalars as input in order to capture the range of values that radar parameters can possess. Although in this thesis, identification is performed in two steps, the first one being incremental learning and the second being interval based analysis. To handle another significant challenge related to radar parameters that is jitter, fuzziness is frequently considered for the problem of emitter identification. For example, [53] proposed a self-organizing interval type-2 fuzzy neural network to model the uncertainties in radar signals so that they can be minimized by assigning a Gaussian membership function to each input radar parameter with adjustable mean and the proposed algorithm was observed to be less sensitive to noise and jitter compared to vector neural networks. Hierarchical clustering has also been studied for emitter identification [54] [55], as well as Support Vector Machine (SVM) based approaches [56] [57]. Partitive clustering algorithms such as K-means require the knowledge of number of clusters beforehand [58], and therefore not suitable for the problem of emitter identification.

It should be noted that there are recent trends in machine learning such as deep neural networks or zero shot learning. There are no published papers for emitter identification in the literature based on these recent topics. Although these concepts are quite promising, there should be various reasons for not applying these techniques to the emitter identification problem.

The deep learning literature tries to address incremental learning through progressive neural networks [6], which were first proposed to incorporate new tasks to already trained models in Atari games. Progressive neural networks have the capability to add a new task to an already trained network as input data is presented sequentially to the network. Their motivation comes from the fact that when human brain learns a new task, it does not start from zero or scratch. It instead uses previous experience. In order to integrate the ability to transfer knowledge from previous tasks to improve convergence speed in deep architectures, progressive networks instantiate a new neural network (a column) for each task being solved, while enabling information transfer via lateral connections (shown in Figure 3.9) to features of previously learned columns [6].

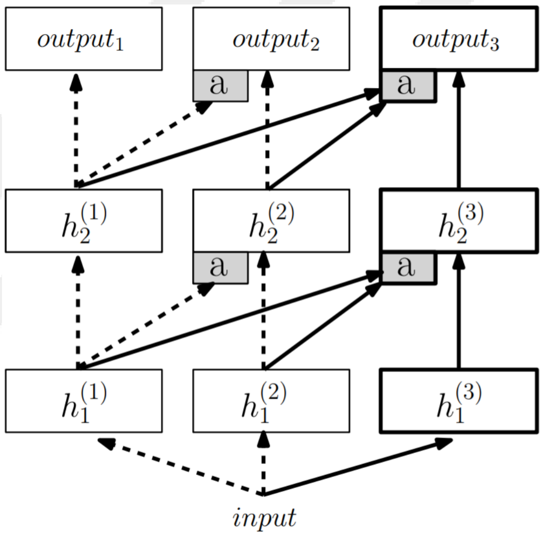


Figure 3.9: Feature transfer in a progressive neural network of three columns. The first two columns on the left, with dashed arrows, are trained on first and second tasks respectively. For the final task, a third column is added which is given access to all previously learned features [6].

Although progressive networks deal with incremental learning, they approach the problem in a task-based manner instead of classification-based. Transferring features from previous columns to a new column might interfere with the nature of classification in the sense that classification can only be successful if classes exhibit different features. Consequently, they are more applicable to reinforcement learning related tasks but not to emitter identification.

On the other hand, as classes having no training samples are of concern in emitter identification, zero shot learning (ZSL) [59] might seem perfectly convenient at the first sight. In classification algorithms, algorithms are trained with samples of all classes in the problem and then made to choose which one of these classes a new test sample belongs to. With ZSL, algorithms can decide which class a test sample belongs to even if they were never shown an example of that class before. In other words, using ZSL, images from unseen classes, i.e. zero shot classes, can be correctly classified. However, while doing so, textual information is utilized. Simply put, in ZSL, the verbal description of each class is assumed to be known, but the training examples of some of them are missing. For instance, a classifier is trained with images of only cats and dogs, but is also provided the description of how birds look like. From the training samples of cats and dogs, the classifier learns a mapping from an input image to its verbal description. Using this mapping, the classifier can obtain the verbal description of zero shot bird images as well. The later task is relatively simple, which is to find the most similar description to the current one [59].

Unfortunately, in emitter identification, the system may lack not only radar samples for training, but also any kind of description for some emitter types. Hence, ZSL does not cover all aspects of emitter identification.

Finally, there are also autoencoder based approaches for incremental learning. Neurogenesis deep learning considers an autoencoder initially trained with a subset of classes and how continuous adaptation can be managed by learning each remaining class. If the reconstruction error at any layer of the autoencoder is too high, a predefined number of new nodes are added to that layer [60].

However, deep architectures in general rely on large datasets and tend to suffer from overfitting in case of insufficient samples [61]. In emitter identification, a list of all active emitters at any moment should be available, which means classes with very few samples are likely to be observed, and they still need to be presented correctly.

Based on the literature survey on emitter identification and incremental learning methods with dynamic architectures and the limitations of emitter identification such as classes having very few samples, four algorithms are considered to address the problem. These are fuzzy ART, Bayesian ART, self-organizing maps and self-organizing incremental neural networks.

3.4 Symbolic Data Analysis

SDA aims to extend classical data analysis methods to be applicable to symbolic data. In this stage, the representations for each cluster are compared with the representations of previously available emitter types.

Table 3.1 shows an example symbolic data set. The person in the first row shows a male with a brain tumor living in Boston in his 20's and having a blood pressure in the range (79,120). The person in the fourth row has breast cancer with probability p and lung cancer with probability (1-p), as the distribution of features can be of concern as well under SDA.

Table 3.1: An example symbolic data set [1]

u	Age	Blood Pressure	City	Type of Cancer	Gender
1	[20,30)	(79,120)	Boston	Brain tumor	Male
2	[50,60)	(90,130)	Boston	Lung, liver	Male
3	[45,55)	(80,130)	Chicago	Prostate	Male
4	[47,47)	(86,121)	El Paso	Breast p, Lung (1-p)	Female

SDA is the general concept of analysing symbolic data, yet mostly interval data is considered under SDA [55]. In fact, many other data types are included within SDA, such as discrete values, stochastic distributions or verbal attributes. In other words, any data type can be considered as symbolic data.

Various metrics are utilized to define the distance between two intervals, $A_1 = [a_1, b_1]$ and $A_2 = [a_2, b_2]$. Hausdorff [62], Wasserstein [63] and Jaccard [64] distance are among the most prominent metrics and have all been used to analyze and cluster symbolic data [65].

3.4.1 Hausdorff Distance

One of the most widely used metrics is the Hausdorff distance [66]. When written in L_2 norm, Hausdorff distance is defined as [67]:

$$d_H(A_1, A_2) = max(|a_1 - a_2|, |b_1 - b_2|) \tag{3.36}$$

For two p-dimensional vectors of intervals, namely $x_i = ([a_i^1, b_i^1], ...[a_i^p, b_i^p])$ and $x_j = ([a_j^1, b_j^1], ...[a_j^p, b_j^p])$, the Hausdorff distance is equal to:

$$d(x_i, x_j) = \sum_{k=1}^{P} d_H(x_i^k, x_j^k) = \sum_{k=1}^{P} max(|a_i^k - a_j^k|, |b_i^k - b_j^k|) \tag{3.37}$$

3.4.2 Wasserstein Distance

Although its calculation is quite simple and fast, Hausdorff metric is proven to underestimate distance [68]. Wasserstein metric takes into account the distribution of the vectors that are being compared. The Wasserstein distance between two vectors with cumulative distribution functions F and G are as follows:

$$d_W(F, G) = \int_0^1 |F^{-1}(t) - G^{-1}(t)|dt \tag{3.38}$$

where F^{-1} and G^{-1} show the inverse distribution, or quantile function. When the histogram of the vectors are used as an empirical estimation of their probability density functions, their quantile functions can also be approximated [69].

Wasserstein metric gives a precise description of the distance, with the price of being costly. It is therefore used as a loss function in machine learning [70]. It provides a much more detailed measure compared to Hausdorff distance. For example, let b_1 be slightly larger than a_2 and smaller than b_2 so that the region of intersection of the two vectors A_1 and A_2 is relatively small compared to $(b_2 - a_1)$. If the distribution of A_1 and A_2 are more dense at the region of intersection, then A_1 and A_2 might actually be similar with both including a few outlier data. However, these outliers will cause the Hausdorff distance between them to be large and thus inaccurate.

3.4.3 Jaccard Distance

Jaccard index (Eq. 4.2) is actually a similarity metric in the range $[0, 1]$. Hence, by subtracting it from 1, it is possible to obtain a dissimilarity or distance metric (shown by J_I) for two intervals, $A_1 = [a_1, b_1]$ and $A_2 = [a_2, b_2]$, based on Jaccard index as follows:

$$J_I(A_1, A_2) = 1 - J(A_1, A_2) = \frac{|A_1 \cup A_2| - |A_1 \cap A_2|}{|A_1 \cup A_2|} \tag{3.39}$$

The set of intersection can be defined as:

$$A_1 \cap A_2 = \begin{cases} [\max(a_1, a_2), \min(b_1, b_2)] & \text{, if} \max(a_1, a_2) < \min(b_1, b_2) \\ 0 & \text{, otherwise.} \end{cases} \tag{3.40}$$

The $|.|$ operation in Eq. 3.39 corresponds to the length of the input interval and $|A_1 \cup A_2|$ can be written as $|A_1| + |A_2| - |A_1 \cap A_2|$.

3.5 Symbolic Data Analysis Literature for Emitter Identification

As radar parameters are quite complex, SDA methods have been applied to radar signal processing before. For example, [55] used Jaccard distance for emitter identification whereas [71] used Hausdorff distance. Symbolic representations have also been used for classification of time series radar data [72].

If only one representative exists for a cluster as in fuzzy ART and Bayesian ART, the minimum and maximum values for each feature in the representative vector are used to construct the interval for that feature. If multiple representatives exist as in SOM and KDESOINN, the range that the representatives under the same cluster can reach to is noted as the interval value of that feature. In either case, the clusters or representatives from the incremental learner are turned into interval valued representatives which will be the input vectors for SDA:

$$\boldsymbol{x} = ([RF_{min}, RF_{max}], [PW_{min}, PW_{max}], [PRI_{min}, PRI_{max}], [DOA_{min}, DOA_{max}]) \tag{3.41}$$

The final output of identification is a list of active emitters, with each emitter begin shown by the range in which its parameters (such as RF) operate. The a priori knowledge, i. e. previously available emitter types, are also of this form. The emitter representatives (or clusters) from the incremental learner are first transformed into interval valued representatives.

Each representative is to be compared with the previously available emitter types and the list of active emitters. If it is close enough to a known type or an already-active emitter, then it is assigned to it and the list of active emitters is extended with its type or the existing active emitter is updated respectively. Otherwise, it is given a different type name to stand for itself, and the list of active emitters is extended with its information. This cycle is summarized in Figure 3.10.

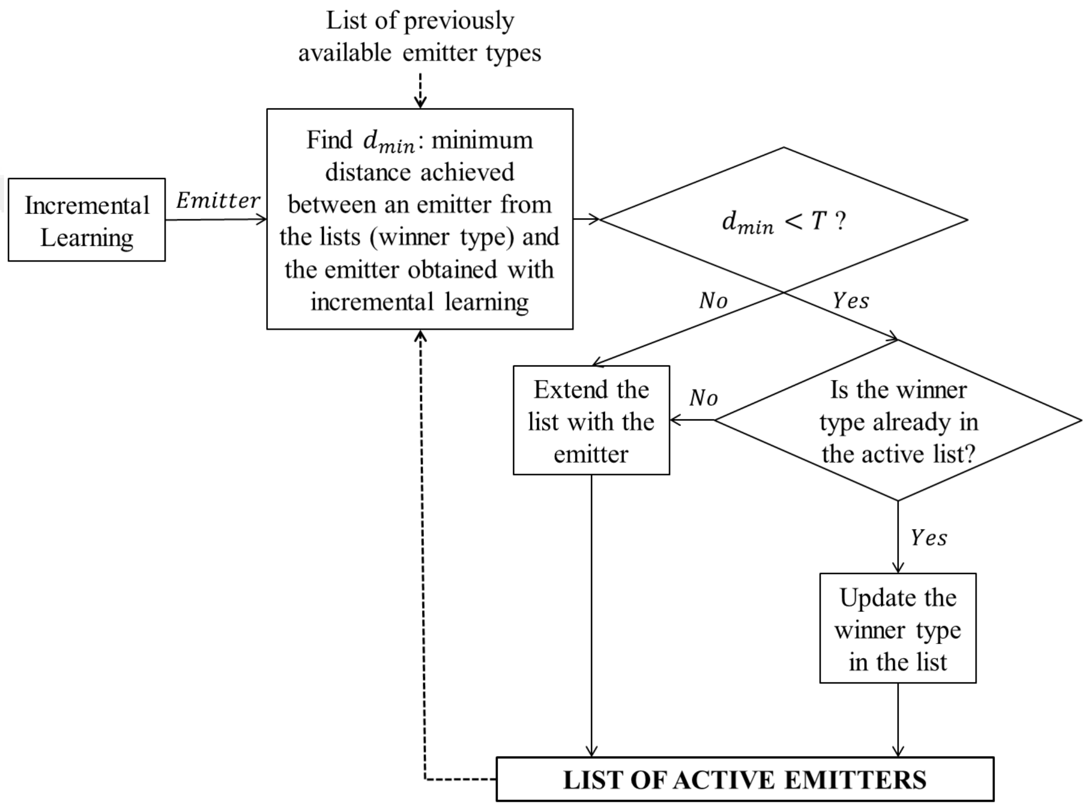

List of previously available emitter types
Incremental Learning
$Emitter$
Find d_{min}: minimum distance achieved between an emitter from the lists (winner type) and the emitter obtained with incremental learning
$d_{min} < T$?
No
Yes
Extend the list with the emitter
No
Is the winner type already in the active list?
Yes
Update the winner type in the list
LIST OF ACTIVE EMITTERS

Figure 3.10: Construction of the list of active emitters.

CHAPTER 4

PROPOSED EMITTER IDENTIFICATION METHODS

Radar data is quite complex in the sense that it is subject to noise, it is interval based and its parameters are very different in magnitude, meaning that its dimensions should also be scaled differently whenever required (such as for normalization). In this thesis, a few extensions are proposed to each algorithm both to make them more convenient to radar signal processing and to enhance their performance in general.

First, the proposed versions of each method is compared with their baseline versions. Then, all of the proposed algorithms are compared to each other. These tests are conducted on simulator data and a provided data set (external). The results in this part are only in terms of accuracy (determined by cluster purity and ground truth purity) as the computation time for all incremental learning methods being tested are observed to be quite similar.

The simulator results are given as the average of 1000 randomly generated scenarios from the simulator (Appendix A), which contains 4 to 20 mixed radars at each scenario. The external data set includes 22 radars, and the radar parameters can be constant or varying for each one of the radars.

All of the tests are run on MATLAB.

4.1 Proposed Extensions on Fuzzy ART

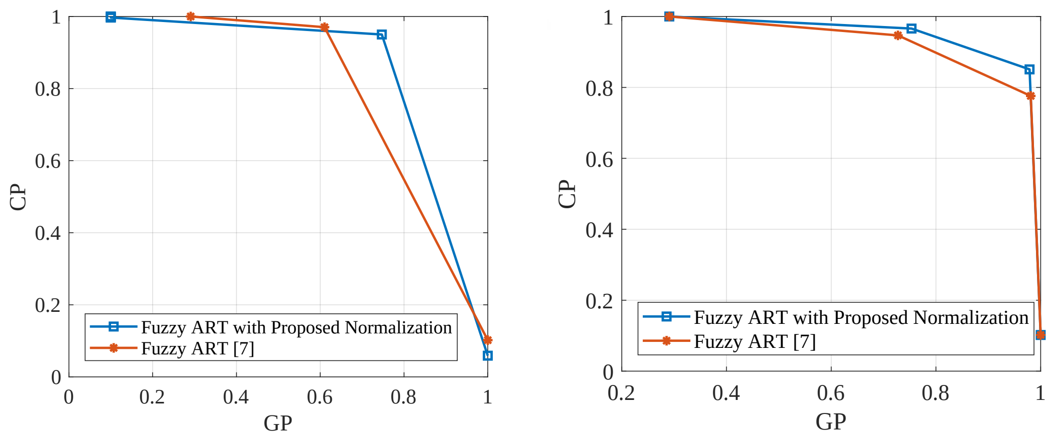
4.1.1 Normalization

Fuzzy ART suffers from the fact that the input data has to be normalized. The suggested normalization in fuzzy ART is as follows:

$$\boldsymbol{x} \leftarrow \boldsymbol{x}/|\boldsymbol{x}| \tag{4.1}$$

However, the dimensions of the vector $\boldsymbol{x}$ in emitter identification are quite different in magnitude. RF is on the order of thousands (of MHz) whereas PW can be under 1 (μs). Dividing $\boldsymbol{x}$ by its magnitude will suppress PW as it will approach to zero. Hence, a vector is constructed form normalization containing the maximum limits of each radar parameter so that when $\boldsymbol{x}$ is element-wise divided by this vector, every dimension will be normalized within its own range.

The effect of the proposed normalization can be observed in Figure 4.1.



(a) Simulator data.

(b) External data set.

Figure 4.1: Fuzzy ART with proposed normalization compared with the original Fuzzy ART [7]. Vigilance parameter is swept.

As ideally both cluster purity and ground truth purity would be 1, the top right corner of the cluster purity - ground truth purity plots is the most desirable point of operation. From this point of view, the proposed normalization can be observed to enhance the performance of fuzzy ART in Figure 4.1.

Note that the convex hull of the cluster purity - ground truth purity pairs is shown in all of the plotted results.

4.1.2 Activation Function and Vigilance Test

Due to the fact that both activation calculation and vigilance test searches for the similarity between weight vectors and the input signal, instead of checking this similarity twice, only one function can be used for each. As the same function is used for both, in case that the most activated prototype is not adequate to represent the input sample, a new class can be added immediately. There will be no need to reset the winner prototype and check the remaining ones, which will decrease the complexity as well.

The proposed activation function is indeed a combined version of the original activation function and vigilance test. Instead of dividing the norm of intersection of the input and prototypes by the norm of the input and the norm of the prototypes separately, the denominator of the combined function can be selected as the union of them [73]. This similarity measured by intersection-over-union is known as Jaccard similarity (Section 3.4.3) [74]:

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{4.2}$$

In order to stay convenient with fuzzy logic, the intersection and union operations are fuzzy in the proposed activation function:

$$T_j(\boldsymbol{A}) = \frac{|\boldsymbol{A} \wedge \boldsymbol{w}_j|}{|\boldsymbol{A} \vee \boldsymbol{w}_j|} \tag{4.3}$$

The flowchart of fuzzy ART with the proposed activation and vigilance test is provided in Figure 4.2.

The proposed activation function and vigilance test has dropped the execution time

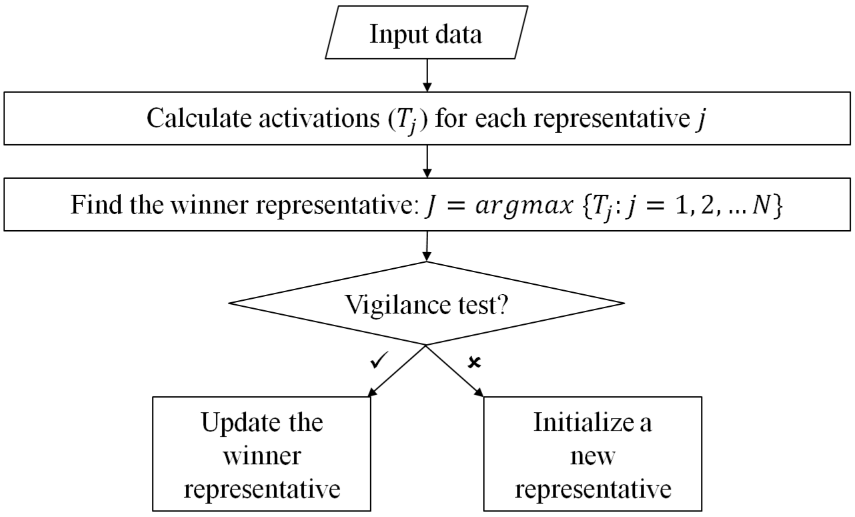

Input data
Calculate activations (T_j) for each representative j
Find the winner representative: $J = argmax\ \{T_j : j = 1, 2, \dots N\}$
Vigilance test?
✓
×
Update the winner representative
Initialize a new representative

Figure 4.2: The flowchart of improved fuzzy ART.

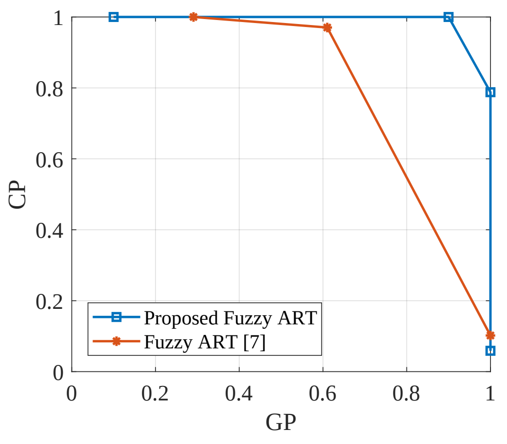
of fuzzy ART from 57.2 milliseconds to 29. This proves that the complexity has been decreased as it was aimed.

The proposed fuzzy ART, with both extensions, is compared with the original fuzzy ART algorithm. The results are provided in Figure 4.3.

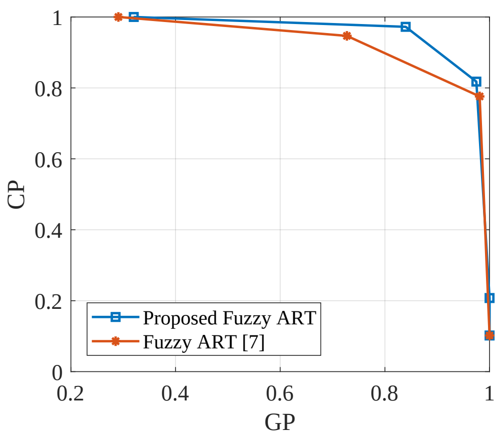
4.2 Proposed Extension on Bayesian ART

The vigilance test of Bayesian ART does not take into account how similar the input vector is to the winner prototype, it only checks the hypervolume of the winner. However, an input vector that is quite similar to the winner may not even induce a noticeable change in its hypervolume, thus not including the input under the winner prototype would not be fair.

Consequently, another vigilance test that considers the effect of the input signal on the winner prototype is proposed with this thesis. The proposed vigilance test compares the Mth power (where the input signal $\boldsymbol{x} \in \mathbb{R}^M$) of the geometric mean of the squared element-wise difference between the input signal and the mean vector of the winner



(a) Simulator data.



(b) External data set.

Figure 4.3: Proposed fuzzy ART (with both extensions) compared with the original Fuzzy ART [7]. Vigilance parameter is swept.

prototype with the determinant of the covariance of the winner prototype, $\boldsymbol{\Sigma}_J$:

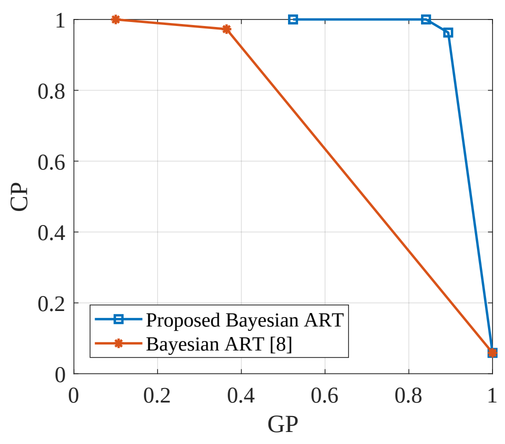
$$\prod_{i=1}^{M}(x_i - \mu_{J_i})^2 < c \times det(\boldsymbol{\Sigma}_J) \tag{4.4}$$

where x_i and μ_{J_i} show the ith entry of $\boldsymbol{x}$ and $\boldsymbol{\mu_J}$ respectively. If the left hand side of Eq. 4.4 is less than or at least similar to (this similarity is defined by the user-input parameter, c) the determinant of $\boldsymbol{\Sigma}_J$, vigilance test is successful.

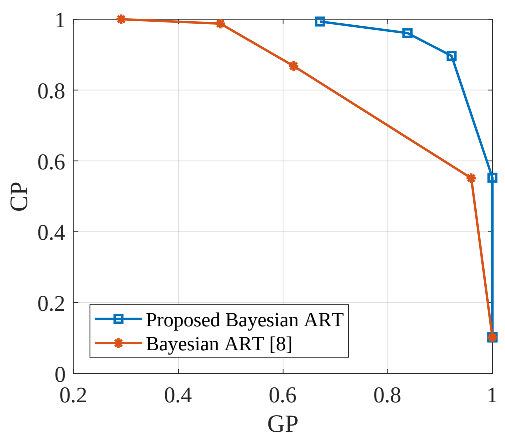
The proposed Bayesian ART is compared to the baseline Bayesian ART. The results are provided in Figure 4.4.

4.3 Clustering Stage of Self-Organizing Map

SOM allows a mapping from a high dimensional input space to a less dimensional (generally two) topological space. However, if the user wants more than to project input data into a low-dimensional space or to visualize it on such a space, a clustering of nodes will be necessary, which is the case for emitter identification.



(a) Simulator data.



(b) External data set.

Figure 4.4: Proposed Bayesian ART compared with the original Bayesian ART [8]. S_{MAX} and c is swept for the original and proposed implementation of Bayesian ART respectively.

The clustering approaches for SOM are divided into two: hierarchical and partitive. Partitive clustering methods usually require the number of clusters k to be known a priori and aim to minimize (or maximize) an objective function [58] such as K-means [75].

Hierarchical clustering can be two ways: bottom-up (agglomerative) and top-down (divisive). Among the both, agglomerative clustering is more widely used. In this thesis, agglomerative clustering is utilized to cluster the SOM nodes.

The steps of agglomerative clustering are as below [76]:

1. Each data point is regarded as a single cluster.
2. The distance between each cluster is calculated.
3. Merge the two clusters that have the least distance.
4. Return to Step 2 until there is only one cluster.

This way, a dendrogram is constructed which indicates the distance between clusters at each iteration, which in the beginning are the input samples themselves. An example dendrogram obtained with SOM node weights on simulator data is provided in Figure 4.5. The wider the links are or the longer the links reach, it means the distance between merged clusters is higher.

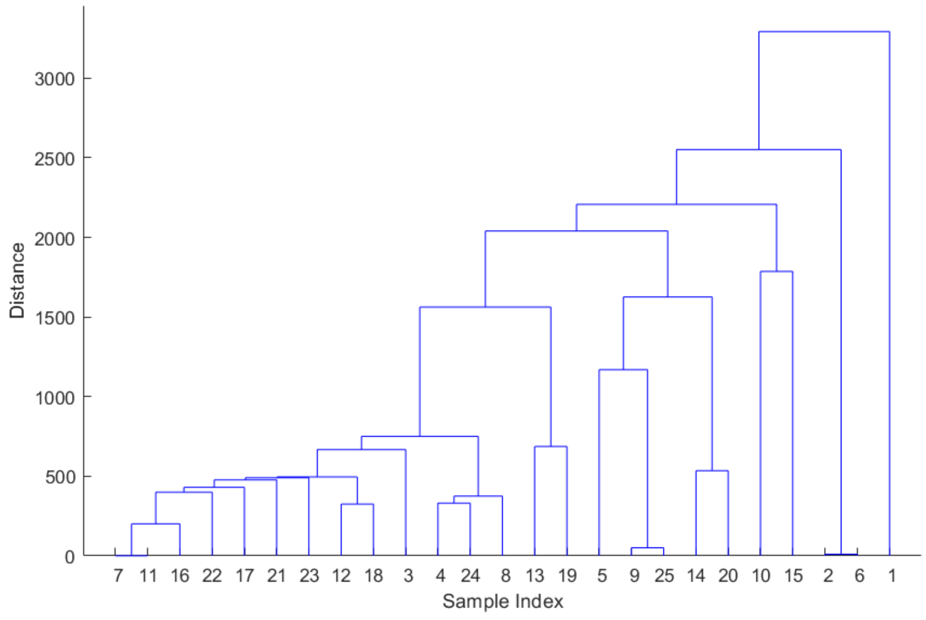


Figure 4.5: An example dendrogram. Each point starts as a single cluster and connections between them are made starting from the nearest ones until there remains only one cluster.

How merging is performed might differ from one method to another. For instance, in some variations, each merged cluster is represented as the centroid of medoid of the clusters to be merged. In single linkage method, the distance of the closest points that belong to the two clusters to be merged is considered. This, however, is observed to be highly affected by outlier data [77]. Thus, average linkage method is utilized in this thesis.

As clustering continues until there is only one cluster, another call is required to decide when to stop merging. The stopping point is either determined by inconsistency coefficient or maximum number of clusters. Inconsistent links are simply the ones that result in relatively high jumps in the merged distances. In other words, a link whose height is noticeably larger than the height of the links below it indicates that the clusters merged at this level are much farther apart than they used to be before, and this link is said to be inconsistent with the links below it. The inconsistency value is calculated by comparing the height of a link with the average height of links below it [78]. By comparing the inconsistency values to a threshold, the step to stop merging is determined, or clustering is finalized. The first link to overstep this threshold, called as α, is where merging is stopped.

4.4 Proposed Extensions on KDESOINN

4.4.1 Distance Metric

One major drawback of SOINN and KDESOINN is the use of Euclidean distance as the distance measure. Euclidean distance treats the input space as isotropic [79] and neglects any differences that might exist among different input dimensions. This problem has been discussed for fuzzy ART as well in Section 4.1.1.

To solve this, either the input signal has to be normalized or a different distance measure has to be used. As cosine distance is observed to beat Euclidean distance in prototypical architectures [80], cosine distance is utilized. The cosine distance between two vectors P_1 and P_2 can be written as [81]:

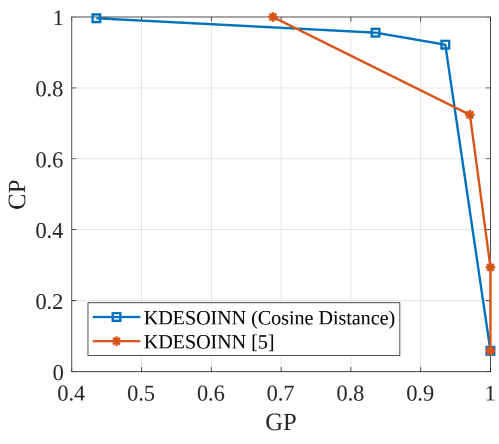
$$d_{cos}(P_1, P_2) = 1 - \frac{P_1.P_2}{\sqrt{(P_1.P_1)(P_2.P_2)}} \tag{4.5}$$

where $(.)$ shows the dot product. If two vectors are the same, cosine distance will be 0. If they are perpendicular, it will be 1.

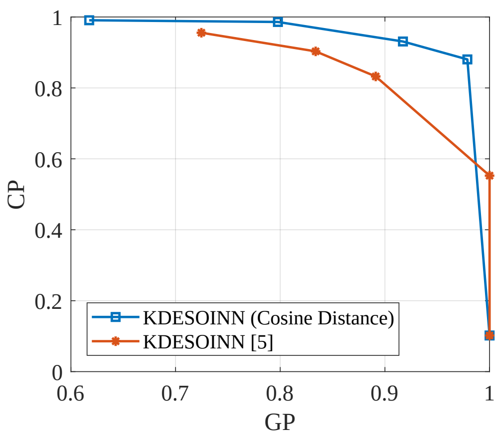
The effect of using cosine distance can be observed in Figure 4.6.

4.4.2 Covariance Calculation

If the winner node does not have any edges, $\boldsymbol{C_i}$ (given in Eq. 3.34) is not calculable and taken as zero. Hence, the addition of $\rho\gamma_i\boldsymbol{I}$ is indeed necessary to determine an approximate threshold region for a class that does not have any connections yet.



(a) Simulator data.



(b) External data set.

Figure 4.6: KDESOINN with cosine distance compared with the original KDESOINN [5] (Euclidean distance). k (of k-NN) is swept.

However, as a class grows, its range can be more safely determined by the covariance calculation in Eq. 3.33.

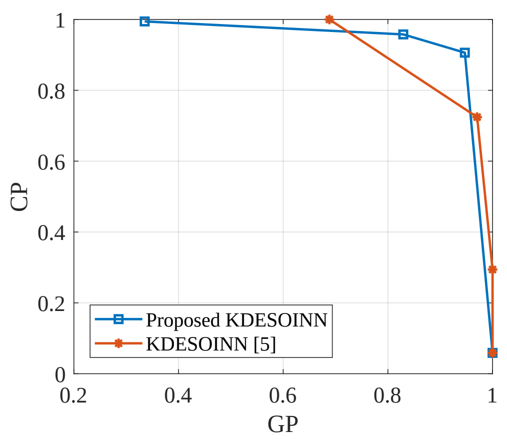
Following this, in the proposed implementation of KDESOINN, the parameter ρ is slightly dropped at each iteration to enhance the strength of $\boldsymbol{C_i}$.

The proposed KDESOINN (including both extensions) is compared to the original KDESOINN in Figure 4.7.

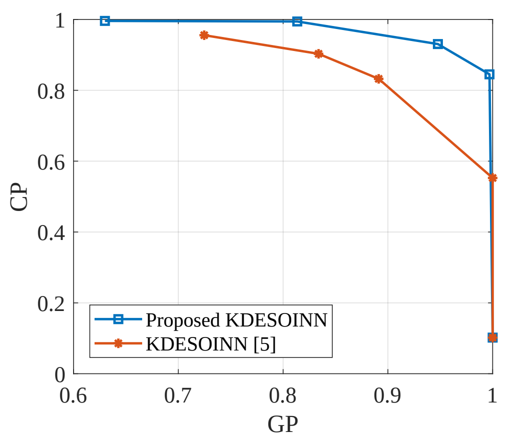
4.5 Proposed SDA Method

The proposed method utilizes a weighted Jaccard distance to determine the dissimilarity between two inputs as proposed in [55]. Both inputs are of the same form as in Eq. 3.41, one calculated from the incremental learning stage and the other is a previously known emitter type.

There are four intervals in an input vector as it can be seen in Eq. 3.41 and each one will provide its own distance. The overall Jaccard distance between two inputs x_i and x_j can be written as:



(a) Simulator data.



(b) External data set.

Figure 4.7: Proposed KDESOINN (with both extensions) compared with the original KDESOINN [5]. k (of k-NN) is swept.

$$\begin{aligned} J(x_i, x_j) &= w_{RF} J_I(RF_i, RF_j) + w_{PW} J_I(PW_i, PW_j) \\ &+ w_{PRI} J_I(PRI_i, PRI_j) + w_{DOA} J_I(DOA_i, DOA_j) \end{aligned} \tag{4.6}$$

where RF_i, PW_i, PRI_i and DOA_i shows the RF, PW, PRI and DOA intervals for the ith input and w_{RF}, w_{PW}, w_{PRI} and w_{DOA} show their corresponding weights.

As one feature can be more significant than others, instead of selecting equal weights for each feature, different weights are assigned. The concept of weighted clustering has also been experimented with K-means. The objective function of K-means is replaced by the following expression in weighted K-means [82]:

$$\sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v=1}^{V} w_v^{\beta} (y_{iv} - c_{kv})^2 \tag{4.7}$$

subject to $\sum_{v=1}^{V} w_v^{\beta} = 1$. In Eq. 4.7, S_k shows the kth cluster, c_k shows the centroid of the kth cluster, V shows the set of features, w_v shows the weight for feature v and y_i is the ith input sample. The term w_v is missing in the original K-means algorithm, which allows to give more power to the most significant feature (or less power to the least) while determining the distance.

The weight of feature v is determined by:

$$w_v^\beta = \frac{1}{\sum_{u \in V} [D_{kv}/D_{uv}]^{1/\beta - 1}} \tag{4.8}$$

where D_{kv}/D_{ku} is the ratio of within cluster separation to between cluster separation and β determines the power of this ratio on the weight assignment.

In SDA, a similar approach is followed. At each λth iteration, the feature weights are updated according to how discriminative that feature is. For a feature f, its discriminativity is defined as follows [55]:

$$dis_f = \bar{J}_{f_{within}} - \bar{J}_{f_{between}} \tag{4.9}$$

where $\bar{J}_{f_{within}}$ shows the average within cluster distance and $\bar{J}_{f_{between}}$ shows the average between cluster distance, both computed according to only one feature, f.

The more discriminative a feature is, the more weight it should have on the overall dissimilarity. Hence the weight update suggested by [55] is as follows:

$$w_f = \frac{dis_f}{\sum_{k \in F} dis_k} \tag{4.10}$$

where F is the set of features, i.e., $F = \{RF, PW, PRI, DOA\}$.

Nevertheless, during the experiments, it has been observed that using Eq. 4.10 directly causes a great imbalance between feature weights. However, it is possible that, for example, PW has not been quite discriminative until a point but it may be later on. Hence, instead of using Eq. 4.10, the weight update problem is turned into a linear optimization problem in this thesis as follows:

$$\text{Maximize} \sum_{f \in F} w_f.dis_f \text{ subject to} \tag{4.11}$$

$$\sum_{f \in F} w_f = 1 \text{ and} \tag{4.12}$$

$$lb < w_f < ub, f \in F \tag{4.13}$$

where lb and ub are the lower and upper bounds for feature weights respectively. The use of bounds ensures that no feature can converge to having zero or full effect on the distance.

After determining the distance using Eq. 4.6, an approach similar to ART networks is utilized. If the distance between a found emitter in the battle and the nearest previously known emitter type or active emitter is less than a threshold T, the emitter belongs to that type. If the distance is larger than the threshold, a new emitter type is initialized with the name $\mathcal{X}$, meaning that the type name is indeed not known, whose boundaries are determined using the characteristics of the found emitter. The list of active emitters is then extended with $\mathcal{X}$, so that its upcoming members can be compared with it and recognized. While updating a representative in the list of active emitters, its minimum and maximum limits are redetermined according to the widest range the samples of that representative has reached.

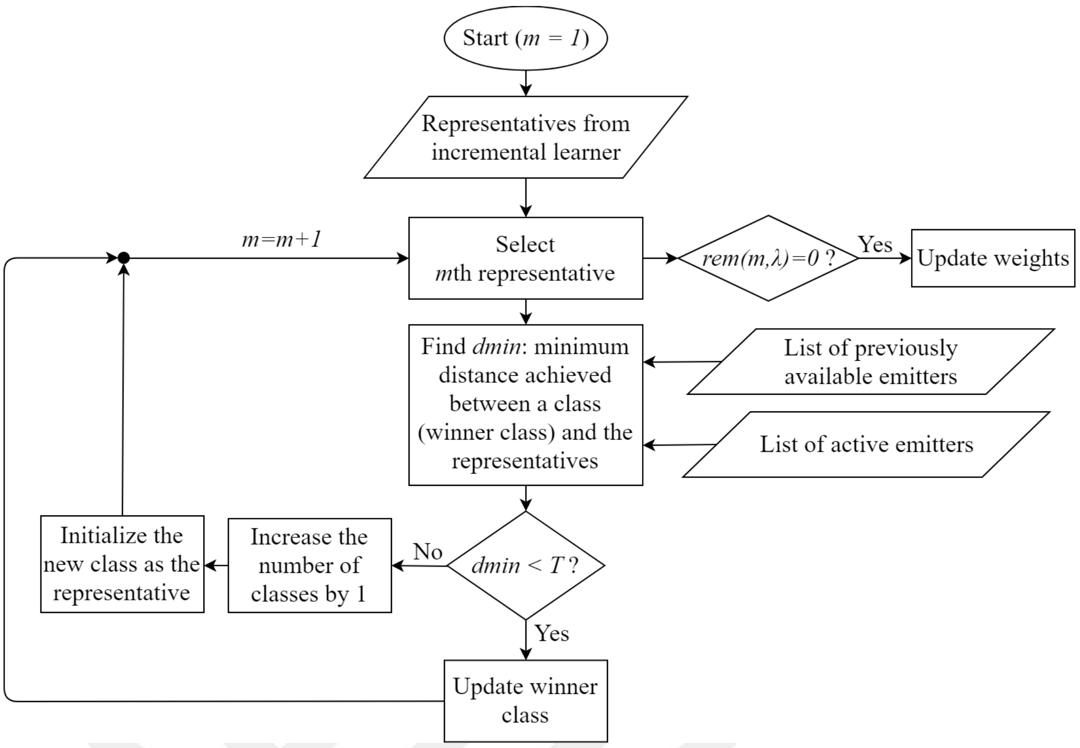
The flowchart of the proposed method is provided in Figure 4.8. The classes in the flowchart are emitter types from the previously available emitters and active emitters list.

4.6 Tests and Simulations

The experiments are performed on both simulator data and the external data set.

4.6.1 Results on Incremental Learning

As from here, the proposed versions of the incremental learner algorithms (with extensions) are utilized and compared with each other. First, the tests are performed on



Start (m = 1)
Representatives from incremental learner
m=m+1
Select mth representative
$rem(m,\lambda)$=0 ?
Yes
Update weights
Find $dmin$: minimum distance achieved between a class (winner class) and the representatives
List of previously available emitters
List of active emitters
Initialize the new class as the representative
Increase the number of classes by 1
No
$dmin$ < T ?
Yes
Update winner class

Figure 4.8: Flowchart of the proposed SDA method.

simulator data and then an external data set.

Simulator generates PDWs, but as mentioned in Figure 1.2, emitter identification works with summarized information from sPDWs. As deinterleaving is out of concept for this thesis, the ground truth labels of the PDWs are directly used while constructing sPDWs from the simulator. After sPDWs are obtained, minimum; mean and maximum values of the radar parameters are calculated to construct the vector x as shown in Eq. 3.35.

The simulator generates a whole set of mixed/interleaved PDWs at once. This set is then divided into n_b number of blocks. From each block, sPDWs are generated using the ground truth labels. Blocks are provided to the identifier one after another (Figure 4.9). An emitter might be on multiple (or all) blocks or might have started and ended in just one block.

Due to the nature of the considered problem, only growing architectures are considered. To allow growth, every algorithm uses a threshold. In fuzzy ART, vigilance parameter is directly used as the similarity threshold. In Bayesian ART, the ratio

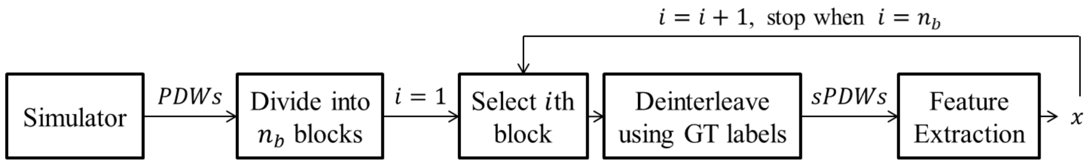


Figure 4.9: Block diagram for simulator data pre-processing. The simulator data is divided into consecutive (with respect to TOA) blocks of the same pulse length. The next block in line is deinterleaved using the ground truth (GT) labels to obtain sPDWs. The sPDWs are then subject to feature extraction to obtain the input signal x for the incremental learner.

of the determinant of the covariance between the current input and a cluster representative to the determinant of the covariance of the cluster itself is compared to a threshold. In KDESOINN, the nodes that are connected to one another with edges are considered as one cluster. The addition of edges is determined by a parameter, k, used for k-NN. In SOM, after the node weights converge, agglomerative hierarchical clustering is performed. Agglomerative clustering continues to merge clusters until there is only one cluster left. Therefore, to perform clustering, a break point to stop merging has to be determined. This cut-off point is the parameter that determines the performance of SOM.

Figure 4.10 shows the results on simulator data. In this figure, the parameter that determines the performance of an algorithm is swept to obtain a performance curve for that algorithm. The parameters used in each algorithm are shown in Table 4.1.

Table 4.1: The threshold parameters for each algorithm.

Algorithm	Parameter	Used for
Fuzzy ART	ρ	Vigilance Test
Bayesian ART	c	Vigilance Test
SOM	α	Hierarchical Clustering
KDESOINN	k	k-Nearest Neighbor

The results on the external data set are provided in Figure 4.11.

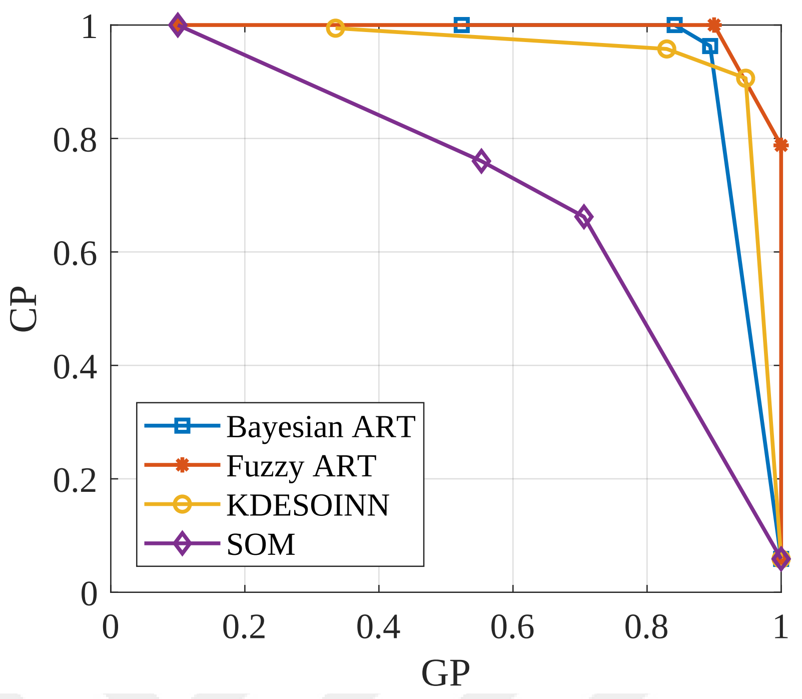


Figure 4.10: Results on simulator data set. (Note that these algorithms are based on the extensions proposed in this thesis.)

4.6.2 Results on SDA

The evaluation of SDA methods is based on IMQ. As IMQ is a measure of the resemblance of the estimated list of active emitters and the true list of active emitters, it is desired to be as high as possible. The results are as follows:

The ratio of how many samples (which are interval based here) are classified correctly is added as an evaluation criteria to support IMQ. IMQ provides a perspective in the sense that it makes it possible to comment on actually how wrong are the misclassifications. For example, although the ratio of correct classification by Jaccard distance is 88%, a few comparably distant intervals being merged together cause its IMQ metric to be less than 88%. When weight update is introduced, the ratio of correctly classified samples slightly decrease, but IMQ gets much higher. This means that even though there are misclassified intervals, these intervals were not assigned to irrelevant or distant classes, and that the final estimated list of active emitters is

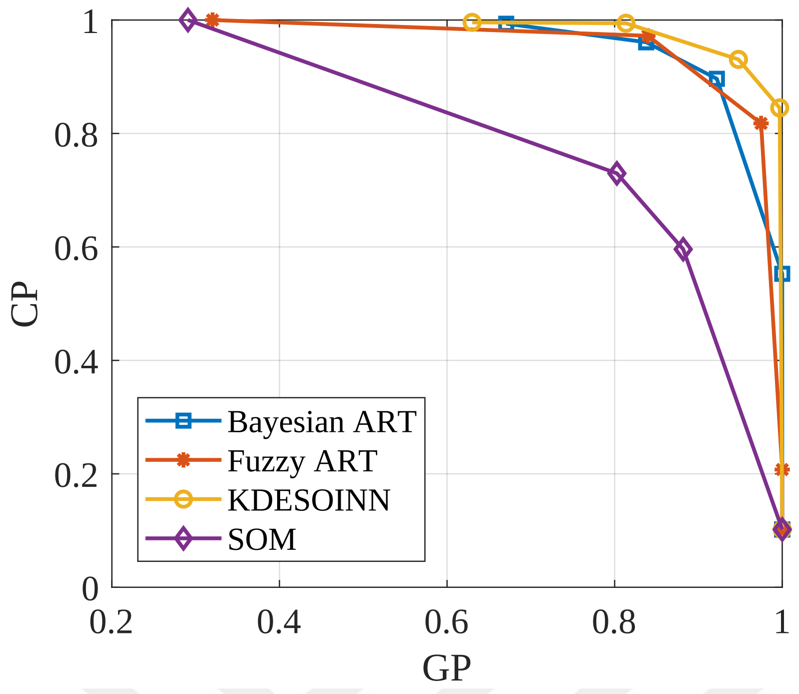


Figure 4.11: Results on external data set. (Note that these algorithms are based on the extensions proposed in this thesis.)

indeed quite close to the actual list of active emitters.

4.7 Conclusions

In all of the incremental learning methods, the relationship between cluster purity and ground truth purity is similar, which is indeed expected. The more strict the similarity search is, the more pure clusters become. However, as an algorithm goes towards searching for a 100% match between an input and a cluster representative, extra or unnecessary clusters are initialized, which decreases ground truth purity. Likewise, if only 1% match is considered to be enough, all the ground truth classes will be stored under the same cluster, which will result in a ground truth purity of 1, unfortunately the purity of the cluster itself will be much less.

It is clear that SOM has performed the worst and Bayesian ART performs similarly to

Table 4.2: SDA results with respect to various distance metrics.

Distance Metric	IMQ	Ratio of Correctly Classified Samples	Execution Time (s)
Wasserstein	70	0.62	10
Hausdorff	67	0.57	0.1
Jaccard	86	0.88	0.1
Jaccard (with weight update)	95	0.85	0.8

KDESOINN and fuzzy ART, although slightly worse in both data sets. The distance of the breakpoints of KDESOINN and fuzzy ART to the top right corner are very close, yet it is possible to say that fuzzy ART performs the best by a narrow margin on the simulator data. On the other hand, on the external data set, KDESOINN outperforms fuzzy ART by a much more noticeable margin.

Although noise is added to the simulated data, the deinterleaver is still assumed to be correct all the time. In case that the deinterleaver estimates the sPDWs inaccurately, and hence noise or outlier data is given to the incremental learner as input, KDESOINN and Bayesian ART can be more preferable compared to other methods. KDESOINN deletes inactive edges after a certain while and can utilize more than one node to represent an input cluster, thus it is capable of embracing the main structure of the input classes (Figure 3.8). Bayesian ART is also robust to outlier data as it makes use of prior probabilities while calculating the activation values of prototypes. It allows diminishing of unused prototypes (the ones that have -relatively- less input patterns categorized under them) through prior probability.

As for SDA results, the execution time of Wasserstein distance is about $\times 100$ more than Hausdorff and Jaccard distances. It does not provide an improvement on IMQ as well when compared to Jaccard distance. It could provide an improvement, however, as the underlying distributions of the intervals are not known, they are assumed to be uniform. Hausdorff distance is quite fast but can not perform as good as Jaccard distance, due to its over simplification of the distance that results in its susceptibility to outliers and underestimation of the true distance [68].

Jaccard distance performs the best when compared to other distances. With the addition of weight update (which is determined as a solution to the optimization problem given in Eq. 4.11), its performance is enhanced. As the considered optimization task is linear, the execution time, although a lot less compared to when weight update is not included, can still be considered to be tolerable for emitter identification.

CHAPTER 5

CONCLUSIONS

Emitter identification is the final and core part of electronic intelligence. It aims to correctly represent the pulse groups being received from the deinterleaver and use these representations to find the types of the pulse groups. Due to the fact that the problem requires a life-long processing of radar pulses, the concept of incremental learning is investigated. In order to account for the unknown emitter types, networks allowing growth (of clusters) are considered. The selected networks are fuzzy ART, Bayesian ART, SOM and KDESOINN.

Two extensions are proposed for fuzzy ART. In fuzzy ART, vigilance test and activation calculation are through different functions, although both try to measure the similarity between the input vector and a representative. In the proposed implementation of fuzzy ART, vigilance test and activation is united through one function that is the Jaccard index, which halved the computation time of fuzzy ART. Secondly, the normalization of fuzzy ART, which is dividing the input vector by its norm, is replaced by normalization with a vector constructed according to a priori knowledge in order to be able to treat input dimensions differently. This extension increased both cluster and ground truth purity.

In Bayesian ART, a vigilance test that takes into account the similarity between the input vector and the candidate prototype is proposed instead of only checking the hypervolume of the prototype. The results show an increase in both cluster and ground truth purity.

The Euclidean distance in KDESOINN is replaced by cosine distance as the input dimensions, i. e. radar parameters, are of very different magnitudes. Secondly, the smoothing parameter ρ is proposed to be monotonically decreasing as the more samples are received, the more reliable covariance calculation becomes. Both extensions are observed to enhance KDESOINN performance.

The later task is to compare the received representatives with previously available emitter types to construct the list of active emitters. For this purpose, as radar parameters are generally interval based, a symbolic analysis method based on ART structure is proposed with Jaccard index. In order to evaluate the estimated list of active emitters, a metric called as interval match quality is proposed. This metric takes into account the shifts in the final ranges of estimated emitter parameters caused by misclassification as well, instead of counting how many interval based elements are classified correctly.

The results on incremental learning show that KDESOINN (with proposed extensions) has performed the best, while SOM has performed the worst. For symbolic data analysis, different distance metrics (Hausdorff and Wasserstein) are compared with Jaccard index, proving that Jaccard index performs the best for emitter identification.

REFERENCES

[1] L. Billard and E. Diday, “Symbolic data analysis: Definition and examples,” 01 2004.

[2] D. Adamy, *EW 101: A First Course in Electronic Warfare*. Artech House radar library, 685 Canton Street, Norwood, MA 02062: Artech House, 2001.

[3] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: Incremental classifier and representation learning,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, 2016.

[4] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience, 2 ed., November 2000.

[5] Y. Nakamura and O. Hasegawa, “Nonparametric density estimation based on self-organizing incremental neural network for large noisy data,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 8–17, Jan 2017.

[6] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” *CoRR*, vol. abs/1606.04671, 2016.

[7] G. A. Carpenter, S. Grossberg, and D. B. Rosen, “Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system,” *Neural Networks*, vol. 4, no. 6, pp. 759 – 771, 1991.

[8] B. Vigdor and B. Lerner, “The bayesian artmap,” *IEEE Transactions on Neural Networks*, vol. 18, pp. 1628–1644, Nov. 2007.

[9] R. G. Wiley, *ELINT: The Interception and Analysis of Radar Signals*. Artech House, 2 ed., 2006.

[10] T. Pietkiewicz and B. Wajszczyk, “Fusion of identification information from elint-esm sensors,” p. 1, 04 2018.

[11] C. Lau and I. N. N. Council, *Neural networks: theoretical foundations and analysis*. IEEE Press Selected Reprint Series, IEEE Press, 1992.

[12] A. De Martino, *Introduction to Modern EW Systems*. Artech House radar library, Artech House, 2 ed., 2012.

[13] X. Xu, W. Wang, and W. Jianhong, "A three-way incremental-learning algorithm for radar emitter identification," *Frontiers of Computer Science*, vol. 10, 12 2015.

[14] N. Burlutskiy, M. Petridis, A. Fish, A. Chernov, and N. Ali, "An investigation on online versus batch learning in predicting user behaviour," pp. 135–149, 11 2016.

[15] I. Guyon, "A scaling law for the validation-set training-set size ratio," in *AT T Bell Laboratories*, 1997.

[16] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European Symposium on Artificial Neural Networks (ESANN)*, (Bruges, Belgium), 2016.

[17] *Electronic Warfare and Radar Systems Engineering Handbook*. Naval Air Warfare Center Weapons Division, 2013.

[18] K. Gençol, *New methods for radar emitter identification*. PhD thesis, Eskişehir Technical University, 2015.

[19] M. Jankiraman, N. Willis, and H. Griffiths, *Design of Multi-Frequency CW Radars*. Electromagnetics and Radar, Institution of Engineering and Technology, 2007.

[20] W. A. Metz, "Electronic warfare receiver resource management and optimization," 2016.

[21] D. K. Barton, *Radar System Analysis and Modeling*. No. 1. c. in Artech House radar library, Artech House, 2005.

[22] K. P. Mason, "A knowledge-based strategy for the re-association of fragmented sensor reports," in *Proceedings of the 8th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*,

IEA/AIE '95, (Newark, NJ, USA), pp. 563–567, Gordon and Breach Science Publishers, Inc., 1995.

[23] M. Hoffman, D. Steinley, and M. J. Brusco, "A note on using the adjusted rand index for link prediction in networks," *Social Networks*, vol. 42, pp. 72 – 79, 2015.

[24] H. Venkateswara Reddy, P. Agrawal, and S. Viswanadha Raju, "Data labeling method based on cluster purity using relative rough entropy for categorical data clustering," in *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 500–506, Aug 2013.

[25] L. Jain, M. Sato-Ilic, M. Virvou, G. A. Tsihrintzis, V. Balas, and C. Abeynayake, *Computational Intelligence Paradigms, Innovative Applications*, vol. 137. 01 2008.

[26] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54 – 71, 2019.

[27] S. Sarwar, A. Ankit, and K. Roy, "Incremental learning in deep convolutional neural networks using partial network sharing," 12 2017.

[28] Z. Chen and B. Liu, "Lifelong machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, p. 55, 08 2018.

[29] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, 09 2017.

[30] S. Fong, Z. Luo, and B. W. Yap, "Incremental learning algorithms for fast classification in data stream," in *2013 International Symposium on Computational and Business Intelligence*, pp. 186–190, Aug 2013.

[31] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," *CoRR*, vol. abs/1807.09536, 2018.

[32] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Networks*, vol. 19, no. 1, pp. 90 – 106, 2006.

[33] L. Billard and E. Diday, “Symbolic data analysis: Definition and examples,” 2004.

[34] S. Cho, J. A. Reggia, and M. Jang, “A learning sensorimotor map of arm movements: a step toward biological arm control,” in *Neural Systems for Control* (O. Omidvar and D. L. Elliott, eds.), pp. 61 – 86, San Diego: Academic Press, 1997.

[35] S. Kaski and T. Kohonen, “Winner-take-all networks for physiological models of competitive learning,” *Neural Networks*, vol. 7, pp. 973–984, 1994.

[36] R. J. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*. New York, NY, USA: John Wiley & Sons, Inc., 1991.

[37] P. J. Braspenning, F. Thuijsman, and A. J. M. M. Weijters, eds., *Artificial Neural Networks: An Introduction to ANN Theory and Practice*, (Berlin, Heidelberg), Springer-Verlag, 1995.

[38] L. E. Brito da Silva, I. El-Nabarawy, and D. Wunsch, “A survey of adaptive resonance theory neural network models for engineering applications,” 05 2019.

[39] E. Gómez-Sánchez, Y. Dimitriadis, J. Manuel Cano-izquierdo, A. Member, and J. Lpez-coronado, “Artmap: Use of mutual information for category reduction in fuzzy artmap,” 11 2002.

[40] R. D. Luce and W. E. Edwards, “The derivation of subjective scales from just noticeable differences.,” *Psychological review*, vol. 65 4, pp. 222–37, 1958.

[41] T. Kohonen, “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, Sep. 1990.

[42] X. Lin, D. Soergel, and G. Marchionini, “A self-organizing semantic map for information retrieval,” in *SIGIR*, 1991.

[43] T. Kohonen, “Essentials of the self-organizing map,” *Neural Networks*, vol. 37, pp. 52 – 65, 2013. Twenty-fifth Anniversay Commemorative Issue.

[44] B. Fritzke, “A growing neural gas network learns topologies,” in *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS’94, (Cambridge, MA, USA), pp. 625–632, MIT Press, 1994.

[45] F. Shen and O. Hasegawa, “A fast nearest neighbor classifier based on self-organizing incremental neural network,” *Neural Networks*, vol. 21, no. 10, pp. 1537 – 1547, 2008. ICONIP 2007.

[46] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, “Background and foreground modeling using nonparametric kernel density estimation for visual surveillance,” *Proceedings of the IEEE*, vol. 90, pp. 1151–1163, July 2002.

[47] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. Berlin, Heidelberg: Springer-Verlag, 1985.

[48] S. Mahmod, “Deinterleaving pulse trains with dbscan and fart,” 2019.

[49] E. Granger, M. A. Rubin, S. Grossberg, and P. Lavoie, “A what-and-where fusion neural network for recognition and tracking of multiple radar emitters,” *Neural Networks*, vol. 14, no. 3, pp. 325 – 344, 2001.

[50] E. Granger, Y. Savaria, P. Lavoie, and M.-A. Cantin, “A comparison of self-organizing neural networks for fast clustering of radar pulses,” *Signal Processing*, vol. 64, pp. 249–269, 02 1998.

[51] E. Granger, M. A. Rubin, S. Grossberg, and P. Lavoie, “Classification of incomplete data using the fuzzy artmap neural network,” in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 6, pp. 35–40 vol.6, July 2000.

[52] Ching-Sung Shieh and Chin-Teng Lin, “A vector neural network for emitter identification,” *IEEE Transactions on Antennas and Propagation*, vol. 50, pp. 1120–1127, Aug 2002.

[53] C.-M. Lin, Y.-M. Chen, and C.-S. Hsueh, “A self-organizing interval type-2 fuzzy neural network for radar emitter identification,” *International Journal of Fuzzy Systems*, vol. 16, pp. 20–30, 03 2014.

[54] J. Dudczyk, “Radar emission sources identification based on hierarchical agglomerative clustering for large data sets,” *Journal of Sensors*, vol. 2016, p. 9, 04 2016.

[55] X. Xu, J. Lu, and W. Wang, “Hierarchical clustering of complex symbolic data and application for emitter identification,” *Journal of Computer Science and Technology*, vol. 33, pp. 807–822, 07 2018.

[56] W. Zhu, M. Li, W. Chen, and X. Ran, “Radar emitter recognition based on transfer learning,” *DEStech Transactions on Computer Science and Engineering*, 01 2018.

[57] W. Zhu, M. Li, and C. Zeng, “Research on online learning of radar emitter recognition based on hull vector,” in *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, pp. 328–332, June 2017.

[58] J. Vesanto and E. Alhoniemi, “Clustering of the self-organizing map,” *IEEE Transactions on Neural Networks*, vol. 11, pp. 586–600, May 2000.

[59] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, “Zero-shot learning through cross-modal transfer,” in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 935–943, Curran Associates, Inc., 2013.

[60] T. J. Draelos, N. E. Miner, C. C. Lamb, C. M. Vineyard, K. D. Carlson, C. D. James, and J. B. Aimone, “Neurogenesis deep learning,” *CoRR*, vol. abs/1612.03770, 2016.

[61] E. Triantafillou, R. S. Zemel, and R. Urtasun, “Few-shot learning through an information retrieval lens,” *CoRR*, vol. abs/1707.02610, 2017.

[62] F. d. A. T. de Carvalho, R. M. C. R. de Souza, M. Chavent, and Y. Lechevallier, “Adaptive hausdorff distances and dynamic clustering of symbolic interval data,” *Pattern Recogn. Lett.*, vol. 27, pp. 167–179, Feb. 2006.

[63] A. Irpino and R. Verde, *A New Wasserstein Based Distance for the Hierarchical Clustering of Histogram Symbolic Data*, pp. 185–192. 12 2005.

[64] A. P. Reynolds, G. Richards, B. Iglesia, and V. Rayward-Smith, “Clustering rules: A comparison of partitioning and hierarchical clustering algorithms,” *J. Math. Model. Algorithms*, vol. 5, pp. 475–504, 12 2006.

[65] A. Gardner, J. Kanno, C. A. Duncan, and R. Selmic, “Measuring distance between unordered sets of different sizes,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 137–143, June 2014.

[66] J. Ouyang and I. Sethi, “A novel distance measure for interval data.,” pp. 49–58, 01 2007.

[67] M. Chavent and Y. Lechevallier, “Dynamical clustering of interval data: Optimization of an adequacy criterion based on hausdorff distance,” *Journal of Classification*, 01 2002.

[68] A. Fischer, R. Plamondon, Y. Savaria, K. Riesen, and H. Bunke, “A hausdorff heuristic for efficient computation of graph edit distance,” in *Structural, Syntactic, and Statistical Pattern Recognition* (P. Fränti, G. Brown, M. Loog, F. Escolano, and M. Pelillo, eds.), (Berlin, Heidelberg), pp. 83–92, Springer Berlin Heidelberg, 2014.

[69] A. Irpino, R. Verde, and Y. Lechevallier, *Dynamic clustering of histograms using Wasserstein metric*, pp. 869–876. 01 2006.

[70] C. Frogner, C. Zhang, H. Mobahi, M. Araya-Polo, and T. Poggio, “Learning with a wasserstein loss,” in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS’15, (Cambridge, MA, USA), pp. 2053–2061, MIT Press, 2015.

[71] F. Çoğun, F. Altiparmak, and H. S. Balaban, “Queue-based sequential clustering method for interval datasets,” in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, May 2018.

[72] D. Larsson, “Aravq for discretization of radar data : An experimental study on real world sensor data,” 2015.

[73] A. Erol, O. Can, and A. A. Alatan, “Adaptive classification of radar pulses with improved fuzzy artmap,” in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, May 2018.

[74] A. Gupta and N. Sardana, “Significance of clustering coefficient over jaccard index,” pp. 463–466, 08 2015.

[75] A. K. Jain, “Data clustering: 50 years beyond k-means,” in *ECML/PKDD*, 2008.

[76] D. Beeferman and A. Berger, “Agglomerative clustering of a search engine query log,” in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’00, (New York, NY, USA), pp. 407–416, ACM, 2000.

[77] G. Karypis, Eui-Hong Han, and V. Kumar, “Chameleon: hierarchical clustering using dynamic modeling,” *Computer*, vol. 32, pp. 68–75, Aug 1999.

[78] M. Ghasemigol, H. sadoghi yazdi, and R. Monsefi, “A new hierarchical clustering algorithm on fuzzy data (fhca),” *International Journal of Computer and Electrical Engineering-IJCEE*, vol. 2, pp. 134–, 02 2010.

[79] Y. Raykov, A. Boukouvalas, F. Baig, and M. A. Little, “What to do when k-means clustering fails: A simple yet principled alternative algorithm,” *PLOS ONE*, vol. 11, p. e0162259, 09 2016.

[80] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 4077–4087, Curran Associates, Inc., 2017.

[81] G. Qian, S. Sural, Y. Gu, and S. Pramanik, “Similarity between euclidean and cosine angle distance for nearest neighbor queries,” in *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC ’04, (New York, NY, USA), pp. 1232–1237, ACM, 2004.

[82] R. Amorim and P. Komisarczuk, “On initializations for the minkowski weighted k-means,” vol. 7619, pp. 45–55, 10 2012.

APPENDIX A

SIMULATOR

The output of the constructed test platform for emitter identification is in the form of PDWs. For each radar feature (RF, PW, PRI and DOA), there exist possible operation modes. For instance, constant PRI mode generates PRI values centered around a point with 1% variation whereas jittered mode may cause around 20% variation among the PRI values. Therefore, the test platform covers all of the operation modes of the features.

The proposed platform generates random scenarios: In each scenario, the number of radars is selected randomly between 4 to 20. Each radar contains a sequence for all of its features, with all sequences having the same length. Each feature is provided with a random operation mode, independent from one another. To construct a sequence for a feature, a center value is assigned randomly within an a priori expected range of the feature. This range is [1000, 40000] MHz for RF, [1, 250] μs for PRI, [1, 100] μs for PW and [1, 360] degrees for DOA.

For any feature (if applicable), if the operation mode is

- Constant: 1% jitter is added around the center value. For instance, for PRI:

$$PRI_{min} = PRI_c - PRI_c \times jitter \tag{A.1}$$

$$PRI_{max} = PRI_c + PRI_c \times jitter \tag{A.2}$$

  where $jitter$ is a random number between 0 and 1 and PRI_c shows the center PRI value. While constructing the PRI sequence, the values are selected randomly within the range $[PRI_{min}, PRI_{max}]$.

- Jittered: Up to 10% (random) jitter is added.
- Constant / Random Staggered: Between 5% and 40% (random) jitter is added. A random number between 2 and 9 is selected to stand for the number of staggers. The stagger levels are chosen randomly between the minimum and maximum possible values (calculated according to Eq. A.1). Each stagger level is subject to 1% jitter on its own. If the mode is constant staggered, a constant switching pattern exists between the levels, otherwise levels appear randomly.
- Dwell & Switch: Between 5% and 40% (random) jitter is added. A random number between 2 and 9 is selected to stand for the number of levels. These levels are chosen randomly between the minimum and maximum possible values. Another random number between 4 and 10 is selected for each level which represents their repetition count.

The output of the constructed test platform is a matrix composed of the sequences of each feature.